Elektrobit

EB tresos® AutoCore Generic 8
LIN Stack documentation

product release 8.8.7

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

https://www.elektrobit.com/support

Legal disclaimer

Confidential information.

Table of Contents

1. Overview of EB tresos AutoCore Generic 8 LIN Stack documentation

Welcome to the EB tresos AutoCore Generic 8 LIN Stack (ACG8 LIN Stack) product documentation.

This document provides:

- Chapter 2, “Supported features“: list of features supported by the ACG8 LIN Stack
- Chapter 3, “ACG8 LIN Stack release notes“: release notes for the ACG8 LIN Stack modules
- Chapter 4, “ACG8 LIN Stack user guide“: background information and instructions
- Chapter 5, “ACG8 LIN Stack module references“: information about configuration parameters and the application programming interface

2. Supported features

2.1. Supported LinIf features

- **Support for post-build:** Support for handling post-build loadable configuration.
- **Support for error/success status callouts:** Callouts can be configured to report a specific error status or success status from the driver. The status can be included in the callout prototype.
- **Dem/Det reporting:** LinIf can report the driver status for a specific frame directly to Dem or Det. For the reporting to Dem, you can additionally select a debouncing method.
- **End of schedule notification:** LinIf can notify the user that a schedule table has ended, i.e. that the last entry in the schedule table was executed, through a specifically configured callout.
- **CDD support:** LinIf can notify a CDD about
  - state changes, replacing LinSM
  - PDU indications, replacing PduR.
- **Bus mirroring:** LinIf can mirror a LIN channel to the Mirror module by forwarding all received/transmitted frames to the Mirror module via the Mirror_ReportLinFrame() API. Diagnostic frames are included. Alternatively, mirroring to a CDD is possible. The bus mirroring feature supports multi-core usage.
- **Support for multiple AUTOSAR LIN Driver versions:** LinIf supports LIN Drivers of different AUTOSAR versions. The used LIN Driver version can be selected with the `LinIfLinDriverAPI` parameter.

2.2. Supported LinSM features

- **Support for post-build:** Support for handling post-build loadable configuration.
- **Support for BSW distribution:** Support for the interaction of network-specific <Net>SM modules mapped to dedicated cores with a central ComM.

3. ACG8 LIN Stack release notes

3.1. Overview

This chapter provides the ACG8 LIN Stack product specific release notes. General release notes that are applicable to all products are provided in the EB tresos AutoCore Generic documentation. Refer to the general release notes in addition to the product release notes documented here.

3.2. Scope of the release

3.2.1. Configuration tool

Your release of EB tresos AutoCore is compatible with the release of the EB tresos Studio configuration tool:

- EB tresos Studio: 29.2.0 b220916-0321

3.2.2. AUTOSAR modules

The following table lists the AUTOSAR modules that are part of this ACG8 LIN Stack release.

Module name	AUTOSAR version and revision	SWS version and revision	Module version	Supplier
LinIf	4.0.3 []	4.0.0 [0000]	5.8.38	Elektrobit Automotive GmbH
LinSM	4.0.3 []	1.3.0 [0000]	3.4.27	Elektrobit Automotive GmbH

Table 3.1. Hardware-Independent Modules specified by the AUTOSAR standard

3.2.3. EB (Elektrobit) modules

The following table lists all modules which are part of this release but are not specified by the AUTOSAR standard. These modules include tooling developed by EB or they may hold files shared by all other modules.

Module name	Module version	Supplier
No EB modules available		

Table 3.2. Modules not specified by the AUTOSAR standard

3.2.4. MCAL modules and EB tresos AutoCore OS

For information about MCAL modules and OS, refer to the respective documentation, which is available as PDF at `$TRESOS_BASE/doc/3.0_EB_tresos_AutoCore_OS` and `$TRESOS_BASE/doc/5.0_MCAL_modules`[1]. It is also available in the online help in EB tresos Studio. Browse to the folders `EB tresos AutoCore OS` and `MCAL modules`.

3.3. Module release notes

3.3.1. LinIf module release notes

- AUTOSAR R4.0 Rev 3
- AUTOSAR SWS document version: 4.0.0
- Module version: 5.8.38.B567464
- Supplier: Elektrobit Automotive GmbH

3.3.1.1. Change log

This chapter lists the changes between different versions.

Module version 5.8.38
2022-09-16

- ASCLINIF-1340 Fixed known issue: LinIf slave node does not go to sleep after bus idle timeout
- Node response tolerance 0-40% for SAE standard (instead of ISO 40% fixed)

[1] `$TRESOS_BASE` is the location at which you installed EB tresos Studio.

- Updated requirement Id format in module documentation and source code tracing comments. Note: This does not change the Baseline, nor functionality.

Module version 5.8.37
2022-08-19

- Add Tx and Rx Bus-Adapter user specific callout functions

Module version 5.8.36
2022-06-10

- ASCLINIF-1310 Fixed known issue: Compilation error occurs due to inclusion of inexistent header

Module version 5.8.35
2022-05-13

- ASCLINIF-1300 Fixed known issue: LinIf_Cbk.h singleton inclusion (by Lin driver) causes compiler warning/error

Module version 5.8.34
2022-03-18

- Internal module improvement. This module version update does not affect module functionality

Module version 5.8.33
2022-02-18

- ASCLINIF-1294 Fixed known issue: Det is wrongfully called for LIN_TX_HEADER_ERROR and LIN_RX_ERROR
- Extended configurable user callout to SRF and MFR

Module version 5.8.32
2022-01-28

- Internal module improvement. This module version update does not affect module functionality

Module version 5.8.31
2021-11-12

- ASCLINIF-1281 Fixed known issue: LinIf_Cbk exported functions not available to some drivers

Module version 5.8.30
2021-10-08

- ASCLINIF-1276 Fixed known issue: LinIf Slave is affected by shared data race

Module version 5.8.29
2021-09-17

- Internal module improvement. This module version update does not affect module functionality

Module version 5.8.28
2021-08-20

- ASCLINIF-1272 Fixed known issue: End of schedule notification is not called if retrieving data of the last frame failed
- ASCLINIF-1263 Fixed known issue: LinIf_SetPIDTable() and LinIf_SetConfiguredNAD() fail if Det is enabled and masters are configured

Module version 5.8.27
2021-06-25

- Added LIN Slave support

Module version 5.8.26
2021-05-28

- Internal module improvement. This module version update does not affect module functionality

Module version 5.8.25
2021-04-30

- Internal module improvement. This module version update does not affect module functionality

Module version 5.8.24
2021-04-09

- Internal module improvement. This module version update does not affect module functionality

Module version 5.8.23
2021-03-05

- Internal module improvement. This module version update does not affect module functionality

Module version 5.8.22
2021-01-22

- ASCLINIF-1241 Fixed known issue: Duplicate frame priority possible

Module version 5.8.21
2020-12-18

- ASCLINIF-1237 Fixed known issue: Transceiver function list is wrongly populated in LinIf_Cfg.c

Module version 5.8.20
2020-10-23

- Increased upper-bound configuration limit of the LinTpP2Max and LinTpP2Timing parameters

Module version 5.8.19
2020-09-25

- Internal module improvement. This module version update does not affect module functionality

Module version 5.8.18
2020-07-31

- ASCLINIF-1218 Fixed known issue: LinIfEbGeneralBswmdImplementation raises an error if transceiver support is disabled

Module version 5.8.17
2020-06-19

- ASCLINIF-1210 Fixed known issue: LinIf switches to operational before time
- Schedule table switch behavior when same schedule table is called refined

Module version 5.8.16
2020-04-24

- ASCLINIF-1207 Fixed known issue: LinIf confirms sleep to LinSM even though a CDD is configured

Module version 5.8.15
2020-03-25

- ASCLINIF-1203 Fixed known issue: Wakeup during sleep transition does not work as expected for ASR 4.2.2 and above drivers

Module version 5.8.14
2020-02-21

- ASCLINIF-1191 Fixed known issue: LinIf does not confirm a schedule switch to NULL_SCHEDULE caused by a sleep request

Module version 5.8.13
2020-01-24

- ASCLINIF-1189 Fixed known issue: Transceiver list is wrongly populated in LinIf_Macros.m

Module version 5.8.12
2019-12-06

- ASCLINIF-1187 Fixed known issue: The LinTp_GetAvailablePduRTxBufferLength does not initialize pduInfo.SduDataPtr

Module version 5.8.11
2019-11-08

- Internal module improvement. This module version update does not affect module functionality

Module version 5.8.10
2019-10-11

- ASCLINIF-1165 Fixed known issue: Module configuration pointer access occurs before checking for uninitialized access of the function

Module version 5.8.9
2019-09-06

- Add 4.0 and 4.2 Lin driver initialization support

Module version 5.8.8
2019-07-12

- ASCLINIF-1150 Fixed known issue: LinIf_ScheduleRequest uses LinIf Channel ID as ComM Channel ID
- ASCLINIF-1155 Fixed known issue: End of Schedule Notification erroneously called before the last entry's status check

Module version 5.8.7
2019-06-14

- ASCLINIF-1136 Fixed known issue: LinTp does not notify PduR that functional/physical transmission was aborted because schedule table change failed

Module version 5.8.6
2019-05-17

- Internal module improvement. This module version update does not affect module functionality

Module version 5.8.5
2019-04-18

- ASCLINIF-1130 Fixed known issue: Frame reporting to Mirror during transmission non-functional

Module version 5.8.4
2019-03-22

- ASCLINIF-1127 Fixed known issue: LinIf accesses the post-build configuration without checking the channel ID

Module version 5.8.3
2019-02-15

- Internal module improvement. This module version update does not affect module functionality
- ASCLINIF-1119 Fixed known issue: LinTp_Transmit()/LinIf_Transmit() do not notify the upper layer if the Lin channel is in NO_COMM.

Module version 5.8.2
2019-01-25

- Added Support for NMoE (BusMirroring).

Module version 5.8.1
2018-12-21

- ASCLINIF-1112 Fixed known issue: Symbolic name values for LinIfChannels are erroneously taken from ComM.

Module version 5.8.0
2018-10-26

- ASCLINIF-1101 Fixed known issue: LinIf assigns slave-to-slave frames to incorrect slots

- Changed LinIf APIs incorrectly expecting ComM handle IDs

Module version 5.7.5
2018-08-24

- Added support for forwarding the status from `Lin_GetStatus()` to the user callout
- Added support for `Lin Confirmation Notification` and `LIN_RX_NO_REPONSE` handling in the user callout

Module version 5.7.4
2018-06-22

- Added support referenceable NULL_SCHEDULE LinIfScheduleTable

Module version 5.7.3
2018-05-25

- Added support for configurable upper layer

Module version 5.7.2
2018-04-20

- Add support for UINT32 PduLengthType.
- Added support for custom end-of-schedule notifications

Module version 5.7.1
2017-09-22

- Internal module improvement. This module version update does not affect module functionality

Module version 5.7.0
2017-07-28

- Fine grained DEM reporting

- Comply to MISRA-C:2012

Module version 5.6.3
2017-06-30

Module version 5.6.2
2017-06-02

Module version 5.6.1
2017-05-05

- ASCLINIF-1041 Fixed known issue: LinIf_LinDriverConfig[ ] is generated empty if Lin configuration name is not LinGlobalConfig_0
- ASCLINIF-1042 Fixed known issue: If the VendorApiInfix parameter is not present in the Lin driver, the LinIf will not generate
- ASCLINIF-1043 Fixed known issue: If LinIfLinDriverAPI is 'REV42' and LinIfCheckWakeupSupported is not activated, LinIf_LinDriverWakeupIntFctPtrType is not available

Module version 5.6.0
2017-03-31

- Internal module improvement. This module version update does not affect module functionality
- Add proper name mangling for header files and API functions of Lin and LinTrcv
- Implement Lin transceiver support

Module version 5.5.0
2017-03-10

- Internal module improvement. This module version update does not affect module functionality
- Implement support for 4.2.x Lin drivers

Module version 5.4.9
2017-02-03

- Internal module improvement. This module version update does not affect module functionality

Module version 5.4.8
2016-11-04

- Internal module improvement. This module version update does not affect module functionality

Module version 5.4.7
2016-09-09

- ASCLINIF-1005 Fixed known issue: Config parameter NumberOfRespPendingFrames is used in a wrong way. Decrement NumberOfRespPendingFrames by one in order to keep the same (erroneous) behavior as before.

Module version 5.4.6
2016-08-05

- ASCLINIF-1004 Fixed known issue: NRC response pending frame does not restart P2 timer

Module version 5.4.5
2016-05-25

- Internal module improvement. This module version update does not affect module functionality

Module version 5.4.4
2016-02-05

- ASCLINIF-990 Fixed known issue: Nested MemMap section if TS_MERGED_COMPILE is activated
- ASCLINIF-991 Fixed known issue: LinIfSupplierId cannot be set to 32767
- Added support for Debug & Trace with custom header file configurable via parameter `BaseDbgHeaderFile`

Module version 5.4.3
2015-11-06

- Internal module improvement. This module version update does not affect module functionality

Module version 5.4.2
2015-06-19

- Fixed error reported by broken ENABLE xdm check of the LinIfCollisionResolvingRef parameter
- Adapted source code comments with ReqM2 tags to conventions
- Removed misra deviation comment 19.1 from source code

Module version 5.4.1
2015-02-20

- Removed configuration parameter LinIfTrcvWakeupNotification (LINIF048_Conf)
- Changed parameter range for LinTpP2Timing, LinTpP2Max, LinIfFunctionId
- Modified LinIf to cancel a go-to-sleep command request if wakeup is requested before the go-to-sleep command is transmitted
- Added configuration check for maximum Pdu length
- Modified LinIf to call LinSM_ScheduleRequestConfirmation() even if the current run continuous schedule table is requested

Module version 5.4.0
2014-10-03

- ASCLINIF-930 Fixed known issue: If more than 255 unconditional frames are configured, a schedule table might process an unexpected frame
- Added an optional callout which is called in case of Lin bus errors for user error handling
- ASCLINIF-939 Fixed known issue: It is not possible to send MRF and receive SRF frames without LinTp
- Removed obsolete legacy symbolic name values
- ASCLINIF-946 Fixed known issue: LinIf_GotoSleep, LinIf_Wakeup, LinIf_ScheduleRequest may access configuration data of not initialized module

Module version 5.3.3
2014-04-25

- Removed xdm check which verifies that `LinIfEntryIndex`must start from `0`and be consecutive within one schedule table
- Added xdm check which verifies that `LinIfDelay`is bigger than the maximum frame transmission duration + `LinIfJitter`

- ASCLINIF-909 Fixed known issue: LinIf may call Lin API functions with an incorrect channel ID if `LinIfMapChannelIdDirect`is set to true
- ASCLINIF-913 Fixed known issue: LinIf may call ComM API functions with an incorrect channel ID if `LinIfMapComMChannelIdDirect`is set to true
- ASCLINIF-912 Fixed known issue: LinIf BSWMD is generated with invalid information causing RTE to report an error
- ASCLINIF-923 Fixed known issue: Build error due to missing file `LinIf/LinTp_PBcfg.c`if code generation for LinIf/LinTp is disabled and only post-build configuration is compiled

Module version 5.3.2
2013-10-11

- Removed compiler warning about unused variable `ScheduleChange`if `LINTP_SCHEDULE_CHANGE_DIAG_API == STD_OFF`
- Added defensive programming instrumentation for unreachable code fragments
- ASCLINIF-837 Fixed known issue: Physical transmission might not properly abort if a new physical transmission is invoked on the same channel
- ASCLINIF-838 Fixed known issue: `LinTp_Transmit()`is rejected if a previous transmission has been requested on the same LIN channel, but the `LinIf_Mainfunction()`has not executed in between these requests
- ASCLINIF-836 Fixed known issue: LinTp does not expect response for user-defined diagnostic messages
- Removed compiler warning about unused variable `invalidWakeupSource`if `LINIF_DEV_ERROR_DETECT == STD_OFF`
- Added xdm check which verifies that LinSM confirmation timeout is greater than the time it takes to execute a goto-sleep command
- ASCLINIF-853 Fixed known issue: A compiler error occurs if `PbCfgM`is used for passing a post-build time configuration to LinTp, but not to LinIf
- ASCLINIF-856 Fixed known issue: `LinIf_Init()`uses const void* for post-build config instead of `const LinIf_ConfigType*`
- Convert enum type definitions to `uint8`types
- ASCLINIF-866 Fixed known issue: If the master request frame (MRF) for a functional transmission fails, `PduR_LinTpTxConfirmation()`is called with a wrong `TxPduId`value
- ASCLINIF-868 Fixed known issue: LinTp might call `BswM_LinTp_RequestMode()`with `LINTP_APPLICATIVE_SCHEDULE`even if LinTp communication is no longer active
- ASCLINIF-869 Fixed known issue: Wrong memory might be accessed when evaluating configuration parameter value `LinTpScheduleChangeDiag`in case of P2 timeout

- Updated symbolic name value naming schema according to AUTOSAR 4.0 Rev 3
- ASCLINIF-870 Fixed known issue: If `LinTp_Transmit()` is called for an uninitialized LinTp, an illegal memory is accessed even if Det is enabled
- Extended MCG to generate XML code for Binary Code Generation

Module version 5.3.1
2013-06-21

- ASCLINIF-755 Fixed known issue: Configuration parameters `LinTpNumberOfRxNSdu` and `LinTpNumberOfTxNSdu` have invalid default values
- ASCLINIF-758 Fixed known issue: LinIf passes wrong `HandleId` when calling `PduR_LinIfRxIndciation` for unconditional Rx-frames
- Added checking of configuration and platform-specific signature to prevent loading of incompatible post-build configuration
- Added checking of published information signature to prevent loading of incompatible post-build configuration
- ASCLINIF-788 Fixed known issue: It is not possible to receive messages with a payload length larger than 255 bytes
- ASCLINIF-789 Fixed known issue: `LinTp_CancelReceive()` does not work if the value of parameter `LinTpRxSduId` is larger than 255
- ASCLINIF-804 Fixed known issue: `PbcfgM` cannot differentiate LinTp and LinIf configuration
- ASCLINIF-801 Fixed known issue: LinIf post-build time configuration does not compile if used by `PBcfgM`
- ASCLINIF-797 Fixed known issue: LinTp ignores receive messages containing 7 bytes payload length
- ASCLINIF-796 Fixed known issue: LinTp passes the wrong value for the `network` parameter when calling `BswM_LinTp_RequestMode()`
- ASCLINIF-817 Fixed known issue: Memory mapping macros incorrectly define both variables and constants with the same memory section name
- ASCLINIF-808 Fixed known issue: Processing of empty schedule tables may cause transmission of unexpected frames
- ASCLINIF-821 Fixed known issue: LinTp does not call `BswM_LinTp_RequestMode()` with parameter `LINTP_DIAG_REQUEST` at the beginning of a functional transmission

Module version 5.3.0
2013-02-14

- Registered `HandleId` wizard for `ScheduleTableIdx` generation

- Updated reference paths of LinIf- `ComMChannel`reference for the introduction of `ComMConfigSet`container
- Added relocatability to post-build configuration

Module version 5.2.0
2012-10-24

- ASCLINIF-653 Fixed known issue: Post-build configuration of LinIf and LinTp references external symbols when used with post-build configuration manager
- ASCLINIF-651 Fixed known issue: The configuration name is different from the name of the `MULTIPLE-CONFIGURATION`container
- Implemented Tp-API according to AUTOSAR 4.0 Rev 3
- Implemented Handle ID policy according to AUTOSAR 4.0 Rev 3
- Changed the top-level structure of the software-component description in the ARXML files from `/AUTOSAR/LinIf`to `/AUTOSAR_LinIf`
- Updated to Lin Specification Package Revision 2.1
- ASCLINIF-702 Fixed known issue: Wrong `ComMChannelId`is used if `LinIfMapComMChannelIdDirect`is enabled, but `LinIfChannelId`does not match `ComMChannelId`

Module version 5.1.0
2012-06-20

- Introduced post-build data structures

Module version 5.0.0
2012-03-16

- Initial AUTOSAR 4.0 version
- Updated naming scheme for #defines for symbolic name values to AUTOSAR 4.0 Rev 3 naming scheme
- Updated config to AUTOSAR 4.0 Rev 3 schema
- Added support of AUTOSAR 4.0 Rev 3 Lin MCAL module

3.3.1.2. New features

- LinIf supports user specific callout functions Tx and Rx Bus-Adapter.

- LinIf supports variable node response tolerance according to SAE J2602 OCT2021 standard.

3.3.1.3. Elektrobit-specific enhancements

This chapter lists the enhancements provided by the module.

- Configurable support of AUTOSAR 4.0 Rev 3, and 4.2 Lin MCAL Module

  The configuration parameter `LinIfLinDriverAPI` allows to configure the LIN Interface module to support a specific Lin MCAL Module.

  `LinIfLinDriverAPI`:

  - Rev 2: Use Lin according to AUTOSAR Specification of LIN Driver V1.4.0 R4.0 Rev 2.
  - Rev 3: Use Lin according to AUTOSAR Specification of LIN Driver V1.5.0 R4.0 Rev 3.
  - 4.2: Use Lin according to AUTOSAR Specification of LIN Driver 4.2.1/4.2.2.
  - 4.3.1: Use Lin according to AUTOSAR Specification of LIN Driver 4.3.1.
  - 4.4: Use Lin according to AUTOSAR Specification of LIN Driver 4.4.0.
- Implementation of receive cancellation

  Contrary to the AUTOSAR 4.0 Rev 3 specification, cancellation of ongoing receptions by a call to `LinTp_CancelReceive` is implemented.
- Callout for Lin bus error-handling

  EB LinIf implements the two additional configuration parameters `LinIfLinErrorCalloutName` and `LinIfLinErrorCalloutHeaderFile` which enable LinIf to call a user-definable callout function in case of Lin bus communication errors.
- Vendor specific configuration parameters were introduced to support configurable reporting of the production errors "Bit-Error `(LINIF_E_TX_BIT_ERROR)` ", "Checksum-Error `(LINIF_E_RX_CHECKSUM_ERROR)` " and "Slave-Not-Responding-Error `(LINIF_E_RX_NO_RESPONSE_ERROR)` ".

  Description:

  Vendor specific configuration parameters `LinIfTxBitErrorReportToDem` , `LinIfTxBitErrorDemDetErrorId` , `LinIfTxBitErrorDebounceMethod` , `LinIfRxChecksumErrorReportToDem` , `LinIfRxChecksumErrDemDetErrorId` , `LinIfRxChecksumDebounceMethod` , `LinIfRxNoRespErrorReportToDem` , `LinIfRxNoRespErrDemDetErrorId` and `LinIfRxNoRespDebounceMethod` ,were introduced to support configurable reporting of the production errors above.
- Vendor specific configuration parameters: `LinIfScheduleTableEndNotificationSupported`, `LinIfScheduleTableEndNotificationCallout`, `LinIfScheduleTableEndNotificationRef` allow having custom end-of-schedule notifications.

- Added support referenceable NULL_SCHEDULE LinIfScheduleTable.
- Added support for forwarding the status from `Lin_GetStatus()` to the user callout.
- Added support for `Lin Confirmation Notification` and `LIN_RX_NO_REPONSE` handling in the user callout.
- LinIf now supports referencing BSWMD for Lin driver/transceiver from which to extract the Vendor ID and Vendor API Infix.
- Added support for solving the inconsistency between the LinIf and Lin drivers with an autosar version lower than 4.3. (check https://bugzilla.autosar.org/show_bug.cgi?id=73095). If the LinIf channel starts in SLEEP, at initialization LinIf forces the driver channel into sleep. If calling `Lin_GoToSleepInternal()` returns *E_NOT_OK*, a DET is called. `LINIF_DRIVER_CHANNEL_NOT_IN_SLEEP` was chosen for this purpose with reserved ID `0xFF`.
- Added support for requesting the same schedule table. If the same schedule table is requested (as the one that is running) the schedule table will be restarted.
- Increased upper-bound configuration limit of the LinTpP2Max and LinTpP2Timing parameters to 65535s.

3.3.1.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- Physical reception is not aborted by functional transmission

  Description:

  If an ongoing physical reception is preempted by a functional transmission request, then the physical reception is suspended during processing of the functional transmission. After the functional transmission has been finished, the physical reception is resumed.

  Rationale:

  This behavior is implemented according to LIN Diagnostic Specification 2.1, chapter 5.4.4.1.

  Requirements:

  SWS_LinIf_00615
- LinTp does not provide the API function `LinTp_Shutdown()` (reference to product description: ASCPD-96)

  Description:

  The API function `LinTp_Shutdown()` is not implemented in the LinTp module.

  Rationale:

There is no AUTOSAR internal user for the API function `LinTp_Shutdown()` and the behavior and operating constraints are not clearly specified in the AUTOSAR SWS. Using the function might be risky since expectations and actual behavior might differ, so it was decided to skip the function implementation.

Requirements:

SWS_LinIf_00355, SWS_LinIf_00356, SWS_LinIf_00433, SWS_LinIf_00357, SWS_LinIf_00482, SWS_LinIf_00484, SWS_LinIf_00683

- ASCCCB-1403: Initialization check in `LinIf_MainFunction()`

Description:

If `LinIf_MainFunction()` is called while the module is not yet initialized, `LinIf_MainFunction()` returns immediately without performing any functionality and without raising any Det error. This initialization check is always performed independent of the development error detection setting.

Rationale:

The SchM module may schedule the modules main function before the module is initialized. This would result in lots of Det errors during startup. Therefore the module's main function does not throw a Det error if the module is not yet initialized and simply returns in this case.

Requirements:

SWS_LinIf_00535

- The function `LinIf_CheckWakeup()` is reentrant only for different LIN channels

Description:

`LinIf_CheckWakeup()` cannot be interrupted by another `LinIf_CheckWakeup()` call.

Rationale:

`LinIf_CheckWakeup()` calls `Lin_CheckWakeup` which is non-reentrant, therefore `LinIf_CheckWakeup()` also needs to be non-reeantrant.

Requirements:

SWS_LinIf_00378

- Call of `LinIf_ScheduleRequest()` within 100ms after `LinIf_Wakeup()` may lead to an unexpected behavior

Description:

If `LinIf_ScheduleRequest()` is called after `LinIf_Wakeup()` within 100ms, it could be that a slave neither receives data nor transmits a response.

Rationale:

After a wake-up signal is sent to a LIN cluster in sleep mode, the slaves may take up to 100ms before they can communicate. Only if the slaves are ready, the master shall start communication again (LIN Protocol Specification, Revision 2.0, Section 5.1). The LIN Interface does not enforce this delay, so if frames are sent immediately after the wake-up, slaves might miss them.

- Race conditions might lead to a wrong schedule table being active during sleep mode.

  Description:

  Issuing a schedule request (via `LinIf_ScheduleRequest()`) while the LinIf is performing the transition into sleep mode (due to a `LinIf_GotoSleep()`) call. might cause the LinIf to end up in sleep mode with another schedule table than the NULL schedule being active due to internal race conditions. Note: If the LIN State Manager (LinSM) is used as upper layer for the LinIf (as designed by AUTOSAR) the module takes care that `LinIf_ScheduleRequest()` is not called during transition into sleep mode.

- ASCLINIF-579: Configuration parameter `LinIfFunctionId` has an extended range

  Description:

  The configuration parameter `LinIfFunctionId`has an extended range of 0-65535.

  Rationale:

  According to LIN Specification Package Revision 2.1 LIN function identifiers are 16-bit values. AUTOSAR 4.0 defines only a range of 0-255 for `LinIfFunctionId`. See http://www.autosar.org/bugzilla/show_bug.cgi?id=56273

- Only one frame reference per schedule table entry supported

  Description:

  A schedule table entry does not allow configuration of more than one frame reference.

  Requirements:

  ECUC_LinIf_00016

- `LinIfChannelId` does not equal `ComMChannelId`

  Description:

  It is possible to configure `LinIfChannelId` with configuration parameter `LinIfMapComMChannelId-Direct`. If the parameter is set to `TRUE`, `LinIfChannelId` must be equal to `ComMChannelId`, otherwise mapping is performed between `ComMChannelId` and `LinIfChannelId`.

  Rationale:

If other bus systems are used additionally to LinIf, the ComM channel ID must not necessarily be equal to the LinIf channel ID.

Requirements:

ECUC_LinIf_00002

- `LinIfClusterTimeBase` is not used

Description:

Configuration parameter `LinIfClusterTimeBase` is not used. Instead, the time base is derived from configuration parameter `LinIfTimeBase`.

Requirements:

ECUC_LinIf_00006

- The API function `LinIf_CancelTransmit()` is not supported (reference to product description ASCPD-24)

Description:

The API function `LinIf_CancelTransmit()` is not implemented.

Requirements:

SWS_LinIf_00580, SWS_LinIf_00649, SWS_LinIf_00581, SWS_LinIf_00594

- `LinTp_CancelTransmit()` always returns `E_NOT_OK`

Description:

If `LinTp_CancelTransmit()` is called and a transmission is ongoing, `BswM_LinTp_RequestMode()` with the parameter `LINTP_APPLICATIVE_SCHEDULE` is not called.

Rationale:

`LinTp_CancelTransmit()` is implemented as a dummy function and always returns `E_NOT_OK`.

Requirements:

SWS_LinIf_00645

- LinIfPublicCddHeaderFile parameter

Description:

The configuration parameter LinIfPublicCddHeaderFile besides CDDs is used for user defined end-of-schedule notifications as well.

Requirements:

LinIf.ASR40.ECUC_LinIf_00631

- Deviating post-build implementation

Description:

The `PbcfgM` offers the opportunity to initialize the LinIf and LinTp with different configurations during runtime. Therefore it is possible to call `LinIf_Init()` and `LinTp_Init()` more than once.

Requirements:

SWS_LinIf_00562, SWS_LinIf_00593, SWS_LinIf_00376

- Development error code

`LINIF_E_NC_NO_RESPONSE` is not reported

Description:

If a SRF is put in a schedule table after a node configuration frame and a slave does not answer the runtime error code, `LINIF_E_NC_NO_RESPONSE` is not reported.

Requirements:

SWS_LinIf_00405, SWS_LinIf_00376

- No support of configuration parameter

`LinIfNcOptionalRequestSupported` (reference to product description: ASCPD-61)

Description:

The configuration parameter `LinIfNcOptionalRequestSupported` is not supported. Node configuration frames cannot be disabled.

Rationale:

Configuration node frames are sent as fixed frames and they are not distinguished.

Requirements:

SWS_LinIf_00310

- No AUTOSAR Debugging support

Description:

LinIf is not instrumented for the usage with AUTOSAR Debugging.

Requirements:

SWS_LinIf_00515, SWS_LinIf_00516, SWS_LinIf_00517, SWS_LinIf_00518

- LinTp reception is not aborted if PDU with invalid data length is received

Description:

If a PDU is received with invalid data length, `PduR_LinTpRxIndication()` with the result `NTFRSLT_E_UNEXP_PDU` and `BswM_LinTp_RequestMode()` with the parameter `LINTP_APPLICATIVE_SCHEDULE` are not called to abort the reception. Instead the PDU is ignored.

Rationale:

Implementation according to LIN 2.1 Specification, otherwise SWS and LIN spec would be inconsistent. See http://www.autosar.org/bugzilla/show_bug.cgi?id=52375, AUTOSAR 4.1.1 [SWS_LinIf_00652]

Requirements:

SWS_LinIf_00614, SWS_LinIf_00654

- LinTp reception is not aborted if PDU with unexpected PCI is received

Description:

If a PDU is received with an unexpected PCI (CF is received instead of a FF or SF, or unknown PCI), `BswM_LinTp_RequestMode()` with the parameter `LINTP_APPLICATIVE_SCHEDULE` is not called to abort the reception. Instead the PDU is ignored.

Rationale:

Implementation according to LIN 2.1 Specification, otherwise SWS and LIN spec would be inconsistent. See http://www.autosar.org/bugzilla/show_bug.cgi?id=52375, AUTOSAR 4.1.1 [SWS_LinIf_00696] [SWS_LinIf_00697]

Requirements:

SWS_LinIf_00614

- Only post-build configuration is supported

Description:

The LinIf module only supports configuration variant `VARIANT-POST-BUILD`. `VARIANT-PRE-COMPILE` and `VARIANT-LINK-TIME` are not supported.

Requirements:

SWS_LinIf_00491, SWS_LinIf_00492, SWS_LinIf_00371, SWS_LinIf_00427

- Inter-module consistency checks are not supported

  Description:

  LinIf does not perform any inter-module consistency checks to avoid integration of incompatible files.

  Rationale:

  The module consistency check is not within the responsibility of the basic software, but part of the configuration management and delivery process.

  Requirements:

  SWS_LinIf_00383

- No macro for `LinIf_GetVersionInfo()`

  Description:

  `LinIf_GetVersionInfo()` is implemented as a C-function.

  Requirements:

  SWS_LinIf_00487

- Reception is aborted if `PduR_StartOfReception()` returns `BUFREQ_E_BUSY`

  Description:

  If `PduR_StartOfReception()` returns `BUFREQ_E_BUSY` and a buffer size smaller than the payload of the SF or FF, the LIN interface does not retry to copy data to PduR.

  Rationale:

  The LinTp does not support buffering of received data from the Lin driver.

  Requirements:

  SWS_LinIf_00679

- `LinIf_Wakeup` shall return E_NOT_OK if LIN Interface has not been initialized, if the referenced channel does not exist (identification is out of range), or if the Driver function calls within return E_NOT_OK.

  Description:

  If the LIN Interface has not been initialized, `LinIf_Wakeup` shall return E_NOT_OK. If the referenced channel does not exist (identification is out of range), `LinIf_Wakeup` shall return E_NOT_OK. If the return code of the function `Lin_Wakeup` is `E_NOT_OK`, `LinIf_Wakeup` shall return E_NOT_OK. If the return code of the function `Lin_WakeupInternal` is `E_NOT_OK`, `LinIf_Wakeup` shall return E_NOT_OK.

  Rationale:

According to Autosar SWS 4.2.1 [SWS_LinIf_00205] `LinIf_Wakeup` will not accept the request to wakeup due to one or more of the following reasons: - LIN Interface has not been initialized - referenced channel does not exist (identification is out of range) - Lin_Wakeup has returned E_NOT_OK - Lin_WakeupInternal has returned E_NOT_OK

Requirements:

SWS_LinIf_00205

- `LinIf_Wakeup` shall only call `Lin_Wakeup` if the channel state is LINIF_CHANNEL_SLEEP and the wake flag is not set.

Description:

The function `LinIf_Wakeup` shall call the function `Lin_Wakeup` of the LIN Driver module to transmit a wake-up request on the selected channel, if the channel is in the channel state LINIF_CHANNEL_SLEEP and the wakeup flag of the selected channel is not set.

Rationale:

According to Autosar SWS 4.2.1 [SWS_LinIf_00205] `LinIf_Wakeup` shall only call `Lin_Wakeup` on a certain channel, if both of the following conditions are true: - the channel is in the channel state LINIF_-CHANNEL_SLEEP - the wakeup flag of the selected channel is not set

Requirements:

SWS_LinIf_00296

- `LinSM_GotoSleepConfirmation` shall be called with the parameter TRUE if a go-to-sleep command was sent successfully or Lin_GoToSleepInternal was called.

Description:

When the go-to-sleep command was sent successfully or the function `Lin_GoToSleepInternal` was called, the LIN Interface shall invoke the function `<User>_GotoSleepConfirmation` with the parameter TRUE.

Rationale:

According to Autosar SWS 4.2.1 [SWS_LinIf_00205] `LinSM_GotoSleepConfirmation` shall be called with the parameter TRUE if one of the following reasons occur: - the go-to-sleep command was sent successfully - the function Lin_GoToSleepInternal was called

Requirements:

SWS_LinIf_00557

- LinTrcv.h header inclusion

Description:

The `LinTrcv.h`header is included via the `LinIf_TrcvTypes.h` header, not directly in the main source file. Also, the name depends on the configuration parameters - `LinIfSingleLinTrcvAPIInfixEnable`- `LinIfMultipleTrcvDriverSupported` If any of the above parameters is set to TRUE, the naming is according to http://www.autosar.org/bugzilla/show_bug.cgi?id=53325 .

Requirements:

SWS_LinIf_00555

- Parameter type differs from specified

Description:

The configuration parameter LinIfCddRef isn't implemented as a having the type of a foreign reference but as a choice reference with values limited to [ECUC-MODULE-CONFIGURATION-VALUES].

Requirements:

ECUC_LinIf_00637

- Parameter existence criteria

Description:

The requirement from the SWS states that LinIfCddRef is only needed when LinIfWakeupConfirmationUL, LinIfScheduleRequestConfirmationUL and/or LinIfGotoSleepConfirmationUL is set to CDD. This enumeration is extended by LinIfUserRxIndicationUL and LinIfUserTxUL.

Requirements:

ECUC_LinIf_00637

- Parameter existence criteria

Description:

The requirement from the SWS list the `LinIfRxIndicationUL`, `LinIfTxConfirmationUL` and `LinIfTxTriggerTransmitUL` parameters as having the type `EcucFunctionNameDef`.

Due to the fact that parent container is PB, the type was changed to `EcucReferenceDef`.

Requirements:

ECUC_LinIf_00055

ECUC_LinIf_00054

ECUC_LinIf_00628

- Unexpected NAD during TP reception

  Description:

  The SWS states that when an incorrect NAD is received the reception shall be stopped and this should be reported through `PduR_LinTpRxIndication()` with the result `NTFRSLT_E_UNEXP_PDU`.

  This applies only to consecutive frames.

  Excerpt from LIN Spec 2.1 :

  After reception of a Single Frame (SF) or First Frame (FF) PDU, with a NAD that is not equal to the functional NAD, during an ongoing message transmission the current reception shall be aborted. Reception of the new message shall be started on the receiver side if the NAD equals the node's own NAD or broadcast NAD.

  Requirements:

  SWS_LinIf_00613

  SWS_LinIf_00655

- Behavior for requesting the same run continuous table while it's running

  Description:

  The behavior of LinIf for managing a request of a run continuous table that is currently running was updated to reflect the solution of AUTOSAR 4.4.0.

  Excerpt from LinIf SWS AUTOSAR 4.4.0 :

  It is possible to request the same schedule table again. In this case, the table is restarted.

  Requirements:

  SWS_LinIf_00444

  SWS_LinIf_00028

  SWS_LinIf_00495

- Upper Limit for P2 parameters is changed

  Description:

  The allowed configurable upper limit for the LinTpP2Max has been changed from 2s to 65535s.

  The allowed configurable upper limit for the LinTpP2Timing has been changed from 0.5s to 65535s.

This has been done to allow timeout values that are greater than the ones specified in the ISO 17987-2:2016(E) standard.

Requirements:

ECUC_LinTp_00625

ECUC_LinTp_00622

- AR 4.0.3, AR 4.4.0 requirements replaced by their AR20-11 equivalent

  Description:

  The following requirements were replaced by their AR20-11 counterpart:

  LINIF364_Conf, LINIF329, LINIF341, LINIF076, LINIF078, LINIF676, LINIF079, LINIF674, LINIF106, LINIF248, LINIF254, LINIF422, LINIF466, LINIF329, LINIF330, LINIF672, LINIF068, LINIF073, LINIF075, LINIF376

  by ECUC_LinIf_00364, SWS_LinIf_00329.Master, SWS_LinIf_00341.Master, SWS_LinIf_00076.-SRF, SWS_LinIf_00078.SRF, SWS_LinIf_00676.SRF, SWS_LinIf_00079.SRF, SWS_LinIf_00106.-Master, SWS_LinIf_00674, LinIf.SWS_LinIf_00248, SWS_LinIf_00254.RX_BUSY, LinIf.SWS_LinIf_-00422, SWS_LinIf_00466, SWS_LinIf_00329.Master, SWS_LinIf_00330.Master, SWS_LinIf_00672.Master, SWS_LinIf_00068.Master, SWS_LinIf_00073.Master, SWS_LinIf_00075.Master, LinIf.SWS_LinIf_-00376_1
- Slave nodes do not support SAE.

  Description:

  Slave nodes do not support SAE_J2602 standard.
- Master nodes do not support 0x3E for MRF when SAE is used.

  Description:

  SAE_J2602 standard uses the id 0x3E for MRF and it is not supported.

3.3.1.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- Limitation: Link time Cdd support

  Description:

The configuration container LinIfChannel is post-build capable but the ConfigurationClass of upper layer Cdd support parameters is VARIANT-LINK-TIME.

Rationale:

The function pointers aren't generated within the post-build data structure (limitation).

It is considered to be the integrator's responsibility when modifying the LinIfChannel container to ensure that the set of references to LinSM, PduR or Cdds are identical to one present during LinkTime configuration.

Limitation: Compatibility with LinSM module

Description:

If used with a LinSM module from Elektrobit, the supported minimum LinSM version is 3.4.0.

Rationale:

APIs from earlier versions expect ComM, instead of LinIf handle IDs.

Limitation: Bus Mirroring number of channels

Description:

Maximum number of channels that are mirrored is 16

Rationale:

Implementation constraint from using uint16. In case of channel ID greater than the maximum mirrored channels, there will be an error reported to DET (error ID LINIF_E_INVALID_MIRROR_CHANNEL 0x70U).

Limitation: Drivers of different Autosar version

Description:

LinIf cannot use drivers of different Autosar version.

Rationale:

The configuration parameter LinIfLinDriverAPI specifies what version of Autosar the driver is expected to be. All other drivers of different Autosar versions are ignored.

Limitation: LinIfBusIdleTimeoutPeriod

Description:

The value that can be configured for LinIfBusIdleTimeoutPeriod as an upper limit depends on LinIfTimeBase. The division between the two of them needs to be smaller that 65535(i.e for a LinIfTimeBase of 0.005, the max value LinIfBusIdleTimeoutPeriod can have is 327s) .

Rationale:

To be able to monitor the timeout, a transformation into Number of Main Functions needed to be done. The decision is to limit it to uint16, since the Idle Timeout is supposed to be between 4s and 10s . uint16 seems like a reasonable value to allow also some flexibility. Bigger values were not considered as an Idle Timeout greater than 10s is anyhow not ISO compliant.

3.3.1.6. Open-source software

LinIf does not use open-source software.

3.3.2. LinSM module release notes

- AUTOSAR R4.0 Rev 3
- AUTOSAR SWS document version: 1.3.0
- Module version: 3.4.27.B567464
- Supplier: Elektrobit Automotive GmbH

3.3.2.1. Change log

This chapter lists the changes between different versions.

Module version 3.4.27
2022-09-16

- ASCLINSM-439 Fixed known issue: LinSM slave node does not wakeup after bus idle timeout

Module version 3.4.26
2022-07-22

- Updated requirement Id format in module documentation and source code tracing comments. Note: This does not change the Baseline, nor functionality..

Module version 3.4.25
2022-05-13

- Internal module improvement. This module version update does not affect module functionality.

Module version 3.4.24
2022-03-18

- Internal module improvement. This module version update does not affect module functionality.

Module version 3.4.23
2022-01-28

- Internal module improvement. This module version update does not affect module functionality.

Module version 3.4.22
2021-09-17

- Internal module improvement. This module version update does not affect module functionality.

Module version 3.4.21
2021-08-20

- ASCLINSM-418 Fixed known issue: LinSM is affected by shared data race for slave configurations

Module version 3.4.20
2021-07-28

- ASCLINSM-425 Fixed known issue: Rte interface has wrong name regarding ComM function

Module version 3.4.19
2021-06-25

- Added support for LIN slave nodes.

Module version 3.4.18
2021-05-28

- Internal module improvement. This module version update does not affect module functionality.

Module version 3.4.17
2021-04-09

- ASCLINSM-413 Fixed known issue: LinSM_GlobalState uninitialized in VAR_INIT_8 memory section

Module version 3.4.16
2021-03-05

- Internal module improvement. This module version update does not affect module functionality.

Module version 3.4.15
2020-10-23

- Internal module improvement. This module version update does not affect module functionality.

Module version 3.4.14
2020-09-25

- Internal module improvement. This module version update does not affect module functionality.

Module version 3.4.13
2020-08-28

- ASCLINSM-398 Fixed known issue: Linkage error occurs due to wrong symbolic names for LinSM schedule tables

Module version 3.4.12
2020-07-31

- Internal module improvement. This module version update does not affect module functionality.

Module version 3.4.11
2020-06-19

- Internal module improvement. This module version update does not affect module functionality.

Module version 3.4.10
2020-05-22

- Internal module improvement. This module version update does not affect module functionality.

Module version 3.4.9
2020-03-25

- ASCLINSM-388 LinSM does not enter to correct state if WakeUp is requested and GoToSleep is undergoing, or the other way around

Module version 3.4.8
2020-02-21

- Internal module improvement. This module version update does not affect module functionality.

Module version 3.4.7
2020-01-24

- Internal module improvement. This module version update does not affect module functionality.

Module version 3.4.6
2019-06-14

- Internal module improvement. This module version update does not affect module functionality.

Module version 3.4.5
2019-04-18

- Internal module improvement. This module version update does not affect module functionality.

Module version 3.4.4
2019-03-22

- Internal module improvement. This module version update does not affect module functionality
- ASCLINSM-377 Fixed known issue: A generation error occurs if LinSMScheduleIndex is configured with the value 0

Module version 3.4.3
2019-02-15

- ASCLINSM-374 Fixed known issue: LinSM generates a linker error if LinSMDevErrorDetect is disabled

Module version 3.4.2
2019-01-25

- Internal module improvement. This module version update does not affect module functionality

Module version 3.4.1
2018-12-21

- ASCLINSM-369 Fixed known issue: Async server calls for bus indication are only generated for single channel
- ASCLINSM-370 Fixed known issue: Out-of-bounds access may occur for the array LinSM_ChannelConfig

Module version 3.4.0
2018-10-26

- ASCLINSM-359 Fixed known issue: LinIf transceiver functionality does not translate the ComM channel to a LinIf channel
- Added multicore support.

Module version 3.3.7
2018-08-24

- Internal module improvement. This module version update does not affect module functionality

Module version 3.3.6
2018-06-22

- Internal module improvement. This module version update does not affect module functionality

Module version 3.3.5
2018-05-25

- Internal module improvement. This module version update does not affect module functionality

Module version 3.3.4
2018-04-20

- Internal module improvement. This module version update does not affect module functionality

Module version 3.3.3
2018-02-16

- Internal module improvement. This module version update does not affect module functionality

Module version 3.3.2
2017-09-22

- Internal module improvement. This module version update does not affect module functionality

Module version 3.3.1
2017-07-28

- Internal module improvement. This module version update does not affect module functionality

Module version 3.3.0
2017-06-30

- LinSMScheduleIndex is now calculated by using the HandleIdWizard

Module version 3.2.11
2017-05-05

- Internal module improvement. This module version update does not affect module functionality

Module version 3.2.10
2017-03-31

- Internal module improvement. This module version update does not affect module functionality

Module version 3.2.9
2017-03-10

- Internal module improvement. This module version update does not affect module functionality

Module version 3.2.8
2017-03-03

- Added Lin Transceiver support
- Move integration requirements to separate reqm file.

Module version 3.2.7
2017-01-05

- Internal module improvement. This module version update does not affect module functionality

Module version 3.2.6
2016-11-04

- Internal module improvement. This module version update does not affect module functionality

Module version 3.2.5
2016-05-25

- Internal module improvement. This module version update does not affect module functionality

Module version 3.2.4
2016-02-05

- Added support for Debug & Trace with custom header file configurable via parameter `BaseDbgHeaderFile`

Module version 3.2.3
2015-11-06

- Internal module improvement. This module version update does not affect module functionality

Module version 3.2.2
2015-06-19

- ASCLINSM-302 Fixed known issue: Configuration parameter `LinSMScheduleIndex` cannot be used

Module version 3.2.1
2015-02-20

- ASCLINSM-290 Fixed known issue: LinSM may fail to schedule another Lin schedule table via `LinSM_ScheduleRequest()` when `LinSMConfirmationTimeout` is set to zero
- ASCLINSM-296 Fixed known issue: `LinSM_ScheduleRequest()` does not call `LinIf_ScheduleRequest()` if the requested run once schedule table is already running

Module version 3.2.0
2014-10-03

- Changed the generation of symbolic name value macros for `LinSMScheduleIndex`. The macro now expands to the symbolic name value macro of the referenced `LinIfScheduleTableIndex`
- Updated LinSM module to store the requested communication mode and retry to reach it in case LinIf returns an error or does not confirm the request
- Removed obsolete legacy symbolic name values

Module version 3.1.2
2013-10-11

- ASCLINSM-252 Fixed known issue: Value of variable `LinSM_GlobalState` is not reported to Dbg module

Module version 3.1.1
2013-06-14

- Internal module improvement. This module version update does not affect module functionality

Module version 3.1.0
2013-02-08

- ASCLINSM-175 Fixed known issue: `LinSM_Version.h` defines incorrect values for the macros `LINSM_AR_MAJOR_VERSION`, `LINSM_AR_MINOR_VERSION`, `LINSM_AR_PATCH_VERSION`
- Updated reference paths of LinSm-`ComMChannel` reference for the introduction of `ComMConfigSet` container

Module version 3.0.2
2012-10-12

- Changed the top-level structure of the software-component description in the ARXML-files from `/AUTOSAR/LinSM` to `/AUTOSAR_LinSM`

Module version 3.0.1
2012-06-20

- Internal module improvement. This module version update does not affect module functionality

Module version 3.0.0
2012-03-16

- Initial AUTOSAR 4.0 version

3.3.2.2. New features

- No new features have been added since the last release.

3.3.2.3. Elektrobit-specific enhancements

This chapter lists the enhancements provided by the module.

- ScheduleRequest behavior on slave nodes

  Description:

  `LinSM` supports slave nodes and these nodes do not accept schedule requests. In this case, if `LinSM_ScheduleRequest` is called for such a node, it will return E_NOT_OK and not proceed with any action.

  Rationale:

  A slave node (configured via LinSMNodeType), does not support schedules or schedule requests, as described by SWS_LinSM_00241 and ECUC_LinSM_00146.

- Confirmation timer for GoToSleep on slaves

  Description:

  `LinSM` will start a timer when waiting for a confirmation of a mode change or a schedule change. While waiting for the GoToSleep confirmation, the slave nodes will not stop the timer when confirmation from lower layer arrives, but when transition is requested by the upper layer (ComM).

  Rationale:

  A slave node (configured via LinSMNodeType), will call lower layer for transitioning to NO_COM, and will notify ComM that it did so. After that, `LinSM` will wait for confirmation of the change from lower layer and

approval from upper layer (in the form of a request for NO_COM). After both of these are received, `LinSM` will stop the confirmation timer, as all layers have transitioned to NO_COM.

3.3.2.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- `#define` for symbolic name value `LinSMScheduleIndex` not according to AUTOSAR naming scheme

  Description:

  The name of `#defines` for the configuration parameter `LinSMScheduleIndex` does not correspond to the naming scheme for symbolic name values in the AUTOSAR document "Specification of ECU Configuration", item [ecuc_sws_2108].

  The trailing part of the symbol name which [ecuc_sws_2108] defines as "`shortName` of the container which holds the configuration parameter value", is replaced by the `shortName` of the grandparent container. This is followed by an underscore which is followed by the `shortName` of the parent container.

  For example, the symbol name is `LinSMConf_LinSMSchedule_LinSMChannel_0_LinSMSchedule_0` rather than `LinSMConf_LinSMSchedule_LinSMSchedule_0`.

  Rationale:

  This naming scheme is required to assure that symbolic name `#defines` according to item [ecuc_sws_2108] have a unique name (if default container names are used).

- Support of pre-compile time configuration only (reference to product description: ASCPD-77)

  Description:

  This LinSM module implements configuration variant 1: pre-compile time configuration.

  Requirements:

  LINSM0221

- `LinSM_Init()` accepts and ignores non-null pointer

  Description:

  Contrary to LINSM0218, `LinSM_Init()` does not check that the `ConfigPtr` argument is null.

  Rationale:

  Enable the EcuM module to initialize all modules in a uniform way (with a pointer to a default post-build configuration structure).

Requirements:

SWS_LinSM_00218

- Reporting to DET if `LinSM_ScheduleRequest` is called incorrectly

Description:

Contrary to SWS_LinSM_10211, `LinSM_ScheduleRequest` reports the vendor-specific error code `LINSM_E_NOT_IN_RUN_SCHEDULE` with the value `0x51` if the specified channel is not in the right sub-state.

Requirements:

SWS_LinSM_10211

- Reporting of vendor-specific DET error codes on unexpected call of callback functions

Description:

The vendor-specific DET error code `LINSM_E_UNEXPECTED_CALLOUT` with the value `0x60` is signaled to the DET if the LinIf calls a confirmation function (`LinSM_WakeupConfirmation`, `LinSM_ScheduleRequestConfirmation` or `LinSM_GotoSleepConfirmation`) if the confirmation is not expected.

The LinIf is allowed to signal the activation of the `NULL_SCHEDULE` (e.g. at LinIf initialization or if going to sleep) via calling `LinSM_ScheduleRequestConfirmation` at any time. In this case, DET is not signaled.

- Initialization check in main function

Description:

If the main function is called while the module is not yet initialized, the main function returns immediately without performing any functionality and without raising any Det error. This initialization check is always performed independent of the development error detection setting.

Rationale:

The RTE may schedule the module's main function before the module is initialized. This would result in lots of Det errors during startup. Therefore the module's main function does not throw a Det error if the module is not yet initialized and simply returns in this case.

Requirements:

SWS_LinSM_00179

- LinSM does not check the versions of other modules

Description:

The LinSM does not check the version numbers of included header files from other modules. Therefore it partially deviates from LINSM209.

Rationale:

In general, the modules are delivered within a whole EB tresos AutoCore delivery, in which the versions are consistent and therefore do not have to be checked.

Furthermore, this allows the combination of the module with other AUTOSAR-compatible but not fully compliant modules. This might e.g., permit to combine the module with (adapted) modules from different AUTOSAR releases or with non-AUTOSAR modules that simulate the necessary behavior.

Requirements:

SWS_LinSM_00209

- LinSM_Init does not set `NULL_SCHEDULE` for configured channels

Description:

Contrary to LINSM0216 the LinSM does not set the schedule type `NULL_SCHEDULE` for each configured channel.

Rationale:

This requirement is in contradiction to LINSM151. It is also not required because `LinIf_Init` sets the `NULL_SCHEDULE` for each configured channel anyway (see LINIF233).

Requirements:

LINSM0216

- Some API functions are not configurable

Description:

The callback functions `LinSM_WakeupConfirmation` and `LinSM_GotoSleepConfirmation` are not configurable and cannot be disabled.

Rationale:

In LinIf it cannot be configured if these functions are called or not. For further information see http://www.-autosar.org/bugzilla/show_bug.cgi?id=54715.

Requirements:

SWS_LinSM_00198, SWS_LinSM_00199

- ComM and BswM are only notified in case of mode change

Description:

The callback functions `ComM_BusSM_ModeIndication` and `BswM_LinSM_CurrentState` are only called in case the communication mode has actually changed. Thus it is not called if `LinSM_RequestComMode` returns `E_NOT_OK` or if the functions `LinSM_GotoSleepConfirmation` or `LinSM_WakeupConfirmation` are called with the argument *success=false*.

Rationale:

This is no functional limitation for the user and it improves the alignment to other state manager modules (FrSM, CanSM).

Requirements:

LINSM046, SWS_LinSM_00170, SWS_LinSM_00177, SWS_LinSM_00202, SWS_LinSM_00215

- Support of configuration variant pre-compile

Description:

Only the configuration variant pre-compile is supported. Variants link time and post-build are not supported.

Requirements:

LINSM003, LINSM0217

- No Debug & Trace support

Description:

LinSM is not instrumented for the usage with Debug & Trace.

Requirements:

SWS_LinSM_00184, SWS_LinSM_00185, SWS_LinSM_00186, SWS_LinSM_00187, SWS_LinSM_00188, SWS_LinSM_00189

- No checking of valid schedule table indices

Description:

`LinSM_ScheduleRequest` does not check if the schedule table indices are valid.

Rationale:

Configuration check already exists in LinIf. Also this check does not make sense since LinSM is a pre-compile module but the `LinIfScheduleIndex` is post-build changeable.

Requirements:

LINSM115

- Behavior of `LinSM_RequestComMode()` is changed to match ComM and other `<Net>Sm` modules.

  Description:

  `LinSM_RequestComMode()` silently ignores requests to ComM mode `SILENT_COM` and returns `E_OK`. `LinSM_RequestComMode()` returns `E_OK` on every call with valid parameters and tries to reach the requested mode no matter what the current state is. Also, it stores the requested mode in case `LinIf_Wakup` or `LinIf_GotoSleep` return `E_NOT_OK` and retries in the next main function, as specified in AUTOSAR 4.1 Rev 1.

  Rationale:

  Streamlines behavior for all `<Net>Sm` modules and thus makes special treatment of LinSm in ComM superfluous.

  Requirements:

  SWS_LinSM_00176, SWS_LinSM_00177, SWS_LinSM_00183, SWS_LinSM_00035,SWS_LinSM_00044,SWS_LinSM_10210

- `LinSM_GotoSleepIndication()` is exported via LinSM_Cbk.h

  Description:

  `LinSM_GotoSleepIndication()` according to AUTOSAR R20-11 should be exported via LinSM.h In the proprietary implementation, it will be exported via LinSM_Cbk.h.

  Rationale:

  Maintain backport compatibility with older versions.

  Requirements:

  SWS_LinSM_91000

- `LinSM` slave node shall not store NO_COM

  Description:

  `LinSM` shall not store a NO_COM request from ComM, but have NOTHING_STORED instead.

  Rationale:

  `LinSM` does not in any way use the NO_COM stored according to SWS_LinSM_00230, since there is no sleep transition retry on the MainFunction. For that, it is a simpler, more efficient implementation to have NOTHING_STORED instead.

Requirements:

SWS_LinSM_00230

- `LinSM` does not depend on NO_COM request to call ComM_BusSM_BusSleepMode

Description:

If LinIf_GotoSleep returns E_OK while `LinSM` is in FULL_COM, WAKEUP or NO_COM, it shall call ComM_BusSM_BusSleepMode without checking if there's a NO_COM request already received.

Rationale:

`LinSM` does not store NO_COM on slave nodes and it only uses the request as a confirmation that ComM is also transitioning to NO_COM.

Requirements:

SWS_LinSM_00233

- Confirmation timeout raises a DET error instead of runtime error

Description:

`LinSM` will raise a DET error with the code LINSM_E_CONFIRMATION_TIMEOUT if a confirmation timer has elapsed.

Rationale:

`LinSM` will raise a DET error instead of runtime error because it is still based on ASR 4.0.3, despite Autosar R20-10 features.

Requirements:

SWS_LinSM_00224

- Wakeup on slave will not be allowed during sleep

Description:

`LinSM` will not allow a FULL_COM request in between the request from LinSM to LinIf to GoToSleep and until the process is completed or a timer will expire.

Rationale:

ComStack and LinStack could end up in different states if a wakeup is triggered in this time-frame.

Requirements:

SWS_LinSM_00234

- GotoSleepIndication can be processed in FULL_COM, WAKEUP or NO_COM

  Description:

  `LinSM` will allow `LinSM_GotoSleepIndication` to call `LinIf_GotoSleep` when in states `LINSM_FULL_COM`, `LINSM_WAKEUP` or `LINSM_NO_COM`, not only `LINSM_FULL_COM`, because if no frame arrives after the wakeup request, `LinSM` will not reach `FULL_COM` and needs to still be able to go back to sleep.

  Rationale:

  ComStack and LinStack could end up in different states if a wakeup is triggered and no header arrives before bus idle timeout expires.

  Requirements:

  SWS_LinSM_00231, SWS_LinSM_00232

3.3.2.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- Limitation: Compatibility with LinIf module

  Description:

  If used with a LinIf module from Elektrobit, the supported minimum LinIf version is 5.8.0.

  Rationale:

  LinIf APIs from earlier versions expect ComM, instead of LinIf handle IDs.
- Limitation: Wakeup request during sleep for slave nodes

  Description:

  If a FULL_COM request comes after the GoToSleep was triggered in LinIf and before the GoToSleep process is finished (successfully or because of a timeout), the request will be ignored.

  Rationale:

  ComStack and LinStack could end up in different states if this happens.
- Limitation: Incomplete wakeup and transition to sleep

  Description:

  The following scenario can happen: 1. A wakeup process is incomplete because the first header after wakeup request is not arriving, so the LinIfBusIdleTimeoutPeriod expires. 2. When LinIfBusIdleTimeout-

Period expires, a go-to-sleep process starts that interrupts the wakeup process. If the driver returns E_NOT_OK, LinIf will end up stuck in OPERATIONAL state and LinSM will end up stuck in WAKEUP state. Given that go-to-sleep has failed, latest request from ComM will be FULL_COM. so LinSM should not transition to NO_COM, but considering the bus is idle for more than LinIfBusIdleTimeoutPeriod, LinSM transition to FULL_COM is not appropriate.

Rationale:

Theoretically, the driver should not reject the sleep (return E_NOT_OK) unless there's an invalid call (development error). Also, not receiving a header from the master for a long period of time is also considered a problem. The scenario above is considered a double-fault and if considered necesary by the project the issue can be avoided by configuring LinIfBusIdleTimeoutPeriod to a value between: lowerLimit = ((LinSMModeRequestRepetitionMax + 1) * LinSMConfirmationTimeout) and upperLimit = (((LinSMModeRequestRepetitionMax + 1) * LinSMConfirmationTimeout) + LinSMSilenceAfterWakeupTimeout) If LinIfBusIdleTimeoutPeriod needs to be greater than upperLimit, then make sure it is between than (n * upperLimit + lowerLimit) and ((n+1) * upperLimit) (where n is the number of times LinSMSilenceAfterWakeupTimeout expired). Basically, the Bus Idle Timeout should not expire during a Wakeup request + LinSMConfirmationTimeout.

3.3.2.6. Open-source software

LinSM does not use open-source software.

4. ACG8 LIN Stack user guide

4.1. Overview

The ACG8 LIN Stack user guide provides information about the concepts of the LIN stack in the AUTOSAR context. Before you read this user guide, read the general concepts about communication stacks in AUTOSAR that are described in the EB tresos AutoCore Generic documentation.

- Section 4.2, “Background information” describes the concept of LIN communication in the AUTOSAR context.
- Section 4.3, “LIN communication stack dependencies” describes the LIN stack module dependencies that differ from the general communication stack module dependencies as described in the EB tresos AutoCore Generic documentation.
- Section 4.4, “LinIf module user guide” provides `LinIf`-specific information.
- Section 4.5, “LinSM module user guide” provides `LinSM`-specific information.

4.2. Background information

This chapter provides general information about the LIN communication concepts in the AUTOSAR context. If you are not familiar with the general concepts of communication in AUTOSAR, read the general information provided in the EB tresos AutoCore Generic documentation first.

4.2.1. Communication in AUTOSAR LIN

In the LIN communication stack, there is a one-to-one mapping between I/N-PDUs and L-PDUs, i.e. frames. This means each I/N-PDU is packed into exactly one LIN frame and each LIN frame carries exactly one I/N-PDU.

The schedule table managed by the `LinIf` module drives the transmission and reception of L-PDUs. This schedule table contains entries for:

- the transmission of LIN frames,
- the reception of LIN frames,
- and the issuing of transmission confirmations.

Each of these actions is assigned to a dedicated temporal offset from the start of the schedule table. The `LinIf` main function `LinIf_MainFunction()` drives the execution of the schedule table.

If the schedule table contains a transmission entry:

1. The `LinIf_MainFunction()` calls the `PduR_LinIfTriggerTransmit()` function, which queries the `PduR` for the data to be transmitted.
2. As soon as the `PduR` has provided the data (i.e. when `PduR_LinIfTriggerTransmit()` returns), the `LinIf` issues a call to the `Lin` module function `Lin_SendFrame()` to transmit the data.
3. After the time for the transmission of the LIN frame on the bus has elapsed[1], the `LinIf` calls the `Lin_GetStatus()` function.

   This function returns the current transmission, reception, or operation status of the LIN driver.
4. A positive transmission status obtained from the LIN driver is forwarded as a transmission confirmation (i.e, a call to `PduR_LinIfTxConfirmation()`) to the `PduR`.

If the schedule table contains a reception entry:

1. The `LinIf`'s schedule table contains an entry for calling the `Lin_GetStatus()` function. This function returns the current reception status of the LIN driver.
2. If an L-PDU has been received successfully, this function provides a pointer to the received data. This pointer is used as parameter to the call to `PduR_LinIfRxIndication()` to forward the received data to the `PduR`.

As far as different frame types defined by the LIN specification [1] are concerned, the `LinIf` module supports the transmission and the reception of the frame types:

- *unconditional frame*,
- *event triggered frame*,
- *sporadic frame*,

and the *diagnostic frames*

- *master request frame*
- and *slave response frame*.

Further information on the different frame types is available in [1].

Since there is no dedicated Transport Protocol module in the LIN communication stack, the `LinIf` takes care of this functionality as well. The protocol defined here is similar to the ISO Transport Protocol for CAN [2].

4.3. LIN communication stack dependencies

[1] This time depends on the transmission speed and the number of bytes to be transmitted.

This section describes issues in which the functionality and/or the module dependencies of the LIN communication stack modules differ from the description provided in the EB tresos AutoCore Generic documentation.

4.3.1. Module dependencies

The functionality of the LIN Transport Layer is integrated in the `LinIf` module. Thus, no separate `LinTp` module is available.

NOTE

No `LinTp` plug-in is available for EB tresos AutoCore

In EB tresos Studio, the `LinTp` plug-in is available to clearly separate the configuration of the LIN Transport Protocol from the other configuration of the `LinIf` module. EB tresos AutoCore however just contains a `LinIf` module, which provides both Interface as well as Transport Layer functionality (as specified by AUTOSAR).

4.4. LinIf module user guide

4.4.1. Overview

This chapter provides you with `LinIf`-specific information:

- Section 4.4.2, “Background information” explains the concepts of the `LinIf` module.

4.4.2. Background information

4.4.2.1. Support for variable node response tolerance according to the SAE J2602 OCT2021 standard

`LinIf` supports variable node response tolerance according to the SAE J2602 OCT2021 standard. The standard (ISO or SAE) is selected using `LinIfLinProtocolVersion` parameter. For more details, see the parameter description.

If the selected standard is SAE_J2602, then `LinIfNodeResponseTolerance` can be configured to a value between 0 and 40% representing the response time tolerance. If the selected standard is ISO17987, then `LinIfNodeResponseTolerance` has to be configured to 40% (default for the parameter), according to the ISO requirement.

Changing the value of the `LinIfNodeResponseTolerance` (while selecting SAE_J2602) will affect the time `LinIf` assumes is necessary to send a specific frame.

4.4.2.2. Support for Bus Adapter callbacks

`LinIf` supports Bus Adapter callbacks that will determine if the transmission/reception of the frame will be done or not.

To enable this feature, `LinIfBusATxIndication` has to be configured to the function name provided by the Bus Adapter for managing transmission frames, `LinIfBusARxIndication` has to be configured to the function name provided by the Bus Adapter for managing reception frames and `LinIfBusAHeaderfile` has to be configured to the header file name that will provide the functions.

By enabling this feature, before sending the frame information to the driver and before providing the reception frame to the upper layer, `LinIf` will call the appropriate Bus Adapter function and only go through with the transmission/reception if the function returns TRUE. If it returns FALSE, the data will not be passed on from `LinIf`.

Support for Bus Adapter transmission and reception parts can be enabled independently, but if at least one is enabled, the configuration of `LinIfBusAHeaderfile` is mandatory.

Support for Bus Adapter transmission and reception is applicable for both master and slave nodes. If transmission/reception is enabled, it will affect both master and slave nodes.

`LinIfBusATxIndication` will be called:

- on master: for all frames (TX, Sporadic, MRF, config frames, RX, event-triggered and SRF frames)
- on slave: for all outgoing TX frames (the data is copied into the `LinIf_HeaderIndication()` data pointer)

and is expected to have the following prototype: `boolean LinIfBusATxIndication(uint8 linId, PduInfoType *pduInfo, uint8 frameType, uint8 currentChannel)` where:

- `linId` is the ID, i.e PID without the parity bits;
- `pduInfo` contains the PDU information (length and SDU data);
- `frameType` is the frame type as per the macro defines in the `LinIf_Types.h` file;
- `currentChannel` is the LIN channel that is currently processed.

`LinIfBusARxIndication` will be called:

- on master: for all RX frames (and if LIN_RX_OK status was reported by the driver)
- on slave: for all RX frames that are handled in the context of LinIf_RxIndication()

and is expected to have the following prototype: `boolean LinIfBusARxIndication(uint8 linId, PduInfoType *pduInfo, uint8 frameType, uint8 currentChannel)` where:

- `linId` is the ID, i.e PID without the parity bits;
- `pduInfo` contains the PDU information (length and SDU data);
- `frameType` is the frame type as per the macro defines in the `LinIf_Types.h` file;
- `currentChannel` is the LIN channel that is currently processed.

4.4.2.3. Error/success status callouts

4.4.2.3.1. Error callout

With the `LinIfLinErrorCalloutName` configuration parameter, you can configure an error callout to report the following statuses from the driver:

- LIN_TX_HEADER_ERROR
- LIN_TX_ERROR
- LIN_RX_ERROR
- LIN_RX_NO_RESPONSE

NOTE **No callout on LIN_RX_NO_RESPONSE for EVENT_TRIGGERED frames**

The callout is not called for EVENT_TRIGGERED frames on status LIN_RX_NO_-RESPONSE.

By activating the `LinIfLinErrorCalloutStatusForward` parameter, the status is included in the callout prototype. The callout always has the `ComM` channel ID as a parameter.

NOTE **LIN_NOT_OK status on HeaderIndication**

If status forwarding is enabled and `LinIf_HeaderIndication()` is called while the indication of a response reception is expected, the callout is called with the parameter LIN_-NOT_OK.

4.4.2.3.2. Success callout

With the `LinIfLinSuccessCalloutName` configuration parameter, you can configure a success callout to report the following statuses from the driver:

- LIN_RX_OK

- LIN_TX_OK

By activating the `LinIfLinSuccessCalloutStatusForward` parameter, the status is included in the callout prototype. The callout always has the `ComM` channel ID as a parameter.

4.4.2.4. Dem/Det reporting

NOTE **Vendor-specific feature**

This is a vendor-specific feature. The mentioned driver errors are not part of AUTOSAR. So this feature can only be used if the LIN Driver is able to report these errors, i.e. also supports Dem/Det reporting.

`LinIf` can report the following driver statuses for a specific frame directly to `Dem` or `Det`:

- LIN_TX_BIT_ERROR, configured with `LinIfTxBitErrorReportToDem`
- LIN_RX_NO_RESPONSE_ERROR, configured with `LinIfRxNoRespErrorReportToDem`
- LIN_RX_CHECKSUM_ERROR, configured with `LinIfRxChecksumErrorReportToDem`

For each error, you can select a reporting to the `Dem` or to the `Det`, or no reporting at all.

If `Dem` is selected, you can select a debouncing method, e.g. `LinIfTxBitErrorDebounceMethod` for the LIN_TX_BIT_ERROR error. Each frame has a container named `LinIfFrameDemEventParameterRefs` that is comprised of the references to DemEventParameter elements, which shall be invoked using the `Dem_ReportErrorStatus()` API if the corresponding error occurs. The event ID is taken from DemEventId value of the referenced DemEventParameter.

Further notes:

- Activation: This error is reported if a bit error is detected.
- Healing: This error is healed as soon as no bit error is detected.
- Trigger debounce: None. The error is reported on first occurrence.
- Rate of diagnostic checks: Checked on every `LinIf_MainFunction()` call.

If `Det` is selected as an option, a `Det` ID can be added, e.g. `LinIfTxBitErrorDemDetErrorId` for the LIN_TX_BIT_ERROR error. In this case, for all frames for which one of the above error statuses is reported, a `Det` error with the configured ID is reported. Based on the `Det` call, you are not able to distinguish which frame caused the problem.

4.4.2.5. End of schedule notification

`LinIf` is able to notify the user that a schedule table has ended, i.e. that the last entry in the schedule table was executed, through a specifically configured callout.

This feature is activated with the LinIfScheduleTableEndNotificationSupported configuration parameter. Once activated, you can configure a list of callouts in the LinIfScheduleTableEndNotificationCallout container.

After the list is completed, each schedule table that needs to notify the user about its ending must have a reference to one callout in the list. The reference is configured with the LinIfScheduleTableEndNotificationRef parameter, which is available for each LinIfScheduleTable.

4.4.2.6. CDD support

The CDD support is activated if the LinIfUpperLayerCddSupported is enabled.

4.4.2.6.1. State manager CDD support

LinIf can notify a CDD about state changes, replacing LinSM.

This can be enabled by configuring a reference to the CDD using the LinIfCddRef parameter available for each channel. After this parameter is enabled and the reference is valid, the CDD value can be selected from the confirmation function parameters drop-down list box. The confirmation functions that can call the CDD are represented by the following configuration parameters:

- LinIfGotoSleepConfirmationUL
- LinIfScheduleRequestConfirmationUL
- LinIfWakeupConfirmationUL

For each of the above parameters, LinIf calls the following functions if CDD is selected:

- <CDD_name>_GotoSleepConfirmation()
- <CDD_name>_ScheduleRequestConfirmation()
- <CDD_name>_WakeupConfirmation()

4.4.2.6.2. PDU CDD support

LinIf can notify a CDD about PDU indications, replacing PduR.

A list of functions can be created within the LinIfCddFunctionsUL container. Each entry must specify:

- a CDD (CddName parameter, drop-down list box of the available CDDs)
- a function type (CddFunctionType, can be: RxIndication, TxConfirmation, TriggerTransmit)
- the CDD function name (CddFunctionName, can be automatically calculated as CddName_CddFunctionType, or can have any name)

After the list is completed, for each `LinIfFrame/LinIfPduDirection`, **CDD** can be selected from the `LinIfUserTxUL` parameter drop-down list box. This action enables the `LinIfTxConfirmationUL` and `LinIfTxTriggerTransmitUL` parameters. Both allow a selection of the configured functions in the **LinIfCddFunctionsUL** container.

For both CDD features, you need to add a list of the used header files, belonging to the CDD functions, in the **LinIfPublicCddHeaderFile** container.

4.4.2.7. Bus mirroring

`LinIf` is able to mirror a `Lin` channel to the `Mirror` module. `LinIf` forwards all received/transmitted frames to the `Mirror` module using the `Mirror_ReportLinFrame()` API, when `LinIfPublicCfg/LinIfBusMirroringSupport` is enabled. This is done within the `LinIf_MainFunction()` call, after checking the status from the driver through `Lin_GetStatus()`.

The `LinIf_EnableBusMirroring()` API is available via the `LinIfPublicCfg/LinIfBusMirroringSupport` configuration parameters. This API must be called for a specific `Lin` channel before `LinIf` forwards the transmission/reception frame via `Mirror_ReportLinFrame()`.

Diagnostic frames are also mirrored.

Mirroring to a CDD is also available, if the `LinIfMirrorToCDDReportingEnable` parameter is activated in addition to `LinIfPublicCfg/LinIfBusMirroringSupport`. If it is enabled, a function name can be provided in the `LinIfMirrorToCDDReportingFunctionName` parameter. `LinIf` calls this function, instead of `Mirror_ReportLinFrame()`. You must add a header file in the `LinIfMirrorToCDDReportingHeader` parameter.

Mirroring to both `Mirror` and a CDD is not possible.

The bus mirroring feature supports multi-core usage. This can be activated with the `LinIfMultiCoreSupported` parameter. This parameter makes `LinIf` route the calls from outside of the LIN Interface to `SchM` calls. This is restricted to the `Mirror` functionality-related functions `LinIf_EnableBusMirroring()` and `LinIf_GetTrcvMode()`.

4.4.2.8. Support for multiple AUTOSAR LIN Driver versions

`LinIf` supports multiple LIN Drivers of different AUTOSAR versions. The used LIN Driver version can be selected with the `LinIfLinDriverAPI` parameter. For more details, see the parameter description.

If the AUTOSAR version is set to REV2, REV3, or REV42, and the `LinIfStartupState` of a channel is set to SLEEP, the `LinIf`, during initialization, calls the `Lin_GoToSleepInternal()` function on that channel to force the driver to enter SLEEP mode. The reason for this action is that all LIN Drivers of version 4.2 and below automatically start in OPERATIONAL mode, leading to a mismatch of modes between the two modules.

4.5. LinSM module user guide

4.5.1. Overview

This chapter provides you with `LinSM`-specific information:

- Section 4.5.2, “Background information” explains the concepts of the `LinSM` module.

4.5.2. Background information

4.5.2.1. Support for BSW distribution (multi-core)

This feature can be activated if `LinSMMultiCoreSupport` is enabled.

It allows the interaction of network-specific <Net>SM modules mapped to dedicated cores with a central `ComM` under the following conditions:

- The `ComM` module is mapped to a dedicated master core.
- The network-specific <Net>SM modules are mapped to the same core as the network-specific communication stack.

5. ACG8 LIN Stack module references

5.1. Overview

This chapter provides module references for the ACG8 LIN Stack product modules. These include a detailed description of all configuration parameters. Furthermore this chapter lists the application programming interface with all data types, constants and functions.

The content of the sections is sorted alphabetically according the EB tresos AutoCore Generic module names.

For further information on the functional behavior of these modules, refer to the chapter ACG8 LIN Stack user's guide.

5.1.1. Notation in EB module references

EB notation may differ from the AUTOSAR standard notation in the software specification documents (SWS). This section describes the notation of *default value* and *range* fields in the EB module references.

5.1.1.1. Default value of configuration parameters

If there is no default value specified for a parameter, the default value field is omitted to prevent ambiguity with parameters that have `--` as default values.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has no default value field, therefore it is omitted.

5.1.1.2. Range information of configuration parameters

The range of a configuration parameter contains an upper and a lower boundary. However, in special cases the range of allowed values can be computed by means of an XPath function that is evaluated at configuration time. An XPath function can either be a standard `xpath:<function>()` or a custom `cxpath:<function>()` function. The range of a configuration parameter may be computed based on other configuration parameters that are referenced from the XPath function. For more information on custom XPath functions, see section *Custom XPath Functions API* of the EB tresos Studio developer's guide.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has the custom XPath function `cxpath:getCompuMethodsVT()` in the range field which provides the allowed values.

5.2. LinIf

5.2.1. Configuration parameters

5.2.1.1. LinIf

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
LinIfGeneral	1..1	Container that holds all LIN interface general parameters.
LinIfEbGeneral	1..1	Container for EB specific common configurations.
LinIfGlobalConfig	1..1	This container contains the global configuration parameters of the LinIf. It is a MultipleConfigurationContainer, i.e. this container and its sub-containers exit once per configuration set. Please note that only one configuration is supported.
LinIfDefensiveProgramming	1..1	**Label:** Defensive Programming Options Parameters for defensive programming
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

Parameters included	
Parameter name	**Multiplicity**
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	**IMPLEMENTATION_CONFIG_VARIANT**
Label	Config Variant
Multiplicity	1..1

Type	ENUMERATION
Default value	VariantPostBuild
Range	VariantPostBuild

5.2.1.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	**ArMajorVersion**
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	4
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArMinorVersion**
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL

Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArPatchVersion**
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	5
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	8
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**

Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	38
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	62
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**Release**
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**

Origin	Elektrobit Automotive GmbH

5.2.1.1.2. LinIfGeneral

Containers included		
Container name	**Multiplicity**	**Description**
ReportToDem	1..1	**Label:** Production error handling Production error handling
LinIfCddFunctionsUL	0..n	List of ALL the used functions from the configured UL Cdds which are used instead of PduR. Their respective configuration elements are PB (as implemented) and these need to be enumerated latest at Link-Time. The function name can be calculated automatically if the Cdd and the Cdd Function Type are specified.
LinIfScheduleTableEndNotificationCallout	0..n	

Parameters included	
Parameter name	**Multiplicity**
LinIfCancelTransmitSupported	1..1
LinIfDevErrorDetect	1..1
LinIfMultipleDriversSupported	1..1
LinIfMultipleTrcvDriverSupported	1..1
LinIfNcOptionalRequestSupported	1..1
LinIfPublicCddHeaderFile	0..n
LinIfTpSupported	1..1
LinIfTrcvDriverSupported	1..1
LinIfVersionInfoApi	1..1
LinIfBusMirroringSupported	1..1
LinIfRuntimeErrorReporting	1..1
LinIfSingleLinTrcvAPIInfixEnable	1..1
LinIfCheckWakeupSupported	1..1
LinIfScheduleTableEndNotificationSupported	1..1

Parameters included	
LinIfUpperLayerCddSupported	1..1
LinIfDriverAPIInfixEnable	1..1
LinIfLinDriverAPI	1..1
LinIfLinErrorCalloutName	0..1
LinIfLinErrorCalloutStatusForward	1..1
LinIfLinErrorCalloutHeaderFile	1..1
LinIfLinSuccessCalloutName	0..1
LinIfLinSuccessCalloutStatusForward	1..1
LinIfLinSuccessCalloutHeaderFile	1..1
LinIfMapChannelIdDirect	1..1
LinIfMapComMChannelIdDirect	1..1
LinIfMaxChannels	1..1
LinIfMaxEventTriggeredFrames	1..1
LinIfMaxTxPdus	1..1
LinIfRelocatablePbcfgEnable	1..1
LinIfMirroringOnMultiCoreSupported	1..1
LinIfMirrorToCDDReportingEnable	1..1
LinIfMirrorToCDDReportingFunctionName	1..1
LinIfMirrorToCDDReportingHeader	1..1
LinIfBusATxIndication	0..1
LinIfBusARxIndication	0..1
LinIfBusAHeaderfile	0..1
LinIfResponseErrorSignalChangedCallout	0..1
LinIfSaveConfigurationCallout	0..1

Parameter Name	LinIfCancelTransmitSupported
Description	Global Pre-Compile Switch to reliably prevent the generation of the dummy LinIf_CancelTransmit API. **This parameter is currently not used.**
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfDevErrorDetect**	
Description	Switches the Development Error Detection and Notification ON or OFF. **Optimization Effect:** ► **ROM increase (code):** Enabling this parameter increases the ROM consumption of the module code. ► **Execution time increase (code):** Enabling this parameter increases the execution time of the module code.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfMultipleDriversSupported**	
Description	States if multiple drivers are included in the LIN Interface or not. The reason for this parameter is to reduce the size of LIN Interface if multiple drivers are not used.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfMultipleTrcvDriverSupported**	
Description	States if multiple LIN Transceivers are supported by the LIN Interface or not. The reason for this parameter is to reduce the size of LIN Interface if multiple LIN Transceivers are not used.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild

Origin	AUTOSAR_ECUC

Parameter Name	**LinIfNcOptionalRequestSupported**	
Description	This parameter is ignored as disabling the node configuration commands Assign NAD and Conditional Change NAD does not have an effect for this LinIf implementation. **This parameter is currently not used.**	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfPublicCddHeaderFile**	
Description	Defines header files for callback functions which shall be included in case of CDDs. Range of characters is 1.. 32.	
Multiplicity	0..n	
Type	STRING	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfTpSupported**
Description	States if the TP is included in the LIN Interface or not. The reason for this parameter is to reduce the size of LIN Interface if the TP is not used. **Optimization Effect:** ► **ROM increase (config):** Enabling this parameter increases the ROM consumption of the module configuration. ► **RAM increase (config):** Enabling this parameter increases the RAM consumption of the module configuration. ► **ROM increase (code):** Enabling this parameter increases the ROM consumption of the module code. ► **Execution time increase (code):** Enabling this parameter increases the execution time of the module code.
Multiplicity	1..1

Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfTrcvDriverSupported**	
Description	States if transceiver drivers are included in the LIN Interface or not. The reason for this parameter is to reduce the size of LIN Interface if transceiver drivers are not used.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfVersionInfoApi**	
Description	Switch to enable/disable the API function `LinIf_GetVersionInfo()` to read out the module's version information. ► `true`: Version info API enabled. ► `false`: Version info API disabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfBusMirroringSupported**	
Description	States if Bus Mirroring is enabled in the LIN Interface or not. The reason for this parameter is to reduce the size of LIN Interface if the Bus Mirroring is not used. ► `true`: Bus Mirroring enabled. ► `false`: Bus Mirroring disabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfRuntimeErrorReporting**
Description	Switches the **Runtime Error Reporting to Det** ON or OFF. ► **TRUE**: `LINIF_E_RESPONSE` is reported to Det ► **FALSE**: `LINIF_E_RESPONSE` is **not** reported to Det **Optimization Effect:** ► **ROM increase (code):** Enabling this parameter increases the ROM consumption of the module code. ► **Execution time increase (code):** Enabling this parameter increases the execution time of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**LinIfSingleLinTrcvAPIInfixEnable**
Description	This parameter defines if LinIf shall use the Vendor Id and the API Infix for accessing the LinTrcv module in case a single LinTrcv driver is configured. ► TRUE: LinIf uses the Vendor Id and the API Infix of the LinTrcv for accessing the LinTrcv API (e.g. LinTrcv_1_T01_SetOpMode) in case only a single LinTrcv driver is used. In addtion this name mangling is also used for including the LinTrcv header file (e.g. LinTrcv_1_T01.h) ► FALSE: LinIf does not use the Vendor Id and the API Infix of the LinTrcv in case only a single LinTrcv driver is used. Note: If **more than one LinTrcv** driver is configured, name mangling must be used. (**LinIfSingleLinTrcvAPIInfixEnable**)
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**PreCompile:** VariantPostBuild

Origin	Elektrobit Automotive GmbH

Parameter Name	**LinIfCheckWakeupSupported**	
Description	Specifies if the Lin Interface supports check wake up functionality. ► TRUE: Check wake up functionality is supported. ► FALSE: Check wake up functionality is NOT supported. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfScheduleTableEndNotificationSupported**
Description	Specifies if the Lin Interface supports end-of-schedule notification functionality. ► TRUE: Functionality is supported. ► FALSE: Functionality is NOT supported. The callout names are specified in **LinIfScheduleTableEndNotificationCallout/LinIfScheduleTableEndNotificationCalloutName** Declaration is supplied within a **LinIfPublicCddHeaderFile** entry. **Optimization Effect:** ► **ROM increase (config):** Enabling this parameter increases the ROM consumption of the module configuration. ► **ROM increase (code):** Enabling this parameter increases the ROM consumption of the module code. ► **Execution time increase (code):** Enabling this parameter increases the execution time of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfUpperLayerCddSupported**	
Description	Enables UL Cdd support. Both LinSM and PduR substitution. **Optimization Effect:** ► **ROM increase (config):** Enabling this parameter increases the ROM consumption of the module configuration. ► **ROM increase (code):** Enabling this parameter increases the ROM consumption of the module code.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfDriverAPIInfixEnable**	
Description	This parameter defines if LinIf shall use the Vendor Id and the API Infix for accessing the Lin Driver module in case a single Lin driver is configured. true: LinIf uses the Vendor Id and the API Infix of the Lin Driver for accessing the Driver API (e.g. Lin_1_T01_SendFrame) in case only a single Lin driver is used. In addtion this name mangling is also used for including the Lin Driver header file (e.g. Lin_1_T01.h) false: LinIf does not use the Vendor Id and the API Infix of the Lin Driver in case only a single Lin driver is used. Note: If more than one Lin driver is configured, name mangling must be used.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfLinDriverAPI**

Description	Specifies which AUTOSAR Revision of Lin driver API shall be used by the Lin Interface. ► **REV2**: Use Lin according to AUTOSAR Specification of LIN Driver V1.4.0 R4.0 Rev 2. ► **REV3**: Use Lin according to AUTOSAR Specification of LIN Driver V1.5.0 R4.0 Rev 3. ► **4.2**: Use Lin according to AUTOSAR Specification of LIN Driver 4.2.1/4.2.2. ► **4.3.1**: Use Lin according to AUTOSAR Specification of LIN Driver 4.3.1. (Only difference between this and 4.2, is the Lin_SendFrame function header, no other specific features/changes for 4.3.1 are included.) ► **4.4**: Use Lin according to AUTOSAR Specification of LIN Driver 4.4.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	REV3	
Range	REV2 REV3 REV42 REV431 REV44	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfLinErrorCalloutName**	
Description	If enabled this configuration parameter defines an external user function which is called in case Lin_GetStatus() returns one of the values LIN_TX_HEADER_ERROR, LIN_TX_ERROR, LIN_RX_ERROR or LIN_RX_NO_RESPONSE for any transmission/reception (LIN_RX_NO_RESPONSE is not reported for event triggered frames). If disabled, the user callout is not called. **The signature of the callout depends on the configuration parameter LinIfLinErrorCalloutStatusForward.**	
Multiplicity	0..1	
Type	FUNCTION-NAME	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	LinIfLinErrorCalloutStatusForward	
Description	If ENABLED, this configuration parameter alters the signature of the error callout from **void functionName(NetworkHandleType ComMChannel),** to **void functionName(NetworkHandleType ComMChannel, Lin_StatusType Status),** where functionName is the name of the configured callout function (LinIfLinErrorCalloutName), ComMChannel identifies the affected Lin channel according to the ComM channel configuration. The Status parameter is forwarded as returned by Lin_GetStatus().	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	DISABLE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	LinIfLinErrorCalloutHeaderFile	
Description	This configuration parameter specifies the name of the header file which contains the callout function declaration of the function configured with LinIfLinErrorCalloutName. **Please note that if LinIfLinErrorCalloutName is enabled, no Det calls with error code LINIF_E_RESPONSE are performed in the above mentioned error cases.**	
Multiplicity	1..1	
Type	STRING	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	LinIfLinSuccessCalloutName
Description	If enabled this configuration parameter defines an external user function which is called in case Lin_GetStatus() returns LIN_RX_OK or LIN_TX_OK. If disabled, the user callout is not called. **The signature of the callout depends on the configuration parameter LinIfLinSuccessCalloutStatusForward.**
Multiplicity	0..1

Type	FUNCTION-NAME	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfLinSuccessCalloutStatusForward**
Description	If ENABLED, this configuration parameter alters the signature of the error callout from **void functionName(NetworkHandleType ComMChannel),** to **void functionName(NetworkHandleType ComMChannel, Lin_StatusType Status),** where functionName is the name of the configured callout function (LinIfLinSuccessCalloutName), ComMChannel identifies the affected Lin channel according to the ComM channel configuration. The Status parameter is forwarded as returned by Lin_GetStatus().
Multiplicity	1..1
Type	BOOLEAN
Default value	DISABLE
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**LinIfLinSuccessCalloutHeaderFile**
Description	This configuration parameter specifies the name of the header file which contains the callout function declaration of the function configured with LinIfLinSuccessCalloutName.
Multiplicity	1..1
Type	STRING
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**LinIfMapChannelIdDirect**
Description	► **Map the LinIf channels to the Lin channels directly.** ► TRUE: Map the channels directly. ► FALSE: The channels are not mapped directly.
Multiplicity	1..1

Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfMapComMChannelIdDirect**
Description	► **Map the LinIf channels to the COM channels directly.** ► TRUE: Map the channels directly. ► FALSE: The channels are not mapped directly.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**LinIfMaxChannels**
Description	► **Maximum number of LinIf channels**
Multiplicity	1..1
Type	INTEGER
Default value	1
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**LinIfMaxEventTriggeredFrames**
Description	► **Maximum number of Event triggered frames**
Multiplicity	1..1
Type	INTEGER
Default value	1
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**LinIfMaxTxPdus**
Description	► **Maximum number of TxPdus**
Multiplicity	1..1

Type	INTEGER	
Default value	1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfRelocatablePbcfgEnable**	
Description	Enables/disable support for relocatable postbuild configuration. ► True: Postbuild configuration relocatable in memory. ► False: Postbuild configuration not relocatable in memory.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfMirroringOnMultiCoreSupported**	
Description	States if Mirroring on MultiCore is enabled for the LIN Interface or not. The reason for this parameter is to route the call from outside of LIN Interface to direct calls of the APIs, in case of no MultiCore or to SchM calls, in case of MultiCore. It is only used in case of Bus Mirroring support enabled. (LinIfBusMirroringSupported is set to true) ► True:Mirroring on MultiCore is enabled for LIN. ► False: Mirroring MultiCore is not enabled for LIN.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfMirrorToCDDReportingEnable**	
Description	States if frames are mirrored to the Mirror module or to a specific CDD. ► `true`: Reporting to CDD ► `false`: Reporting to Mirror	
Multiplicity	1..1	

Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPostBuild

Parameter Name	**LinIfMirrorToCDDReportingFunctionName**	
Description	Function name for CDD reporting. Example: Cdd_ReportLinFrame	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfMirrorToCDDReportingHeader**	
Description	Header containing the Cdd function for reporting. Example: Cdd.h	
Multiplicity	1..1	
Type	STRING	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfBusATxIndication**	
Label	LinIfBusATxIndication	
Description	Defines the name of the Tx Bus-Adapter specific callout function.	
Multiplicity	0..1	
Type	FUNCTION-NAME	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfBusARxIndication**	
Label	LinIfBusARxIndication	
Description	Defines the name of the Rx Bus-Adapter specific callout function.	
Multiplicity	0..1	
Type	FUNCTION-NAME	
Configuration class	**PreCompile:**	VariantPostBuild

Origin	Elektrobit Automotive GmbH

Parameter Name	**LinIfBusAHeaderfile**	
Label	LinIfBusAHeaderfile	
Description	Defines the name of Bus-Adapter header file that will be included in the source code. Example: Cdd.h	
Multiplicity	0..1	
Type	STRING	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfResponseErrorSignalChangedCallout**	
Description	This parameter contains the name of the callout function that is called after a response error signal change.	
Multiplicity	0..1	
Type	FUNCTION-NAME	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfSaveConfigurationCallout**	
Description	This parameter contains the name of the callout function that is called when a save configuration node configuration command is processed by this slave node. **The service is only supported when this parameter is configured.**	
Multiplicity	0..1	
Type	FUNCTION-NAME	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.1.3. ReportToDem

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
LinIfTxBitErrorReportToDem	1..1
LinIfTxBitErrorDebounceMethod	1..1
LinIfTxBitErrorDemDetErrorId	1..1
LinIfRxChecksumErrorReportToDem	1..1
LinIfRxChecksumDebounceMethod	1..1
LinIfRxChecksumErrDemDetErrorId	1..1
LinIfRxNoRespErrorReportToDem	1..1
LinIfRxNoRespDebounceMethod	1..1
LinIfRxNoRespErrDemDetErrorId	1..1

Parameter Name	LinIfTxBitErrorReportToDem	
Label	LINIF_E_TX_BIT_ERROR report to	
Description	Selects the handling of the production error LINIF_E_TX_BIT_ERROR. ► DEM: The error is reported to the Diagnostic Event Manager (Dem). ► DET: The error is reported to the Default Error Tracer (Det) if enabled. ► DISABLE: The error is not reported at all. **Optimization Effect:** ► **ROM reduction (code):** Setting this parameter to a value of DISABLE reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Setting this parameter to a value of DISABLE reduces the execution time of the module code.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	DISABLE	
Range	DEM DET DISABLE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	LinIfTxBitErrorDebounceMethod
Label	LINIF_E_TX_BIT_ERROR Dem Debouncing method

Description	If a production error is reported towards Dem, LinIfTxBitErrorDebounceMethod defines the whether Event debouncing is performed in Dem (DEM) or not at all (INTERNAL). In case 'DEM' is selected, LinIf always reports status PRE-PASSED/PRE-FAILED to Dem _ReportErrorStatus(). In case 'INTERNAL' is selected, LinIf always reports status PASSED/FAILED to Dem _ReportErrorStatus().
Multiplicity	1..1
Type	ENUMERATION
Default value	INTERNAL
Range	DEM INTERNAL
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**LinIfTxBitErrorDemDetErrorId**
Label	LINIF_E_TX_BIT_ERROR Dem To Det error ID
Description	If a production error is reported towards the Det, LinIfTxBitErrorDemDetErrorId defines the error ID which is reported towards the Det.
Multiplicity	1..1
Type	INTEGER
Default value	9
Range	<=255 >=9
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**LinIfRxChecksumErrorReportToDem**
Label	LINIF_E_RX_CHECKSUM_ERROR report to
Description	Selects the handling of the production error LINIF_E_RX_CHECKSUM_ERROR. ► DEM: The error is reported to the Diagnostic Event Manager (Dem). ► DET: The error is reported to the Default Error Tracer (Det) if enabled.

	► DISABLE: The error is not reported at all. **Optimization Effect:** ► **ROM reduction (code):** Setting this parameter to a value of DISABLE reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Setting this parameter to a value of DISABLE reduces the execution time of the module code.
Multiplicity	1..1
Type	ENUMERATION
Default value	DISABLE
Range	DEM DET DISABLE
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**LinIfRxChecksumDebounceMethod**
Label	LINIF_E_RX_CHECKSUM_ERROR Dem Debouncing method
Description	If a production error is reported towards the Dem, LinIfRxChecksumDebounceMethod defines the whether Event debouncing is performed in Dem (DEM) or not at all (INTERNAL). In case 'DEM' is selected, LinIf always reports status PRE-PASSED/PRE-FAILED to Dem _ReportErrorStatus(). In case 'INTERNAL' is selected, LinIf always reports status PASSED/FAILED to Dem _ReportErrorStatus().
Multiplicity	1..1
Type	ENUMERATION
Default value	INTERNAL
Range	DEM INTERNAL
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**LinIfRxChecksumErrDemDetErrorId**

Label	LINIF_E_RX_CHECKSUM_ERROR Dem To Det error ID	
Description	If a production error is reported towards Det, LinIfRxChecksumErrDemDetErrorId defines the error ID which is reported towards the Det.	
Multiplicity	1..1	
Type	INTEGER	
Default value	9	
Range	<=255 >=9	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfRxNoRespErrorReportToDem**	
Label	LINIF_E_RX_NO_RESPONSE_ERROR report to	
Description	Selects the handling of the production error LINIF_E_RX_NO_RESPONSE_ERROR. ► DEM: The error is reported to the Diagnostic Event Manager (Dem). ► DET: The error is reported to the Default Error Tracer (Det) if enabled. ► DISABLE: The error is not reported at all. **Optimization Effect:** ► **ROM reduction (code):** Setting this parameter to a value of DISABLE reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Setting this parameter to a value of DISABLE reduces the execution time of the module code.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	DISABLE	
Range	DEM DET DISABLE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfRxNoRespDebounceMethod**

Label	LINIF_E_RX_NO_RESPONSE_ERROR Dem Debouncing method
Description	If a production error is reported towards the Dem, LinIfRxNoRespDebounceMethod defines the whether Event debouncing is performed in Dem (DEM) or not at all (INTERNAL). In case 'DEM' is selected, LinIf always reports status PRE-PASSED/PRE-FAILED to Dem _ReportErrorStatus(). In case 'INTERNAL' is selected, LinIf always reports status PASSED/FAILED to Dem _ReportErrorStatus().
Multiplicity	1..1
Type	ENUMERATION
Default value	INTERNAL
Range	DEM INTERNAL
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**LinIfRxNoRespErrDemDetErrorId**
Label	LINIF_E_RX_NO_RESPONSE_ERROR Dem To Det error ID
Description	If a production error is reported towards the Det, LinIfRxNoRespErrDemDetErrorId defines the error ID which is reported towards the Det.
Multiplicity	1..1
Type	INTEGER
Default value	9
Range	<=255 >=9
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.2.1.1.4. LinIfCddFunctionsUL

Parameters included	
Parameter name	**Multiplicity**
CddName	1..1

Parameters included	
CddFunctionType	1..1
CddFunctionName	1..1

Parameter Name	CddName	
Description	Name of the Cdd. The list of possible choices is populated with the entries of LinIfPublicCddHeaderFile.	
Multiplicity	1..1	
Type	ENUMERATION	
Range	text:order(node:foreach(as:paths(as:modconf('Cdd')[node:exists(CddComStackContribution)]/../..), 'path', 'substring($path, 2)'))	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CddFunctionType	
Description	Type of the function used in the PduR surrogate Cdd	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	RxIndication	
Range	RxIndication TriggerTransmit TxConfirmation	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CddFunctionName	
Description	Function name (from the Cdd) Can be calculated automatically if the Cdd and the Cdd Function Type are specified.	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Configuration class	**VariantPostBuild:**	VariantPostBuild

Origin	Elektrobit Automotive GmbH

5.2.1.1.5. LinIfScheduleTableEndNotificationCallout

Parameters included	
Parameter name	**Multiplicity**
LinIfScheduleTableEndNotificationCalloutName	1..1

Parameter Name	**LinIfScheduleTableEndNotificationCalloutName**	
Description	Custom callout name invoked when the last entry of the schedule table is processed. Declaration is supplied within a **LinIfPublicCddHeaderFile** entry. **Optimization Effect:** ► **ROM increase (config):** Enabling this parameter increases the ROM consumption of the module configuration. ► **ROM increase (code):** Enabling this parameter increases the ROM consumption of the module code. ► **Execution time increase (code):** Enabling this parameter increases the execution time of the module code.	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.2.1.1.6. LinIfEbGeneral

Containers included		
Container name	**Multiplicity**	**Description**
LinIfEbGeneralBswmdImplementation	0..1	Container for configuring multiple Lin Drivers/Transceivers to be used by the LinIf for determining the vendorId and vendorApiInfix of a specific driver/transceiver from the corresponding BSWMD. DISABLED = vendorId and vendorApiInfix of all Lin Drivers/Transceiver are determined via CommonPublishedInformation. ENABLED = vendorId and vendorApiInfix of configured Lin Drivers/Transceiver are determined via

Containers included		
		BSWMD and for not configured Lin Drivers/Transceiver via CommonPublishedInformation.

5.2.1.1.7. LinIfEbGeneralBswmdImplementation

Containers included		
Container name	**Multiplicity**	**Description**
LinIfEbGeneralBswmdImplementationRefs	1..n	**Label:** LinIfEbGeneralBswmdReferences Container to configure a specific Lin Driver/Transceiver that shall indicate the vendorId and vendorApiInfix from its BSWMD.

5.2.1.1.8. LinIfEbGeneralBswmdImplementationRefs

Parameters included	
Parameter name	**Multiplicity**
LinIfDrvTrcvRef	1..1
LinIfDrvTrcvBswImplementationRef	0..1

Parameter Name	LinIfDrvTrcvRef	
Description	Reference that points to the used Lin driver/transceiver.	
Multiplicity	1..1	
Type	CHOICE-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	LinIfDrvTrcvBswImplementationRef	
Description	Reference to the BswImplementation of the underlying driver/transceiver which contains the vendorId and vendorApiInfix.	
Multiplicity	0..1	
Type	FOREIGN-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.2.1.1.9. LinIfGlobalConfig

Containers included		
Container name	**Multiplicity**	**Description**
LinIfChannel	1..n	

Parameters included	
Parameter name	**Multiplicity**
LinIfTimeBase	1..1

Parameter Name	LinIfTimeBase	
Description	The time-base for this channel in s (normally 0.002, 0.005 or 0.010s)	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.005	
Range	<=0.255 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.1.10. LinIfChannel

Containers included		
Container name	**Multiplicity**	**Description**
LinIfFrame	0..n	Generic container for all types of LIN frames. The shortName of this container is used as LinIfFrameName.
LinIfNodeType	1..1	This container defines the LIN node type of this channel.
LinIfMaster	1..1	Each Master can only be connected to one physical channel. This could be compared to the Node parameter in a LDF file.
LinIfScheduleTable	0..n	Describes a schedule table. Each LinIfChannel may have several schedule tables. Each schedule table can only be connected to one channel.
LinIfSlave	0..n	The Node attributes of the Slaves are provided with these parameter. **This parameter is currently not used.**

Containers included		
		The Slave can be configured under LinIfChannel/LinIfNodeType/LinIfSlave
LinIfTransceiverDrvConfig	0..1	This container contains the configuration (parameters) of all addressed LIN transceivers by each underlying LIN Transceiver Driver.

Parameters included	
Parameter name	**Multiplicity**
LinIfChannelId	1..1
LinIfCddRef	0..1
LinIfChannelRef	1..1
LinIfComMNetworkHandleRef	1..1
LinIfBusIdleTimeoutPeriod	1..1
LinIfMaxFrameCnt	1..1
LinIfGotoSleepConfirmationUL	1..1
LinIfGotoSleepIndicationUL	0..1
LinIfScheduleRequestConfirmationUL	1..1
LinIfStartupState	1..1
LinIfWakeupConfirmationUL	1..1

Parameter Name	**LinIfChannelId**	
Description	Implementation Type: NetworkHandleType	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=255 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfCddRef**
Description	EN: Reference to the CDD module description. This parameter is only required when LinIfWakeupConfirmationUL, LinIfScheduleRequestConfirmationUL, and/or LinIfGotoSleepConfirmationUL is set to CDD.

	Optimization Effect: ► **ROM increase (config):** Enabling this parameter increases the ROM consumption of the module configuration. ► **ROM increase (code):** Enabling this parameter increases the ROM consumption of the module code.	
Multiplicity	0..1	
Type	CHOICE-REFERENCE	
Range	node:paths(/AUTOSAR/TOP-LEVEL-PACKAGES/*/ELEMENTS/Cdd[@type='MODULE-CONFIGURATION' and node:exists(CddComStackContribution)])	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfChannelRef**	
Description	Reference to the used channel in Lin.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfComMNetworkHandleRef**	
Description	Unique handle to identify one certain LIN network. Reference to one of the network handles configured for the ComM.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfBusIdleTimeoutPeriod**	
Description	Bus idle timeout in seconds.	
Multiplicity	1..1	
Type	FLOAT	
Default value	4.0	
Configuration class	**VariantPostBuild:**	VariantPostBuild

Origin	AUTOSAR_ECUC

Parameter Name	**LinIfMaxFrameCnt**	
Description	Maximum number of Frames, not counting `SRF` and `MRF`. This parameter is needed only in case the node is a slave.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfGotoSleepConfirmationUL**	
Description	This parameter defines the upper layer (UL) module to which the confirmation of the goto-sleep command shall be sent. **Must be used in conjunction with LinIfCddRef.**	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	LIN_SM	
Range	CDD	
	LIN_SM	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfGotoSleepIndicationUL**	
Description	This parameter defines the upper layer (UL) module to which the indication of the goto-sleep command shall be sent. **Must be used in conjunction with LinIfCddRef.**	
Multiplicity	0..1	
Type	ENUMERATION	
Default value	LIN_SM	
Range	CDD	
	LIN_SM	
Configuration class	**VariantPostBuild:**	VariantPostBuild

Origin	AUTOSAR_ECUC

Parameter Name	**LinIfScheduleRequestConfirmationUL**	
Description	This parameter defines the upper layer (UL) module to which the confirmation of the successfully performed schedule table change. **Must be used in conjunction with LinIfCddRef.**	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	LIN_SM	
Range	CDD LIN_SM	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfStartupState**	
Description	Defines the state of each LIN channel after startup.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	NORMAL	
Range	NORMAL SLEEP	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfWakeupConfirmationUL**	
Description	This parameter defines the upper layer (UL) module to which the confirmation of the wake-up shall be sent. **Must be used in conjunction with LinIfCddRef.**	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	LIN_SM	
Range	CDD LIN_SM	

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.1.11. LinIfFrame

Containers included		
Container name	**Multiplicity**	**Description**
LinIfFixedFrameSdu	1..1	In case this is a fixed frame this is the SDU (response). This container represent an eight byte array. The Byte order shall be MSB first. This container is only available for the following LinIfFrame-Types of master nodes: ► ASSIGN ► ASSIGN_FRAME_ID_RANGE ► ASSIGN_NAD ► CONDITIONAL ► FREE ► SAVE_CONFIGURATION ► UNASSIGN
LinIfPduDirection	1..1	Direction of the frame.
LinIfFrameDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the 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.
LinIfSubstitutionFrames	0..n	List of unconditional Frames that can be sent in a sporadic Frame slot.

Parameters included	
Parameter name	**Multiplicity**
LinIfChecksumType	1..1
LinIfFrameId	0..1
LinIfFrameIndex	0..1

Parameters included	
LinIfFrameType	1..1
LinIfLength	1..1
LinIfPid	1..1

Parameter Name	LinIfChecksumType	
Description	Type of checksum that the frame is using.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	CLASSIC	
Range	CLASSIC	
	ENHANCED	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	LinIfFrameId	
Description	ID of the LIN frame. The Protected ID including parity is calculated by the generation tool.	
Multiplicity	0..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	LinIfFrameIndex	
Description	PID index of the frame. This index is used in the AssignFrameIdentifierRange node configuration service to identify the frame(s) to which a new PID shall be assigned.	
Multiplicity	0..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	LinIfFrameType

Description	Type of frame that is described (e.g. sporadic frame). Note that types 7-11 are the fixed MRF types. The sporadic slot is not found among the frame types. A sporadic slot is a set of sporadic frames.
Multiplicity	1..1
Type	ENUMERATION
Default value	UNCONDITIONAL
Range	ASSIGN ASSIGN_FRAME_ID_RANGE ASSIGN_NAD CONDITIONAL EVENT_TRIGGERED FREE MRF SAVE_CONFIGURATION SPORADIC SRF UNASSIGN UNCONDITIONAL
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**LinIfLength**
Description	Length of the LIN SDU in bytes.
Multiplicity	1..1
Type	INTEGER
Default value	8
Range	<=8 >=1
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**LinIfPid**

Description	Protected ID of the LIN frame. There is no reason to calculate the Parity in run-time.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=255 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.1.12. LinIfFixedFrameSdu

Containers included		
Container name	**Multiplicity**	**Description**
LinIfFixedFrameSduByte	8..8	This container represents a byte within the 8 byte array. The Byte order shall be MSB first.

5.2.1.1.13. LinIfFixedFrameSduByte

Parameters included	
Parameter name	**Multiplicity**
LinIfFixedFrameSduBytePos	1..1
LinIfFixedFrameSduByteVal	1..1

Parameter Name	**LinIfFixedFrameSduBytePos**	
Description	Index of the Byte in the SDU (response) 8 byte array.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfFixedFrameSduByteVal**
Description	Byte value in the SDU (response) 8-byte array.

Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=255 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.1.14. LinIfPduDirection

Containers included		
Container name	**Multiplicity**	**Description**
LinIfInternalPdu	1..1	Represents a Diagnostic or Configuration frame : no Message ID (no PduId).
LinIfRxPdu	1..1	Represents a received PDU/frame.
LinIfSlaveToSlavePdu	1..1	Represents a slave-to-slave PDU/frame. Master does only send the header but doesn't receive the response. Added for completeness.
LinIfTxPdu	1..1	Represents a transmitted PDU/frame.

5.2.1.1.15. LinIfInternalPdu

5.2.1.1.16. LinIfRxPdu

Parameters included	
Parameter name	**Multiplicity**
LinIfRxIndicationUL	0..1
LinIfRxPduRef	1..1
LinIfUserRxIndicationUL	1..1

Parameter Name	**LinIfRxIndicationUL**

Description	This parameter refers to the defined name of the `User_RxIndication`. This parameter depends on the parameter LinIfUserRxIndicationUL. If LinIfUserRxIndicationUL equals CDD the name of the `User_RxIndication` is selectable. The name is defined in LinIfGeneral/LinIfCddFunctionsUL.	
Multiplicity	0..1	
Type	ENUMERATION	
Range	text:order(../../../../../../../../LinIfGeneral/LinIfCddFunction-sUL/*[CddFunctionType='RxIndication']/@name)	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfRxPduRef**	
Description	Reference to the PDU that is received in this frame.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfUserRxIndicationUL**	
Description	This parameter defines the upper layer (UL) module to which the indication of the successfully received LINRXPDUID has to be routed via `User_LinIfRxIndication`. This `User_LinIfRxIndication` has to be invoked when the indication of the configured LINRXPDUID will be received by a Rx indication event from the LIN Driver module.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	PDUR	
Range	CDD PDUR	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.1.17. LinIfSlaveToSlavePdu

5.2.1.1.18. LinIfTxPdu

Parameters included	
Parameter name	**Multiplicity**
LinIfTxConfirmationUL	0..1
LinIfTxPduId	1..1
LinIfTxPduRef	1..1
LinIfTxTriggerTransmitUL	0..1
LinIfUserTxUL	1..1

Parameter Name	**LinIfTxConfirmationUL**	
Description	This parameter refers to the defined ame of the `User_TxConfirmation`. This parameter depends on the parameter LinIfUserTxUL. If LinIfUserTxUL equals CDD, the name of the `User_TxConfirmation` is selectable. The name is defined in LinIfGeneral/LinIfCddFunctionsUL.	
Multiplicity	0..1	
Type	ENUMERATION	
Range	text:order(../../../../../../../../LinIfGeneral/LinIfCddFunctionsUL/*[CddFunctionType='TxConfirmation']/@name)	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfTxPduId**	
Description	Identifier of the frame for the upper layer. This id is only relevant for sporadic frames.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild

Origin	AUTOSAR_ECUC

Parameter Name	**LinIfTxPduRef**	
Description	Reference to the PDU that is transmitted in this frame.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfTxTriggerTransmitUL**	
Description	This parameter refers to the defined name of the `User_TriggerTransmit`. This parameter depends on the parameter LinIfUserTxUL. If LinIfUserTxUL equals CDD, the name of the `User_TriggerTransmit` is selectable. The name is defined in LinIfGeneral/LinIfCddFunctionsUL.	
Multiplicity	0..1	
Type	ENUMERATION	
Range	text:order(../../../../../../../../LinIfGeneral/LinIfCddFunctionsUL/*[CddFunctionType='TriggerTransmit']/@name)	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfUserTxUL**	
Description	This parameter defines the upper layer (UL) module to which the trigger of the transmitted LinTxPdu (via the `User_TriggerTransmit`) or the confirmation of the successfully transmitted LinTxPdu has to be routed (via the `User_TxConfirmation`).	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	PDUR	
Range	CDD PDUR	
Configuration class	**VariantPostBuild:**	VariantPostBuild

Origin	AUTOSAR_ECUC

5.2.1.1.19. LinIfFrameDemEventParameterRefs

Parameters included	
Parameter name	**Multiplicity**
LINIF_E_TX_BIT_ERROR	0..1
LINIF_E_RX_CHECKSUM_ERROR	0..1
LINIF_E_RX_NO_RESPONSE_ERROR	0..1

Parameter Name	**LINIF_E_TX_BIT_ERROR**	
Description	Reference to the DemEventParameter that shall be issued when the LIN Driver reports a bit error to LinIf. Dependency on parameter(s): ► `LinIfTxBitErrorReportToDem`: Select DEM to enable the reporting of `LINIF_E_TX_BIT_ERROR`. Further notes: ► Activation: This error is reported if a bit error is detected. ► Healing: This error is healed as soon as no bit error is detected. ► Trigger debounce: None. The error is reported on first occurrence. ► Rate of diagnostic checks: Checked on every LinIf_MainFunction() call.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LINIF_E_RX_CHECKSUM_ERROR**
Description	Reference to the DemEventParameter that shall be issued when the LIN Driver reports a checksum error to LinIf. Dependency on parameter(s): ► `LinIfRxChecksumErrorReportToDem`: Select DEM to enable the reporting of `LINIF_E_RX_CHECKSUM_ERROR`. Further notes:

	▶ Activation: This error is reported if a checksum error is detected. ▶ Healing: This error is healed as soon as no checksum error is detected. ▶ Trigger debounce: None. The error is reported on first occurrence. ▶ Rate of diagnostic checks: Checked on every LinIf_MainFunction() call.
Multiplicity	0..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**PostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**LINIF_E_RX_NO_RESPONSE_ERROR**
Description	Reference to the DemEventParameter that shall be issued when the LIN Driver reports a slave not responding error to LinIf. Dependency on parameter(s): ▶ `LinIfRxNoRespErrorReportToDem`: Select DEM to enable the reporting of `LINIF_E_RX_NO_RESPONSE_ERROR`. Further notes: ▶ Activation: This error is reported if a slave not responding error is detected. ▶ Healing: This error is healed as soon as no slave not responding error is detected. ▶ Trigger debounce: None. The error is reported on first occurrence. ▶ Rate of diagnostic checks: Checked on every LinIf_MainFunction() call.
Multiplicity	0..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**PostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.2.1.1.20. LinIfSubstitutionFrames

Parameters included	
Parameter name	**Multiplicity**
LinIfFramePriority	1..1
LinIfSubstitutionFrameRef	1..1

Parameter Name	LinIfFramePriority	
Description	Priority of an unconditional frame if used as a sporadic frame or in case of collision resolving of event triggered frames (0 is the highest priority, 255 the lowest).	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	LinIfSubstitutionFrameRef	
Description	Reference to an unconditional Frame that is used as sporadic frame in a master node or event-triggered frame in a slave node.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.1.21. LinIfNodeType

Containers included		
Container name	**Multiplicity**	**Description**
LinIfMaster	1..1	Each Master can only be connected to one physical channel. This could be compared to the Node parameter in a LDF file.
LinIfSlave	1..1	Describes all parameters which are only relevant for a LIN Slave node.

5.2.1.1.22. LinIfMaster

Parameters included	
Parameter name	**Multiplicity**
LinIfLinProtocolVersion	1..1
LinIfNodeResponseTolerance	1..1
LinIfJitter	1..1

Parameter Name	LinIfLinProtocolVersion

Description	Defines the LIN protocol version of the master node.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	ISO17987	
Range	ISO17987 SAE_J2602	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfNodeResponseTolerance**	
Label	LinIfNodeResponseTolerance	
Description	Channel Response time Tolerance.	
Multiplicity	1..1	
Type	FLOAT	
Default value	40	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfJitter**	
Description	The jitter specifies the differences between the maximum and minimum delay from time base tick to the header sending start point in seconds.	
Multiplicity	1..1	
Type	FLOAT	
Range	<=0.255 >=0.0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.1.23. LinIfSlave

Containers included		
Container name	**Multiplicity**	**Description**

Containers included		
LinIfNodeConfigurationIdentification	1..1	This container is mandatory for all LIN 2.x and ISO17987 LIN slave nodes, and ignored for LIN 1.3 slave nodes and all master nodes.

Parameters included	
Parameter name	**Multiplicity**
LinIfLinProtocolVersion	1..1
LinIfResponseErrorSignal	1..1

Parameter Name	LinIfLinProtocolVersion	
Description	Defines the LIN protocol version of the slave node.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	ISO17987	
Range	ISO17987 LIN13 LIN20 LIN21 LIN22	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	LinIfResponseErrorSignal	
Description	Reference to the response_error signal.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.1.24. LinIfNodeConfigurationIdentification

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
LinIfSerialNumber	0..1
LinIfConfiguredNAD	1..1
LinIfFunctionId	1..1
LinIfInitialNAD	1..1
LinIfNasTimeout	1..1
LinIfSupplierId	1..1
LinIfVariantId	1..1

Parameter Name	LinIfSerialNumber	
Description	LIN serial number.	
Multiplicity	0..1	
Type	INTEGER	
Range	<=0xFFFFFFFF >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	LinIfConfiguredNAD	
Description	Slave node configured NAD.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=125 >=1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	LinIfFunctionId
Description	LIN function Id.
Multiplicity	1..1
Type	INTEGER
Range	<=65535 >=0

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfInitialNAD**	
Description	Slave node initial NAD.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=125 >=1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfNasTimeout**	
Description	N_As timeout in seconds.	
Multiplicity	1..1	
Type	FLOAT	
Range	<=1.0 >=0.0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfSupplierId**	
Description	LIN consortium or ISO LIN supplier Id.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=32767 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfVariantId**
Description	LIN variant Id.
Multiplicity	1..1

Type	INTEGER	
Range	<=255 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.1.25. LinIfMaster

Parameters included	
Parameter name	**Multiplicity**
LinIfClusterTimeBase	1..1
LinIfJitter	1..1

Parameter Name	**LinIfClusterTimeBase**	
Description	Defines a time-base for one LIN cluster in seconds (normally 0.002, 0.005 or 0.-010s). **This parameter is currently not used.**	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.010	
Range	<=0.255 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfJitter**
Description	The jitter specifies the differences between the maximum and minimum delay from time base tick to the header sending start point in seconds. Config item kept for backwards compatibility. Please use LinIfNodeType/LinIf-Master/LinIfJitter
Multiplicity	1..1
Type	FLOAT
Default value	0

Range	<=0.255 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.1.26. LinIfScheduleTable

Containers included		
Container name	**Multiplicity**	**Description**
LinIfEntry	0..n	Describes an entry in the schedule table (also known as Frame Slot).

Parameters included	
Parameter name	**Multiplicity**
LinIfResumePosition	1..1
LinIfRunMode	1..1
LinIfScheduleMode	1..1
LinIfScheduleTableIndex	1..1
LinIfScheduleTableName	0..1
LinIfScheduleTableEndNotificationRef	0..1

Parameter Name	**LinIfResumePosition**	
Description	Defines, where a schedule table shall be proceeded in case if it has been interrupted by a run-once table or MRF/SRF.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	START_FROM_BEGINNING	
Range	CONTINUE_AT_IT_POINT START_FROM_BEGINNING	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfRunMode**
Description	The schedule table can be executed in two different modes.

Multiplicity	1..1
Type	ENUMERATION
Default value	RUN_CONTINUOUS
Range	RUN_CONTINUOUS RUN_ONCE
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**LinIfScheduleMode**
Description	The schedule table can be executed in the following three different modes: ► **LINTP_APPLICATIVE_SCHEDULE**: Applicative schedule is selected ► **LINTP_DIAG_REQUEST**: Master request schedule table is selected ► **LINTP_DIAG_RESPONSE**: Slave response schedule table is selected **This parameter is currently not used.**
Multiplicity	1..1
Type	ENUMERATION
Default value	LINTP_APPLICATIVE_SCHEDULE
Range	LINTP_APPLICATIVE_SCHEDULE LINTP_DIAG_REQUEST LINTP_DIAG_RESPONSE
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**LinIfScheduleTableIndex**
Description	This is the unique index used by upper layers to identify a schedule. Note that the NULL_SCHEDULE for each channel has index 0. Please also note the following rules for setting the schedule table index: ► The indices for the schedule tables of each channel must start with 1 and be consecutive. ► Each index must be unique within a channel. ► The indices of each table must be ordered according to the priority of the schedule tables (parameter LinIfSchedulePriority).

	► The indices of RUN_ONCE tables must be lower than those of RUN_CONTINUOUS tables (parameter LinIfRunMode).
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**LinIfScheduleTableName**
Description	Optional schedule name used to cross-reference with a LDF. LIN_IF_SCHEDULE_INDEX shall be part of the schedule name. **This parameter is currently not used.**
Multiplicity	0..1
Type	STRING
Configuration class	**PostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**LinIfScheduleTableEndNotificationRef**
Description	Reference to a custom callout name invoked when the last entry of the schedule table is processed. The callout name is specified in **LinIfScheduleTableEndNotificationCallout/LinIfScheduleTableEndNotificationCalloutName** Declaration is supplied within a **LinIfPublicCddHeaderFile** entry. **Optimization Effect:** ► **ROM increase (config):** Enabling this parameter increases the ROM consumption of the module configuration. ► **ROM increase (code):** Enabling this parameter increases the ROM consumption of the module code. ► **Execution time increase (code):** Enabling this parameter increases the execution time of the module code.
Multiplicity	0..1
Type	REFERENCE
Configuration class	**PostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.2.1.1.27. LinIfEntry

Parameters included	
Parameter name	**Multiplicity**
LinIfCollisionResolvingRef	0..1
LinIfDelay	1..1
LinIfEntryIndex	1..1
LinIfFrameRef	1..1

Parameter Name	**LinIfCollisionResolvingRef**	
Description	Reference to the schedule table, which resolves the collision.	
Multiplicity	0..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfDelay**	
Description	Delay to next entry in schedule table in seconds.	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.02	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfEntryIndex**	
Description	Position of the Frame Entry in the Schedule Table. The first entry index in the schedule table is 0.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfFrameRef**
Description	Reference to the frames that belong to this schedule table entry.

Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.1.28. LinIfSlave

Parameters included	
Parameter name	**Multiplicity**
LinIfConfiguredNad	1..1
LinIfFunctionId	1..1
LinIfProtocolVersion	1..1
LinIfSupplierId	1..1
LinIfVariant	1..1

Parameter Name	**LinIfConfiguredNad**	
Description	Definition of the initial node address. **This parameter is currently not used.**	
Multiplicity	1..1	
Type	INTEGER	
Range	<=255 >=1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfFunctionId**	
Description	LIN function ID. **This parameter is currently not used.**	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=65535	

	>=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfProtocolVersion**	
Description	Defines the LIN Protocol version which is used by the slave. **This parameter is currently not used.**	
Multiplicity	1..1	
Type	STRING	
Default value	2.1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfSupplierId**	
Description	LIN Supplier ID. **This parameter is currently not used.**	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=32767 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfVariant**	
Description	Specifies the Variant ID. **This parameter is currently not used.**	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=255 >=0	

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.1.29. LinIfTransceiverDrvConfig

Parameters included	
Parameter name	**Multiplicity**
LinIfTrcvIdRef	1..1
LinIfIncludeTrcvCbk	1..1

Parameter Name	**LinIfTrcvIdRef**	
Description	Logical handle of the underlying LIN transceiver to be served by the LIN Interface.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinIfIncludeTrcvCbk**
Description	States if LinIf includes LinTrcv callback hedaer or not. ► `true`: LinIf includes the LinTrcv callback header. ► `false`: LinIf does not include the LinTrcv callback header
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.2.1.1.30. LinIfDefensiveProgramming

Parameters included	
Parameter name	**Multiplicity**
LinIfDefProgEnabled	1..1

Parameters included	
LinIfPrecondAssertEnabled	1..1
LinIfPostcondAssertEnabled	1..1
LinIfStaticAssertEnabled	1..1
LinIfUnreachAssertEnabled	1..1
LinIfInvariantAssertEnabled	1..1

Parameter Name	LinIfDefProgEnabled	
Label	Enable Defensive Programming	
Description	Enables or disables the defensive programming feature for the module LinIf. Note: This feature is dependent on the use of the development error detection module. To use the defensive programming feature, proceed as follows: 1. Enable development error detection 2. Enable defensive programming 3. Enable assertions as required	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	LinIfPrecondAssertEnabled
Label	Enable Precondition Assertions
Description	Enables handling of precondition assertion checks reported from the module LinIf. Dependency on parameter(s): ► Enable Development Error Detection (`LinIfDevErrorDetect`): must be enabled ► Enable Defensive Programming (`LinIfDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfPostcondAssertEnabled**	
Label	Enable Postcondition Assertions	
Description	Enables handling of postcondition assertion checks reported from the module LinIf. Dependency on parameter(s): ► Enable Development Error Detection (`LinIfDevErrorDetect`): must be enabled ► Enable Defensive Programming (`LinIfDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfStaticAssertEnabled**	
Label	Enable Static Assertions	
Description	Enables handling of static assertion checks reported from the module LinIf. Dependency on parameter(s): ► Enable Development Error Detection (`LinIfDevErrorDetect`): must be enabled ► Enable Defensive Programming (`LinIfDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinIfUnreachAssertEnabled**
Label	Enable Unreachable Code Assertions

Description	Enables handling of unreachable code assertion checks reported from the module LinIf. Dependency on parameter(s): ► Enable Development Error Detection (`LinIfDevErrorDetect`): must be enabled ► Enable Defensive Programming (`LinIfDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**LinIfInvariantAssertEnabled**
Label	Enable Invariant Assertions
Description	Enables handling of invariant assertion checks reported from functions of the module LinIf. Dependency on parameter(s): ► Enable Development Error Detection (`LinIfDevErrorDetect`): must be enabled ► Enable Defensive Programming (`LinIfDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.2.1.1.31. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	PbcfgMSupport
Label	PbcfgM support
Description	Specifies whether or not the LinIf can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.2.1.2. LinTp

Containers included		
Container name	**Multiplicity**	**Description**
LinTpGeneral	1..1	Container that holds all LIN transport protocol general parameters.
LinTpGlobalConfig	1..1	This container contains the global configuration parameter of the LinTp. It is a MultipleConfigurationContainer, i.e. this container and its sub-containers exit once per configuration set.
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

Parameters included	
Parameter name	**Multiplicity**
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	IMPLEMENTATION_CONFIG_VARIANT
Label	Config Variant
Multiplicity	1..1
Type	ENUMERATION

Default value	VariantPostBuild
Range	VariantPostBuild

5.2.1.2.1. LinTpGeneral

Parameters included	
Parameter name	**Multiplicity**
LinTpVersionInfoApi	1..1
LinTpRelocatablePbcfgEnable	1..1
LinTpScheduleChangeDiagApiEnable	1..1

Parameter Name	**LinTpVersionInfoApi**	
Description	Switch to enable/disable the API function `LinTp_GetVersionInfo()` to read out the module's version information. ► `true`: Version info API enabled. ► `false`: Version info API disabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinTpRelocatablePbcfgEnable**	
Description	Enables/disable support for relocatable postbuild configuration. ► True: Postbuild configuration relocatable in memory. ► False: Postbuild configuration not relocatable in memory.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinTpScheduleChangeDiagApiEnable**

Description	Switches `BswM_LinTp_RequestMode` API on and off. If turned on diagnostic schedules are requested from the BSwM automatically. This configuration parameter can only be turned off if LinTpScheduleChangeDiag is disabled in every LinTpChannelConfig. ► `true`: Enables change diagnostic schedule mode API. ► `false`: Disables change diagnostic schedule mode API. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. Only used if LIN Master nodes are configured.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.2.1.2.2. LinTpGlobalConfig

Containers included		
Container name	**Multiplicity**	**Description**
LinTpChannelConfig	0..n	This container contains the channel specific configuration parameter of LinTp.
LinTpRxNSdu	0..n	Container for each received N-SDU on any channel the node is connected to.
LinTpTxNSdu	0..n	Container for each transmitted N-SDU on any channel the node is connected to.

Parameters included	
Parameter name	**Multiplicity**
LinTpMaxNumberOfRespPendingFrames	1..1
LinTpNumberOfRxNSdu	1..1
LinTpNumberOfTxNSdu	1..1
LinTpP2Max	1..1

Parameters included	
LinTpP2Timing	1..1

Parameter Name	LinTpMaxNumberOfRespPendingFrames	
Description	Configures the maximum number of allowed response pending frames. Only used for LIN Master nodes, ignored for slave nodes.	
Multiplicity	1..1	
Type	INTEGER	
Default value	8	
Range	<=65534 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	LinTpNumberOfRxNSdu	
Description	**This configuration parameter is not used.** Number of transport protocol messages that can be received for all channels this node is connected to.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=65535 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	LinTpNumberOfTxNSdu	
Description	**This configuration parameter is not used.** Number of transport protocol messages that can be transmitted for all channels this node is connected to.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=65535 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild

Origin	AUTOSAR_ECUC

Parameter Name	**LinTpP2Max**	
Description	P2 Timeout when a response pending frame is expected in seconds. Only used for LIN Master nodes, ignored for slave nodes. **Note: A value of 0.0 disables this timeout.**	
Multiplicity	1..1	
Type	FLOAT	
Default value	2	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinTpP2Timing**	
Description	Definition of the P2 timeout observation parameter in seconds. **Note: A value of 0.0 disables this timeout.**	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.5	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.2.3. LinTpChannelConfig

Parameters included	
Parameter name	**Multiplicity**
LinTpDropNotRequestedNad	1..1
LinTpScheduleChangeDiag	1..1

Parameter Name	**LinTpDropNotRequestedNad**
Description	Configures if TP Frames of not requested LIN-Slaves are dropped or not. ► `false`: Do drop TP Frames of Not requested LIN-Slaves ► `true`: Drop not TP Frames of Not requested LIN-Slaves Only used for LIN Master nodes, ignored for slave nodes.

Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinTpScheduleChangeDiag**	
Description	Enables or disables the call of BswM_LinTp_RequestMode() to diagnostic request/response schedule. ► `false`: BswM is not called ► `true`: BswM is called Only used for LIN Master nodes, ignored for slave nodes.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.2.4. LinTpRxNSdu

Parameters included	
Parameter name	**Multiplicity**
LinTpDl	1..1
LinTpNcr	0..1
LinTpRxNSduId	1..1
LinTpRxNSduNad	1..1
LinTpRxNSduPduRef	1..1
LinTpRxNSduChannelRef	1..1
LinTpRxNSduTpChannelRef	1..1

Parameter Name	**LinTpDl**
Description	Data Length Code of this RxNsdu. In case of variable length message, this value indicates the minimum data length.

	Range of minimum length is 1 to 4095. Note that this is not relevant for Tx. The reason for this is to have identical structures for Tx and Rx.
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	<=4095 >=1
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**LinTpNcr**
Description	Value in seconds of the N_Cr timeout. N_Cr is the time until reception of the next Consecutive Frame N_PDU. **Note: Disabling this config parameter or a value of 0.0 disables this timeout.**
Multiplicity	0..1
Type	FLOAT
Default value	1
Range	<=1 >=0
Configuration class	**PostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**LinTpRxNSduId**
Description	The identifier of the Transport Protocol message.
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**LinTpRxNSduNad**
Description	A N-SDU transported on LIN is identified using the NAD for the specific slave.

	Only used for LIN Master nodes, ignored for slave nodes.	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinTpRxNSduPduRef**	
Description	Reference to the global PDU.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinTpRxNSduChannelRef**	
Description	Index of the channel this N-SDU belongs to.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**LinTpRxNSduTpChannelRef**	
Description	Reference to LinTp configuration for this channel.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.2.1.2.5. LinTpTxNSdu

Parameters included	
Parameter name	**Multiplicity**
LinTpMaxBufReq	1..1

Parameters included	
LinTpNas	1..1
LinTpNcs	0..1
LinTpTxNSduId	1..1
LinTpTxNSduNad	1..1
LinTpTxNSduPduRef	1..1
LinTpTxNSduTpChannelRef	1..1
LinTpTxNSduChannelRef	1..1

Parameter Name	LinTpMaxBufReq	
Description	This parameter defines the maximum number of times the LinTp should request upper layer for the Tx Buffer. It is also used to limit the number of retries for PduR_LinTpCopyTxData when no timer is active. **This parameter is currently not used.**	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=255 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	LinTpNas	
Description	Value in second of the N_As timeout. N_As is the time for transmission of a LIN frame (any N_PDU) on the part of the sender.	
Multiplicity	1..1	
Type	FLOAT	
Default value	1	
Range	<=1 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	LinTpNcs

Description	Value in seconds of for the maximum N_CS. This timeout monitors the time waiting for Tx-data arrival within the Ecu. **Note: A value of 0.0 disables this timeout.**
Multiplicity	0..1
Type	FLOAT
Default value	0.8
Range	<=1 >=0
Configuration class	**PostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**LinTpTxNSduId**
Description	The identifier of the Transport Protocol message. This ID will be the one that is communicated with upper layers.
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**LinTpTxNSduNad**
Description	A N-SDU transported on LIN is identified using the NAD for the specific slave. Only used for LIN Master nodes, ignored for slave nodes.
Multiplicity	1..1
Type	INTEGER
Default value	1
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**LinTpTxNSduPduRef**
Description	Reference to the global PDU.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**VariantPostBuild:** VariantPostBuild

Origin	AUTOSAR_ECUC

Parameter Name	**LinTpTxNSduTpChannelRef**	
Description	Reference to LinTp configuration for this channel.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinTpTxNSduChannelRef**	
Description	Index of the channel this N-SDU belongs to.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.2.6. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	**ArMajorVersion**
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.

Multiplicity	1..1
Type	INTEGER_LABEL
Default value	4
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArMinorVersion**
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArPatchVersion**
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	5
Configuration class	**PublishedInformation:**

Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	8
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	38
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	32770
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list

Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**Release**
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.2.1.2.7. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	**PbcfgMSupport**
Label	PbcfgM support
Description	Specifies whether or not the LinTp can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.2.2. Application programming interface (API)

5.2.2.1. Macro constants

5.2.2.1.1. LINIF_NULL_SCHEDULE

Purpose	Null schedule identification.
Value	0U

5.2.2.1.2. PBCFGM_NO_CFG_REQUIRED

Purpose	
Value	

5.2.2.2. Functions

5.2.2.2.1. LinIf_CheckWakeup

Purpose	Check wakeup function.	
Synopsis	`Std_ReturnType` **`LinIf_CheckWakeup`** `( EcuM_WakeupSourceType WakeupSource );`	
Service ID	0x60	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`WakeupSource`	Source device, which initiated the wake up event: LIN controller or LIN transceiver
Return Value	Result of the operation	
	`E_OK`	No error has occurred during execution of the API
	`E_NOT_OK`	An error has occurred during execution of the API
Description	Will be called when the EcuM has been notified about a wakeup on a specific LIN channel.	

5.2.2.2.2. LinIf_EnableBusMirroring

Purpose	This function indicates the channels that are enabled for bus mirroring.	
Synopsis	`Std_ReturnType` **`LinIf_EnableBusMirroring`** `( NetworkHandleType Channel , boolean MirroringActive );`	
Parameters (in)	`Channel`	Channel which is currently processed
	`MirroringActive`	The state of the channel - if it is enabled for bus mirroring or not
Parameters (in,out)	`Channel`	Channel which is currently processed
	`MirroringActive`	The state of the channel - if it is enabled for bus mirroring or not
Return Value		

5.2.2.2.3. LinIf_GetConfiguredNAD

Purpose	Configured NAD retrieval.	
Synopsis	`Std_ReturnType` **`LinIf_GetConfiguredNAD`** `( NetworkHandleType Channel , uint8 * Nad );`	
Parameters (in)	`Channel`	LinIf Channel ID
Parameters (out)	`Nad`	Configured NAD of slave
Return Value	Std_ReturnType	
	`E_OK`	Request has been accepted
	`E_NOT_OK`	Request has not been accepted, development or production error occurred
Description	Reports the current configured NAD. Only applicable for LIN slave nodes.	

5.2.2.2.4. LinIf_GetPIDTable

Purpose	PID Table retrieval.	
Synopsis	`Std_ReturnType` **`LinIf_GetPIDTable`** `( NetworkHandleType Channel , Lin_FramePidType * PidBuffer , uint8 * PidBufferLength );`	
Parameters (in)	`Channel`	LinIf Channel ID
Parameters (in,out)	`PidBuffer`	Pointer to existing buffer to which the current assigned PID values are copied to

	PidBufferLength	Pointer to actual length of provided buffer. After successful return, it contains the number of copied PID values.
Return Value	Std_ReturnType	
	E_OK	Request has been accepted
	E_NOT_OK	Request has not been accepted, development or production error occurred
Description	Retrieves all assigned PID values. The order is congruent to the LIN frame index. Only applicable for LIN slave nodes.	

5.2.2.2.5. LinIf_GetTrcvMode

Purpose		
Synopsis	`Std_ReturnType` **`LinIf_GetTrcvMode`** `( NetworkHandleType Channel , LinTrcv_TrcvModeType * TransceiverModePtr );`	
Return Value		

5.2.2.2.6. LinIf_GetTrcvWakeupReason

Purpose		
Synopsis	`Std_ReturnType` **`LinIf_GetTrcvWakeupReason`** `( NetworkHandleType Channel , LinTrcv_TrcvWakeupReasonType * TrcvWuReasonPtr );`	
Return Value		

5.2.2.2.7. LinIf_GetVersionInfo

Purpose	Return version Information.	
Synopsis	`void` **`LinIf_GetVersionInfo`** `( Std_VersionInfoType * versioninfo );`	
Service ID	0x03	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (out)	versioninfo	Version information are written to this variable.

5.2.2.2.8. LinIf_GotoSleep

Purpose	Set channel to sleep.	
Synopsis	`Std_ReturnType` **`LinIf_GotoSleep`** `( NetworkHandleType Channel );`	
Service ID	0x06	
Sync/Async	Asynchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`Channel`	The LIN channel to operate on.
Return Value	Result of the request	
	`E_OK`	Request has been accepted or sleep transition is already in progress
	`E_NOT_OK`	Request has not been accepted
Description	This function schedules a sleep request for execution. The channel will not enter sleep mode before the next schedule entry is due.	

5.2.2.2.9. LinIf_HeaderIndication

Purpose	Header Indication function.	
Synopsis	`Std_ReturnType` **`LinIf_HeaderIndication`** `( NetworkHandleType Channel , Lin_PduType * PduPtr );`	
Parameters (in)	`Channel`	LinIf Channel ID
Parameters (in,out)	`PduPtr`	Pointer to PDU providing the received PID and pointer to the SDU data buffer as in parameter. Upon return, the length, checksum type and frame response type are received as out parameter. If the frame response type is LIN_FRAMERESPONSE_TX, then the SDU data buffer contains the transmission data.
Return Value	Std_ReturnType	
	`E_OK`	Request has been accepted
	`E_NOT_OK`	Request has not been accepted, development or production error occurred
Description	This service is called by the LIN Driver to report a received LIN header. Only applicable for LIN slave nodes.	

5.2.2.2.10. LinIf_Init

Purpose	Initialize module.	
Synopsis	`void` **`LinIf_Init`** `( const LinIf_ConfigType * ConfigPtr );`	
Service ID	0x01	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`ConfigPtr`	Not used.
Description	This function initializes the LIN Interface	

5.2.2.2.11. LinIf_IsValidConfig

Purpose	Validate configuration.
Synopsis	`Std_ReturnType` **`LinIf_IsValidConfig`** `( const void * voidConfigPtr );`
Service ID	0x62
Sync/Async	Synchronous
Reentrancy	Reentrant
Return Value	E_OK if the given module configurations is valid otherwise E_NOT_OK.
Description	Checks if the post build configuration fits to the link time configuration part.

5.2.2.2.12. LinIf_LinErrorIndication

Purpose	Error Indication function.	
Synopsis	`void` **`LinIf_LinErrorIndication`** `( NetworkHandleType Channel , Lin_SlaveErrorType ErrorStatus );`	
Service ID	0x7B	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different Channels	
Parameters (in)	`Channel`	LinIf Channel ID
	`ErrorStatus`	Type of detected error
Description	This service is called by the LIN Driver to report a detected error event during header or response processing. Only applicable for LIN slave nodes	

5.2.2.2.13. LinIf_MainFunction

Purpose	LIN Interface main processing function.
Synopsis	`void` **`LinIf_MainFunction`** `( void );`
Service ID	0x80
Production Errors	▶ LINIF_E_RX_CHECKSUM_ERROR: thrown, if a checksum error is detected. ▶ LINIF_E_RX_NO_RESPONSE_ERROR: thrown, if a slave not responding error is detected. ▶ LINIF_E_TX_BIT_ERROR: thrown, if a bit error is detected.
Description	This function performs nearly everything the LIN Interface has to handle. All access to the LIN bus happens here.

5.2.2.2.14. LinIf_RxIndication

Purpose	Reception Indication function.	
Synopsis	`void` **`LinIf_RxIndication`** `( NetworkHandleType Channel , uint8 * Lin_SduPtr );`	
Service ID	0x79	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different Channels	
Parameters (in)	`Channel`	LinIf Channel ID
	`Lin_SduPtr`	Pointer to a shadow buffer or memory mapped LIN Hardware receive buffer where the current SDU is stored. This pointer is only valid if the response is received.
Description	This service is called by the LIN Driver to report a successfully received response and reception data. Only applicable for LIN slave nodes.	

5.2.2.2.15. LinIf_ScheduleRequest

Purpose	Request schedule table for execution.
Synopsis	`Std_ReturnType` **`LinIf_ScheduleRequest`** `( NetworkHandleType Channel , LinIf_SchHandleType ScheduleTable );`

Service ID	0x05	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	`Channel`	The LIN channel to operate on.
	`ScheduleTable`	The Id of the schedule requested.
Return Value	Result of the operation	
	`E_OK`	Schedule table request has been accepted
	`E_NOT_OK`	Schedule table request has been rejected
Description	This function schedules a schedule table for execution. Note that when the NULL_SCHEDULE is requested, all previous requests are deleted.	

5.2.2.2.16. LinIf_SetConfiguredNAD

Purpose	Configured NAD assignment.	
Synopsis	`Std_ReturnType` **`LinIf_SetConfiguredNAD`** `( NetworkHandleType Channel , uint8 Nad );`	
Parameters (in)	`Channel`	LinIf Channel ID
	`Nad`	Configured NAD to set as new slave NAD
Return Value	Std_ReturnType	
	`E_OK`	Request has been accepted
	`E_NOT_OK`	Request has not been accepted, development or production error occurred
Description	Sets the current configured NAD. Only applicable for LIN slave nodes.	

5.2.2.2.17. LinIf_SetPIDTable

Purpose	PID Table assignment.	
Synopsis	`Std_ReturnType` **`LinIf_SetPIDTable`** `( NetworkHandleType Channel , Lin_FramePidType * PidBuffer , uint8 PidBufferLength );`	
Parameters (in)	`Channel`	LinIf Channel ID
	`PidBuffer`	Pointer to buffer which contains the PID values to configure.

	`PidBufferLength`	Number of PID values in the provided buffer
Return Value	Std_ReturnType	
	`E_OK`	Request has been accepted
	`E_NOT_OK`	Request has not been accepted, development or production error occurred
Description	Sets all assigned PID values. The order is congruent to the LIN frame index. Only applicable for LIN slave nodes.	

5.2.2.2.18. LinIf_SetTrcvMode

Purpose		
Synopsis	`Std_ReturnType` **`LinIf_SetTrcvMode`** `( NetworkHandleType Channel , LinTrcv_TrcvModeType TransceiverMode );`	
Return Value		

5.2.2.2.19. LinIf_SetTrcvWakeupMode

Purpose		
Synopsis	`Std_ReturnType` **`LinIf_SetTrcvWakeupMode`** `( NetworkHandleType Channel , LinTrcv_TrcvWakeupModeType LinTrcvWakeupMode );`	
Return Value		

5.2.2.2.20. LinIf_Transmit

Purpose	Schedule transmission of a sporadic frame.	
Synopsis	`Std_ReturnType` **`LinIf_Transmit`** `( PduIdType LinTxPduId , const PduInfoType * PduInfoPtr );`	
Service ID	0x04	
Sync/Async	Asynchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`LinTxPduId`	The PDU Id of the sporadic frame to be sent.
	`PduInfoPtr`	Not used.

Return Value	Result of the operation	
	`E_OK`	Transmit request has been accepted
	`E_NOT_OK`	Transmit request has been rejected
Description	This function schedules sporadic frames for transmission.	

5.2.2.2.21. LinIf_TxConfirmation

Purpose	Transmission Confirmation function.	
Synopsis	`void` **`LinIf_TxConfirmation`** `( NetworkHandleType Channel );`	
Service ID	0x7A	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different Channels	
Parameters (in)	`Channel`	LinIf Channel ID
Description	This service is called by the LIN Driver to report a successfully transmitted response. Only applicable for LIN slave nodes.	

5.2.2.2.22. LinIf_Wakeup

Purpose	Wake up channel.	
Synopsis	`Std_ReturnType` **`LinIf_Wakeup`** `( NetworkHandleType Channel );`	
Service ID	0x07	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	`Channel`	The LIN channel to operate on.
Return Value	Result of the operation	
	`E_OK`	Wakeup request has been accepted
	`E_NOT_OK`	Wakeup request has been rejected
Description	This function wakes up a LIN channel.	

5.2.2.2.23. LinTp_CancelReceive

Purpose	Cancel receive.

Synopsis	`Std_ReturnType` **`LinTp_CancelReceive`** `( PduIdType LinTpRxSduId );`	
Service ID	0x47	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`LinTpRxSduId`	- This parameter contains the LinTP instance unique identifier of the Lin N-SDU reception of which has to be canceled.
Return Value	Result of the operation	
	`E_OK`	The cancellation request was accepted.
	`E_NOT_OK:`	Cancellation request of the reception of the specified Lin N-SDU is rejected
Description	This function requests the cancellation of a segmented reception of the given Rx N-SDU. The cancellation itself will be performed during the next LinIf_MainFunction() call.	

5.2.2.2.24. LinTp_CancelTransmit

Purpose	Cancel transmit.	
Synopsis	`Std_ReturnType` **`LinTp_CancelTransmit`** `( PduIdType LinTpTxSduId );`	
Service ID	0x46	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`LinTpTxSduId`	LIN N-SDU identifier
Return Value	Result of the operation	
	`E_NOT_OK:`	Cancellation request of the transfer of the specified Lin N-SDU is rejected
Description	This function is defined for the upper layer to have a cancel transmit function. It does nothing else than checking the LinTp state if development error detection is enabled and always returns E_NOT_OK. This is a dummy method introduced for interface compatibility.	

5.2.2.2.25. LinTp_ChangeParameter

Purpose	Change parameter.

Synopsis	`Std_ReturnType` **`LinTp_ChangeParameter`** `( PduIdType id , TPParameterType parameter , uint16 value );`	
Service ID	0x44	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`id`	- Identifier of the received N-SDU on which the reception parameter has to be changed.
	`parameter`	- The selected parameter that the request shall change (STmin).
	`value`	- The new value of the parameter.
Return Value	Result of the operation	
	`E_NOT_OK:`	request is not accepted
Description	This function is defined for the upper layer to have a change parameter request function.This service is used to request the change of reception parameter STmin for a specified N-SDU.	

5.2.2.2.26. LinTp_GetVersionInfo

Purpose	Return version Information.	
Synopsis	`void` **`LinTp_GetVersionInfo`** `( Std_VersionInfoType * versioninfo );`	
Service ID	0x42	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (out)	`versioninfo`	Version information are written to this variable.

5.2.2.2.27. LinTp_Init

Purpose	Initialize TP.
Synopsis	`void` **`LinTp_Init`** `( const LinTp_ConfigType * ConfigPtr );`
Service ID	0x40

Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`ConfigPtr`	Not used.
Description	This function initializes the LIN Transport Layer	

5.2.2.2.28. LinTp_IsValidConfig

Purpose	Validate configuration.
Synopsis	`Std_ReturnType` **`LinTp_IsValidConfig`** `( const void * voidConfigPtr );`
Service ID	0x48
Sync/Async	Synchronous
Reentrancy	Reentrant
Return Value	E_OK if the given module configurations is valid otherwise E_NOT_OK.
Description	Checks if the post build configuration fits to the link time configuration part.

5.2.2.2.29. LinTp_Transmit

Purpose	Start a TP transmission.	
Synopsis	`Std_ReturnType` **`LinTp_Transmit`** `( PduIdType LinTpTxSduId , const PduInfoType * LinTpTxInfoPtr );`	
Service ID	0x41	
Sync/Async	Asynchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`LinTpTxSduId`	The PDU Id of the message to be sent
	`LinTpTxInfoPtr`	A PduInfoType to pass the length of the message
Return Value	Result of the operation	
	`E_OK`	Transmit request has been accepted
	`E_NOT_OK`	Transmit request has been rejected
Description	This function starts a LinTP-Transmission if there is currently no other transmission ongoing on the channel identified by the PDU Id.	

5.2.3. Integration notes

5.2.3.1. Exclusive areas

This section describes the exclusive areas used by the `LinIf` and `LinTp` module.

5.2.3.1.1. SCHM_LINIF_EXCLUSIVE_AREA_0

Protected data structures	All shared data that shall be protected from mutual access.
Recommended locking mechanism	This exclusive area must always be protected by a locking mechanism. The options for locking are described in the `EB tresos AutoCore Generic documentation`. Refer to the section `Mapping exclusive areas in the basic software modules` in the `Integration notes` section for details.

5.2.3.2. Production errors

LINIF_E_RX_CHECKSUM_ERROR	▶ LinIf_MainFunction
LINIF_E_RX_NO_RESPONSE_ERROR	▶ LinIf_MainFunction
LINIF_E_TX_BIT_ERROR	▶ LinIf_MainFunction

5.2.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section `Memory mapping and compiler abstraction` in the `Integration notes` section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
CODE
CONST_32

VAR_INIT_16
VAR_INIT_8
VAR_CLEARED_8
VAR_CLEARED_UNSPECIFIED
CONFIG_DATA_UNSPECIFIED
VAR_INIT_UNSPECIFIED
CONST_UNSPECIFIED
NOTIF_CALLOUT_CODE

5.2.3.4. Integration requirements

WARNING 

Integration requirements list is not exhaustive

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user's guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

5.2.3.4.1. lim.LinIf.EB_INTREQ_LinIf_0001

Description	Some LIN Driver API functions must support being called within an interrupt lock Description: The following LIN Driver API functions must support being called by the LinIf within a global interrupt lock: Lin_Wakeup() Lin_WakeupInternal()
Rationale	The LinIf makes calls to these functions in its critical section to make sure, that the state is consistent to the LIN Driver. As it is assumed, that the LinIf critical sections are configured as global interrupt locks this means that these functions must support being called in such an interrupt lock situation.

5.2.3.4.2. lim.LinIf.EB_INTREQ_LinIf_0002

Description	LinIf shall not be initialized as operational Description: The LinIf configuration parameter LinIfStartupState shall only be configured to LINIF_CHANNEL_SLEEP. Configuring it to LINIF_CHANNEL_OPERATIONAL is obsolete.

Rationale	Following bugzilla shall be respected: https://bugzilla.autosar.org/show_bug.cgi?id=73095

5.2.3.4.3. lim.LinIf.EB_INTREQ_LinIf_0003

Description	Slave associated response_error signal access Description: The configuration needs to ensure that LinIf is the only user that has write-access to the response_error signal.

5.2.3.4.4. lim.LinIf.EB_INTREQ_LinIf_0004

Description	Slave associated response_error signal access Description: For a given LinTp channel associated to a LinIf Slave channel a single LinTpRxNSdu has to be configured. It shares all physical and functional requests.

5.2.3.4.5. lim.LinIf.EB_INTREQ_LinIf_0005

Description	Expected values by LinIf_SetPIDTable Description: When the API LinIf_SetPIDTable() is used the provided PID list shall not include the PIDs for MRF and SRF.

5.2.3.4.6. lim.LinIf.EB_INTREQ_LinIf_0006

Description	Incomplete wakeup and transition to sleep Description: The following scenario can happen: 1. A wakeup process is incomplete because the first header after wake-up request is not arriving, so the LinIfBusIdleTimeoutPeriod expires. 2. When LinIf-BusIdleTimeoutPeriod expires, a go-to-sleep process starts that interrupts the wake-up process. If the driver returns E_NOT_OK, LinIf will end up stuck in OPERATIONAL state and LinSM will end up stuck in WAKEUP state. Given that go-to-sleep has failed, latest request from ComM will be FULL_COM. so LinSM should not transition to NO_COM, but considering the bus is idle for more than LinIfBusIdleTimeout-Period, LinSM transition to FULL_COM is not appropriate. Rationale: Theoretically, the driver should not reject the sleep (return E_NOT_OK) unless there's an invalid call (development error). Also, not receiving a header from the master for a long period of time is also considered a problem. The scenario above is considered a double-fault and if considered necesary by the project the issue can be avoided by configuring LinIfBusIdleTimeoutPeriod to a value between: lowerLimit = ((LinSM-ModeRequestRepetitionMax + 1) * LinSMConfirmationTimeout) and upperLimit = (((LinSMModeRequestRepetitionMax + 1) * LinSMConfirmationTimeout) + LinSMSilenceAfterWakeupTimeout) If LinIfBusIdleTimeoutPeriod needs to be greater than up-

	perLimit, then make sure it is between than (n * upperLimit + lowerLimit) and ((n+1) * upperLimit) (where n is the number of times LinSMSilenceAfterWakeupTimeout expired). Basically, the Bus Idle Timeout should not expire during a Wakeup request + LinSMConfirmationTimeout. For a visual description of this integration requirement, see LinSM limitation (Incomplete wakeup and transition to sleep).

5.3. LinSM

5.3.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
LinSMDefensiveProgramming	1..1	**Label:** Defensive Programming Options Parameters for defensive programming
LinSMConfigSet	1..1	This container describes the configuration set of LinSM. This is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
LinSMGeneral	1..1	This container contains general parameters of LIN State Manager module.
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

Parameters included	
Parameter name	**Multiplicity**
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	**IMPLEMENTATION_CONFIG_VARIANT**
Label	Config Variant
Description	Configuration variant. Only pre-compile configuration is supported.

Multiplicity	1..1
Type	ENUMERATION
Default value	VariantPreCompile
Range	VariantPreCompile

5.3.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	**ArMajorVersion**
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArMinorVersion**
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.

Multiplicity	1..1
Type	INTEGER_LABEL
Default value	3
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArPatchVersion**
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	3
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	4
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	SwPatchVersion
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	27
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ModuleId
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	141
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	VendorId
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	Release
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	

Configuration class	**PublishedInformation:**	
Origin	Elektrobit Automotive GmbH	

5.3.1.2. LinSMDefensiveProgramming

Parameters included	
Parameter name	**Multiplicity**
LinSMDefProgEnabled	1..1
LinSMPrecondAssertEnabled	1..1
LinSMPostcondAssertEnabled	1..1
LinSMStaticAssertEnabled	1..1
LinSMUnreachAssertEnabled	1..1
LinSMInvariantAssertEnabled	1..1

Parameter Name	**LinSMDefProgEnabled**	
Label	Enable Defensive Programming	
Description	Enables or disables the defensive programming feature for the module LinSM. Note: This feature is dependent on the use of the development error detection module. To use the defensive programming feature, proceed as follows: 1. Enable development error detection 2. Enable defensive programming 3. Enable assertions as required	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinSMPrecondAssertEnabled**
Label	Enable Precondition Assertions
Description	Enables handling of precondition assertion checks reported from the module LinSM. Dependency on parameter(s):

	► Enable Development Error Detection (`LinSMDevErrorDetect`): must be enabled ► Enable Defensive Programming (`LinSMDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**LinSMPostcondAssertEnabled**
Label	Enable Postcondition Assertions
Description	Enables handling of postcondition assertion checks reported from the module LinSM. Dependency on parameter(s): ► Enable Development Error Detection (`LinSMDevErrorDetect`): must be enabled ► Enable Defensive Programming (`LinSMDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**LinSMStaticAssertEnabled**
Label	Enable Static Assertions
Description	Enables handling of static assertion checks reported from the module LinSM. Dependency on parameter(s): ► Enable Development Error Detection (`LinSMDevErrorDetect`): must be enabled ► Enable Defensive Programming (`LinSMDefProgEnabled`): must be enabled
Multiplicity	1..1

Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinSMUnreachAssertEnabled**	
Label	Enable Unreachable Code Assertions	
Description	Enables handling of unreachable code assertion checks reported from the module LinSM. Dependency on parameter(s): ► Enable Development Error Detection (`LinSMDevErrorDetect`): must be enabled ► Enable Defensive Programming (`LinSMDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**LinSMInvariantAssertEnabled**	
Label	Enable Invariant Assertions	
Description	Enables handling of invariant assertion checks reported from functions of the module LinSM. Dependency on parameter(s): ► Enable Development Error Detection (`LinSMDevErrorDetect`): must be enabled ► Enable Defensive Programming (`LinSMDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Origin	Elektrobit Automotive GmbH

5.3.1.3. LinSMConfigSet

Containers included		
Container name	**Multiplicity**	**Description**
LinSMChannel	1..255	Describes each LIN channel the LinSM is connected to.

5.3.1.4. LinSMChannel

Containers included		
Container name	**Multiplicity**	**Description**
LinSMSchedule	1..254	The schedule references to a schedule that is located in the LinIf configuration.

Parameters included	
Parameter name	**Multiplicity**
LinSMConfirmationTimeout	1..1
LinSMSleepSupport	1..1
LinSMTransceiverPassiveMode	0..1
LinSMComMNetworkHandleRef	1..1
LinSMNodeType	1..1
LinSMSilenceAfterWakeupTimeout	1..1
LinSMModeRequestRepetitionMax	1..1

Parameter Name	**LinSMConfirmationTimeout**
Description	Timeout in seconds for the goto sleep, wakeup and schedule request calls to LinIf. The timeout must be longer than a goto-sleep command on the bus (i.e. it is bit rate dependent). It also must be longer than the expected duration between a schedule request and the next confirmation - that is, it must be longer than the runtime of the longest RUN_ONCE schedule table in the LinIf configuration. Alternatively, setting this parameter to 0 will disable the timeout.

Multiplicity	1..1	
Type	FLOAT	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**LinSMSleepSupport**
Description	Some LIN clusters do not need sleep, they will just shut off. This parameter will affect the behavior to achieve the 'full communication' and 'no communication' states. ► `true`: LinSM will call LinIf_Wakeup() or LinIf_GotoSleep() to change the communication state. ► `false`: LinSM will change the communication state without calling LinIf_-Wakeup() or LinIf_GotoSleep(). **Optimization Effect:** ► **Execution time reduction (code):** Disabling this parameter reduces the execution time of the module code. ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **ROM reduction (config):** Choosing a globally common value for this parameter reduces the ROM consumption of the module configuration.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**LinSMTransceiverPassiveMode**
Description	Selects STANDBY (true) or SLEEP (false) transceiver mode when entering LINSM_NO_COM.
Multiplicity	0..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile

Origin	AUTOSAR_ECUC

Parameter Name	**LinSMComMNetworkHandleRef**	
Description	Unique handle to identify one certain LIN network. Reference to one of the network handles configured in the ComM. **Optimization Effect:** ► **Execution time reduction (code):** Configuring consecutive channel IDs for the ComM channels referenced by LinSM reduces the execution time of the module code. ► **ROM reduction (code):** Configuring consecutive channel IDs for the ComM channels referenced by LinSM reduces the ROM consumption of the module code.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**LinSMNodeType**	
Description	Specifies the LIN node type of this channel.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	MASTER	
Range	MASTER SLAVE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**LinSMSilenceAfterWakeupTimeout**	
Description	Timeout in seconds after a failed wakeup sequence until a new wakeup process is started.	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.0	
Configuration class	**PreCompile:**	VariantPreCompile

Origin	AUTOSAR_ECUC

Parameter Name	**LinSMModeRequestRepetitionMax**	
Description	Specifies the maximal amount of mode request repetitions without a respective mode indication from the LinIf module until the LinSM module reports a development error to the DET and tries to go back to no communication.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.5. LinSMSchedule

Parameters included	
Parameter name	**Multiplicity**
LinSMScheduleIndex	1..1
LinSMScheduleIndexRef	1..1

Parameter Name	**LinSMScheduleIndex**	
Description	This index parameter can be used by the BswM as a SymbolicNameReference target. The LinSM just forwards the request from the BswM to LinIf. Note that the value of the LinSMScheduleIndex shall be the same as the value from the LinIf. **This parameter is currently not used by LinSM module. However for configuration compatibility with other modules, please configure LinSMScheduleIndex properly.**	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**LinSMScheduleIndexRef**
Description	Reference to a schedule table in the LinIf configuration.

	Optimization Effect: ► **Execution time reduction (code):** Configuring consecutive indices for the schedule tables referenced by LinSM reduces the execution time of the module code. ► **ROM reduction (code):** Configuring consecutive indices for the schedule tables referenced by LinSM reduces the ROM consumption of the module code.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.3.1.6. LinSMGeneral

Parameters included	
Parameter name	**Multiplicity**
LinSMDevErrorDetect	1..1
LinSMMainProcessingPeriod	1..1
LinSMVersionInfoApi	1..1
LinSMMultiCoreSupport	1..1

Parameter Name	**LinSMDevErrorDetect**	
Description	Switches the Development Error Detection and Notification ON or OFF. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Disabling this parameter reduces the execution time of the module code.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	LinSMMainProcessingPeriod	
Description	Fixed period that the MainFunction shall be called [s].	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.02	
Range	<=5.0 >=0.00001	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	LinSMVersionInfoApi	
Description	Switches the LinSM_GetVersionInfo function ON or OFF. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	LinSMMultiCoreSupport	
Description	Switches the LinSM MultiCore Support ON or OFF.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile

5.3.1.7. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
PbcfgMSupport	1..1

Parameter Name	PbcfgMSupport
Label	PbcfgM support
Description	Specifies whether or not the LinSM can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

5.3.2. Application programming interface (API)

5.3.2.1. Type definitions

5.3.2.1.1. LinSM_ModeType

Purpose	Type to report the current mode to the BswM.
Type	`uint8`
Description	Range: ▶ LINSM_FULL_COM ▶ LINSM_NO_COM

5.3.2.2. Macro constants

5.3.2.2.1. FULL_COM_STORED

Purpose	full communication stored

Value	1U

5.3.2.2.2. LINSM_E_ALREADY_INITIALIZED

Purpose	DET Error Code.
Value	0x10U
Description	Initialization API is used when already initialized ► This error is not used as it contradicts LINSM043.

5.3.2.2.3. LINSM_E_CONFIRMATION_TIMEOUT

Purpose	DET Error Code.
Value	0x50U
Description	Timeout of the callbacks from LinIf

5.3.2.2.4. LINSM_E_NONEXISTENT_NETWORK

Purpose	DET Error Code.
Value	0x20U
Description	Referenced channel or network does not exist (identification is out of range)

5.3.2.2.5. LINSM_E_NOT_IN_RUN_SCHEDULE

Purpose	DET Error Code.
Value	0x51U
Description	LinSM_ScheduleRequest called for a channel not in FULL_COM state

5.3.2.2.6. LINSM_E_PARAMETER

Purpose	DET Error Code.
Value	0x30U
Description	API service called with wrong parameter

5.3.2.2.7. LINSM_E_PARAMETER_POINTER

Purpose	DET Error Code.
Value	0x40U
Description	API service called with invalid pointer

5.3.2.2.8. LINSM_E_REPETITION_MAX_REACHED

Purpose	DET Error Code:.
Value	0x61U
Description	Repetition max was excedeed

5.3.2.2.9. LINSM_E_UNEXPECTED_CALLOUT

Purpose	DET Error Code:.
Value	0x60U
Description	LinIf signalled an unexpected confirmation

5.3.2.2.10. LINSM_E_UNINIT

Purpose	DET Error Code.
Value	0x00U
Description	API called without initialization of LinSM

5.3.2.2.11. LINSM_FULL_COM

Purpose	full communication (used for LinSM_ModeType and channel state)
Value	1U

5.3.2.2.12. LINSM_GOTO_SLEEP

Purpose	goto sleep in progress (used for internal channel state)
Value	3U

5.3.2.2.13. LINSM_NO_COM

Purpose	no communication (used for LinSM_ModeType and channel state)
Value	2U

5.3.2.2.14. LINSM_SID_GETCURRENTCOMMODE

Purpose	Service Id of LinSM_GetCurrentComMode().
Value	0x11U

5.3.2.2.15. LINSM_SID_GETVERSIONINFO

Purpose	Service Id of LinSM_GetVersionInfo().
Value	0x02U

5.3.2.2.16. LINSM_SID_GOTOSLEEPCONF

Purpose	Service Id of LinSM_GotoSleepConfirmation().
Value	0x22U

5.3.2.2.17. LINSM_SID_GOTOSLEEPINDICATION

Purpose	Service Id of LinSM_GotoSleepIndication().
Value	0x03U

5.3.2.2.18. LINSM_SID_INIT

Purpose	Service Id of LinSM_Init().
Value	0x01U

5.3.2.2.19. LINSM_SID_MAINFUNCTION

Purpose	Service Id of LinSM_MainFunction().
Value	0x30U

5.3.2.2.20. LINSM_SID_REQUESTCOMMODE

Purpose	Service Id of LinSM_RequestComMode().
Value	0x12U

5.3.2.2.21. LINSM_SID_SCHEDULEREQUEST

Purpose	Service Id of LinSM_ScheduleRequest().
Value	0x10U

5.3.2.2.22. LINSM_SID_SCHEDULEREQUESTCONF

Purpose	Service Id of LinSM_ScheduleRequestConfirmation().
Value	0x20U

5.3.2.2.23. LINSM_SID_WAKEUPCONFIRMATION

Purpose	Service Id of LinSM_WakeupConfirmation().
Value	0x21U

5.3.2.2.24. LINSM_WAKEUP

Purpose	wakeup in progress (used for internal channel state)
Value	0U

5.3.2.2.25. NOTHING_STORED

Purpose	no stored mode
Value	0U

5.3.2.2.26. NO_COM_STORED

Purpose	no communication stored
Value	2U

5.3.2.3. Functions

5.3.2.3.1. LinSM_GetCurrentComMode

Purpose	Function to query the current communication mode.	
Synopsis	`Std_ReturnType` **`LinSM_GetCurrentComMode`** `( NetworkHandleType network , ComM_ModeType * mode );`	
Service ID	0x11	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`network`	Identification of the LIN channel
Parameters (out)	`mode`	Returns the active mode, see ComM_ModeType for descriptions of the modes
Return Value	Result of operation	
	`E_OK`	Ok
	`E_NOT_OK`	Not possible to perform the request, e.g. not initialized.
Description	Returns the current communication mode for the specified channel.	

5.3.2.3.2. LinSM_GetVersionInfo

Purpose	Get version information of the LinSM module.	
Synopsis	`void` **`LinSM_GetVersionInfo`** `( Std_VersionInfoType * versioninfo );`	
Service ID	0x02	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (out)	`versioninfo`	Pointer to where to store the version information of this module.
Description	This service returns the version information of this module. The version information includes: ► Vendor Id ► Module Id ► Vendor specific version numbers	

5.3.2.3.3. LinSM_GotoSleepConfirmation

Purpose	Confirmation callout for GotoSleep transition.	
Synopsis	`void` **`LinSM_GotoSleepConfirmation`** `( NetworkHandleType network , boolean success );`	
Service ID	0x22	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`network`	Identification of the LIN channel
	`success`	True if goto sleep was successfully sent, false otherwise
Description	The LinIf will call this callback when the go to sleep command is sent successfully or not sent successfully on the network.	

5.3.2.3.4. LinSM_GotoSleepIndication

Purpose	Indication callout for GotoSleep transition.	
Synopsis	`void` **`LinSM_GotoSleepIndication`** `( NetworkHandleType Channel );`	
Service ID	0x03	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`Channel`	Identification of the LIN channel
Description	The LinIf will call this callback when the go to sleep command is received on the network or a bus idle timeout occurs. Only applicable for LIN slave nodes.	

5.3.2.3.5. LinSM_Init

Purpose	Initializes the LinSM module.
Synopsis	`void` **`LinSM_Init`** `( const LinSM_ConfigType * ConfigPtr );`
Service ID	0x01
Sync/Async	Synchronous

Reentrancy	Non reentrant	
Parameters (in)	`ConfigPtr`	Pointer to the LinSM configuration (ignored)
Description	This function initializes the LinSM. Note that the ConfigPtr parameter is ignored by this implementation as post-build configuration is not supported.	

5.3.2.3.6. LinSM_MainFunction

Purpose	Cyclic MainFunction for the LIN State Manager.
Synopsis	`void` **`LinSM_MainFunction`** `( void );`
Service ID	0x30
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	Periodic function that runs the timers of different request timeouts This function must be called cyclically using a fixed time period specified in LinSMMainProcessingPeriod.

5.3.2.3.7. LinSM_RequestComMode

Purpose	Requesting of a communication mode by ComM.	
Synopsis	`Std_ReturnType` **`LinSM_RequestComMode`** `( NetworkHandleType network , ComM_ModeType mode );`	
Service ID	0x12	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different LIN channels	
Parameters (in)	`network`	Identification of the LIN channel
	`mode`	Requested mode
Return Value	Result of operation	
	`E_OK`	Request accepted
	`E_NOT_OK`	Not possible to perform the request, e.g. not initialized.
Description	The mode switch will not be made instantly. The LinSM will notify the ComM when mode transition is made.	

5.3.2.3.8. LinSM_ScheduleRequest

Purpose	Change schedule table for a LIN channel.	
Synopsis	`Std_ReturnType` **`LinSM_ScheduleRequest`** `( NetworkHandleType network , LinIf_SchHandleType schedule );`	
Service ID	0x10	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different LIN channels	
Parameters (in)	`network`	Identification of the LIN channel
	`schedule`	Index of the new Schedule table
Return Value	Result of operation	
	`E_OK`	Schedule table request has been accepted.
	`E_NOT_OK`	Schedule table switch request has not been accepted due to one of the following reasons: * LinSM has not been initialized * referenced channel does not exist (identification is out of range) * Referenced schedule table does not exist (identification is out of range) * Sub-state is not LINSM_FULL_COM
Description	The upper layer requests a schedule table to be changed on one LIN channel. This services delegates the schedule request to the LinIf.	

5.3.2.3.9. LinSM_ScheduleRequestConfirmation

Purpose	Confirmation callout for schedule table changes.	
Synopsis	`void` **`LinSM_ScheduleRequestConfirmation`** `( NetworkHandleType network , LinIf_SchHandleType schedule );`	
Service ID	0x20	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`network`	Identification of the LIN channel

	`schedule`	Index of the new active Schedule table
Description	The LinIf module will call this callback when the new requested schedule table is active.	

5.3.2.3.10. LinSM_WakeupConfirmation

Purpose	Confirmation callout for WakeUp.	
Synopsis	`void` **`LinSM_WakeupConfirmation`** `( NetworkHandleType network , boolean success );`	
Service ID	0x21	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`network`	Identification of the LIN channel (LinSMChannelIndex)
	`success`	True if wakeup was successfully sent, false otherwise
Description	This callout must be called by the LinIf after a wakeup request has been received using LinIf_Wakeup. It signals if the wakeup request was successful. Note that the LinIf has to call this function in any case if the call to LinIf_Wakeup has returned E_OK. That means, even if there is no wakeup request carried out on the bus (because the LinIf channel is already awake), the confirmation must be called nonetheless.	

5.3.3. Integration notes

5.3.3.1. Exclusive areas

This section describes the exclusive areas used by the `LinSM` module.

5.3.3.1.1. SCHM_LINSM_EXCLUSIVE_AREA_0

Protected data structures	All shared data that shall be protected from mutual access.

Recommended locking mechanism	This exclusive area must always be protected by a locking mechanism. The options for locking are described in the `EB tresos AutoCore Generic documentation`. Refer to the section `Mapping exclusive areas in the basic software modules` in the `Integration notes` section for details.

5.3.3.2. Production errors

Production errors are not reported by the `LinSM` module.

5.3.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section `Memory mapping and compiler abstraction` in the `Integration notes` section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
CODE
VAR_CLEARED_UNSPECIFIED
VAR_INIT_8
CONFIG_DATA_UNSPECIFIED

5.3.3.4. Integration requirements

WARNING 

Integration requirements list is not exhaustive

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user's guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

5.3.3.4.1. lim.LinSM.EB_INTREQ_LinSM_0001

Description	The LinSM_RequestComMode function is non-reentrant if called for a transition from LINSM_FULL_COM to LINSM_NO_COM state for a channel that uses sleep support.

	During such a transition, the LinIf function LinIf_GotoSleep must be called which itself is non-reentrant.
Rationale	

5.3.3.4.2. lim.LinSM.EB_INTREQ_LinSM_0002

Description	LinSM_ScheduleRequest is non-reentrant for the same LIN channel. According to LINSM113, the LinSM_ScheduleRequest function shall be reentrant. Contrary to this, the LinSM_ScheduleRequest function implementation is non-reentrant for the same LIN channel.
Rationale	

5.3.3.4.3. lim.LinSM.EB_INTREQ_LinSM_0003

Description	If the LinSM schedule table will be created for each channel, the name NULL_-SCHEDULE has to be extended with the channel index (_0 for first channel, _1 for second channel and so on). The extension will be related to the index, not to the name of the channel, so it is needed to have the channels 0-based and consecutive, if numbering is used for channels.
Rationale	Considering that LinSM does not provide an ID for its channels, the index of the channel (meaning its order in the channel list) will be used to differentiate the channels.

6. Bibliography

Bibliography

[1] *LIN Specification Package Revision 2.0*, Publish date: September 23, 2003, Publisher: LIN Consortium

[2] *Road Vehicles - Diagnostics on Controller Area Networks (CAN) - Part 2: Network Layer Services*,
1, rue de Varembe, Case postale 56
1211 Geneva 20, Switzerland, Publish date: 2003, Issue Version ISO/DIS 15765-2.2, Publisher: ISO (International Organization for Standardization)