Elektrobit

EB tresos® AutoCore Generic 8

CAN Stack documentation

product release 8.8.4

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 CAN Stack documentation

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

This document provides:

- Chapter 3, “ACG8 CAN Stack release notes“: release notes for the ACG8 CAN Stack modules
- Chapter 4, “ACG8 CAN Stack user guide“: containing background information and instructions
- Chapter 5, “ACG8 CAN Stack module references“: information about configuration parameters and the application programming interface

2. Supported features

2.1. Overview

This chapter provides an overview of the ACG8 CAN Stack and the features that are currently supported.

Section 2.2, “Supported CanAs features” contains an overview of `CanAs` features.

Section 2.3, “Supported CanIf features” contains an overview of `CanIf` features.

Section 2.4, “Supported CanNm features” contains an overview of `CanNm` features.

Section 2.5, “Supported CanSM features” contains an overview of `CanSM` features.

Section 2.6, “Supported CanTp features” contains an overview of `CanTp` features.

2.2. Supported CanAs features

- **HOH assignment:** Assign messages (PDUs) to CAN hardware object handles (HOHs)
- **Optimization of filter mask and CAN ID:** Optimize the filter masks and CAN identifiers of PDUs
- **Bit timing:** Calculate bit timings for a controller

2.3. Supported CanIf features

- **Support for post-build:** Support for handling post-build loadable and selectable configuration.
- **Support for truncation:** Support for configurable truncation options on a transmitting PDU. You can choose between truncating up to the CAN frame length (8 bytes for CAN 2.0, 64 bytes for CAN FD) or up to the configured PDU length.
- **Support for CanIf custom hook in interrupt context:** Support for a callback in CanIf (similar to a CDD) that the module calls when a successful reception is signaled. The information passed is the received message content, i.e. CAN_ID, data, and received CAN index.
- **Support for queue size measurement in decoupled processing:** Support for measurement and reporting of the remaining queue size during decoupled processing.

- **Multicore distribution of the CAN stack along network boundaries:** CAN channels, together with their resources (controllers, HOHs, PDUs, MainFunctions, etc.), can be allocated and work on different cores/ partitions.

2.4. Supported CanNm features

- **Support Autosar network management coordination algorithm**: Support of transmission of periodic Network Management PDUs as long as the bus-communication is requested and detection of Network Management PDUs signalling that other nodes request bus-communication.
- **Support operational modes**: Support of operational modes Network Mode (with internal states Repeat Message State, Normal Operation State, Ready Sleep State), Prepare Bus-Sleep Mode, Bus-Sleep Mode according to Autosar specifications.
- **Support for communication startup**: Support for interface to the upper layer to initiate transmission of NM PDUS due to a user(s) requesting communication.
- **Support for communication shutdown**: Support for interface to the upper layer to stop transmission of NM PDUS due to a user(s) not requesting communication.
- **Support for communication passive wakeup**: Support interface to the upper layer to initiate communication capabilities due to a wakeup event network start or network restart indication.
- **Support for passive mode**: Support for nodes with transmission of network management PDUs disabled.
- **Support for detection of remote sleep**: Support for detecting if all other nodes are ready to sleep.
- **Support for state change notification**: Support for notification function to Nm called when CanNm state is changed.
- **Support for car wakeup**: Support of CarWakeup bit as part of the network management PDU and car wakeup callout function.
- **Support for bus-load reduction mechanism**: Support mechanism to reduce the number of transmitted NM messages for realizing network management algorithm.
- **Support for user data in NM messages**: Support for user data in NM messages. User data can be updated either using CanNM interfaces or communication stack by collecting the data from and an I-PDU.
- **Support for PDU length higher than 8 bytes**: Support payloads higher than 8 bytes of the NM messages (when supported by the bus type).
- **Support for communication control**: Support for interfaces to enable/disable transmission of NM messages.
- **Support for partial networking**: Support for updating and filtering partial network information as part of the NM messages.
- **Support for spontaneous transmission**: Support interface to trigger spontaneous transmission of an NM message with the provided NM user data.

- **Support for immediate transmission**: Support for transmission of a predefined number of NM messages with an different cycle time when entering RepeatMessage state from BusSleep state or PrepareBusSleep state.
- **Support for immediate restart**: Support transmission of an NM messages when the network has been requested in the PrepareBusSleep state.
- **Support of RepeateMsgInd|NodeDetection|NodeIdEnabled channel based configurable**: Support per channel configuration of parameters CanNmRepeatMsgIndEnabled, CanNmNodeDetectionEnabled, CanNmNodeIdEnabled.
- **Support for post-build**: Support for handling post-build loadable and selectable configuration.
- **Multicore distribution of the CAN stack along network boundaries:** CAN channels, together with their resources (controllers, HOHs, PDUs, MainFunctions, etc.), can be allocated and work on different cores/ partitions.

2.5. Supported CanSM features

- **Support for post-build:** Support for handling post-build selectable configuration.
- **Multicore distribution of the CAN stack along network boundaries:** CAN channels, together with their resources (controllers, HOHs, PDUs, MainFunctions, etc.), can be allocated and work on different cores/ partitions.

2.6. Supported CanTp features

- **Support for post-build:** Support for handling post-build loadable and selectable configuration.
- **Multicore distribution of the CAN stack along network boundaries:** CAN channels, together with their resources (controllers, HOHs, PDUs, MainFunctions, etc.), can be allocated and work on different cores/ partitions.

3. ACG8 CAN Stack release notes

3.1. Overview

This chapter provides the ACG8 CAN 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: 28.2.0 b211016-0103

3.2.2. AUTOSAR modules

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

Module name	AUTOSAR version and revision	SWS version and revision	Module version	Supplier
CanIf	4.0.3 []	5.0.0 [0000]	6.10.19	Elektrobit Automotive GmbH
CanNm	4.0.3 []	3.3.0 [3]	6.19.8	Elektrobit Automotive GmbH
CanSM	4.0.3 []	2.2.0 [0000]	3.7.7	Elektrobit Automotive GmbH
CanTp	4.0.3 []	4.0.0 [0000]	6.8.45	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
CanAs	2.4.24	Elektrobit Automotive GmbH

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. CanAs module release notes

- Module version: 2.4.24.B466224
- Supplier: Elektrobit Automotive GmbH

3.3.1.1. Change log

This chapter lists the changes between different versions.

Module version 2.4.24
2021-07-28

- Added support for Can schema files defining CanControllerFdBaudrateConfig as a container list that can hold at most one element

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

Module version 2.4.23
2021-03-05

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

Module version 2.4.22
2021-01-22

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

Module version 2.4.21
2020-12-18

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

Module version 2.4.20
2020-09-25

- ASCCANAS-427 Fixed known issue: Can Buffer Assignment AutoConfigure Wizard is invoked with incorrect parameter values
- Added support to write CanObjectId values of outbound HOHs depending on the priority of the PDUs they send

Module version 2.4.19
2020-06-19

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

Module version 2.4.18
2020-05-22

- Added support for AUTOSAR 4.4.x Can modules

Module version 2.4.17
2020-02-21

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

Module version 2.4.16
2019-06-14

- Internal module improvement. This module version update does not affect module functionality
- Added support for AUTOSAR 4.3.x Can modules

Module version 2.4.15
2019-02-15

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

Module version 2.4.14
2018-10-26

- Added dialog, warnings and error messages for handling of MCU HOH assignment and better usability

Module version 2.4.13
2018-09-28

- Added the command line access to the CanAs Buffer Assignment function via GuidedConfigWizard

Module version 2.4.12
2018-07-27

- ASCCANAS-383 Fixed known issue: Assignment Failure in multi-CC AutoAssignment leaves PDU and HOH grids in inconsistent state
- Internal module improvement. This module version update does not affect module functionality

Module version 2.4.11
2018-06-22

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

Module version 2.4.10
2018-05-25

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

Module version 2.4.9
2018-02-16

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

Module version 2.4.8
2017-09-21

- Removed warnings for HOHs with overlapping CAN Id ranges, added a warning if one HOH covers all CAN Ids of a second HOH

Module version 2.4.7
2017-03-31

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

Module version 2.4.6
2017-03-03

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

Module version 2.4.5
2017-02-03

- The PDU to CanController assignment is now stored in configurations with two or more CanControllers even if the PDU has no HOH assignment

Module version 2.4.4
2016-10-07

- ASCCANAS-345 Fixed known issue: CAN Buffer Assignment Editor stops with an error when writing an ASR 4.2.2 Can configuration

Module version 2.4.3
2016-09-09

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

Module version 2.4.2
2016-08-05

- Added support for CAN FD PDUs

Module version 2.4.1
2016-07-01

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

Module version 2.4.0
2016-05-25

- Added support for AUTOSAR 4.2.x Can modules

Module version 2.3.2
2016-04-29

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

Module version 2.3.1
2016-04-01

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

Module version 2.3.0
2016-02-05

- ASCCANAS-313 Fixed known issue: CanObjectId parameters are configured incorrectly if two or more CanControllers exist
- Add support for new enum values in CanIfRxPduCanIdType and CanIfTxPduCanIdType: STANDARD_FD_CAN, STANDARD_NO_FD_CAN, EXTENDED_FD_CAN, EXTENDED_NO_FD_CAN

Module version 2.2.1
2015-11-06

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

Module version 2.2.0
2015-09-18

- Implemented Bit Timing editor page

Module version 2.1.10
2015-06-19

- ASCCANAS-295 Fixed known issue: AutoAssignment fails with Platforms supplying only Full Rx HOHs

Module version 2.1.9
2015-02-20

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

Module version 2.1.8
2014-10-03

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

Module version 2.1.7
2014-04-25

- Added support for CAN modules supplying two or more types of controllers with different buffer/HOH properties

Module version 2.1.6
2013-10-11

- Added bulk change support in PDU and HOH tables

Module version 2.1.5
2013-06-14

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

Module version 2.1.4
2013-05-08

- Removed support for AUTOSAR 3.x configurations which became obsolete

Module version 2.1.3
2013-02-08

- ASCCANAS-235 Fixed known issue: `NullPointerException` when configurations are written containing at least one Tx Pdu without HOH assignment

Module version 2.1.2
2012-10-12

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

Module version 2.1.1
2012-06-15

- Extended AUTOSAR 4.0 back end to support `CanFilterMaskRef` as single element reference list

Module version 2.1.0
2012-05-16

- Configuration results are now shown in the results view
- ASCCANAS-205 Fixed known issue: Reading in previously written Can/CanIf configurations may fail
- Adaptations due to CanIf parameter changes according to AUTOSAR 4.0 rev 3

Module version 2.0.3
2012-03-16

- Automatic buffer assignment algorithm can now be configured to use dedicated HTHs for standard and extended CAN Id PDU transmission

Module version 2.0.2
2012-01-20

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

Module version 2.0.1
2011-09-30

- ASCCANAS-180 Fixed known issue: PDU to HTH references may be set incorrectly if CAN Id range PDU is contained in CanIf configuration

Module version 2.0.0
2011-09-02

- Initial AUTOSAR 4.0 version

3.3.1.2. New features

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

3.3.1.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

3.3.1.4. Deviations

This module is not part of the AUTOSAR specification.

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.

- Automatic Assignment takes long/does not always find assignment solutions in large configurations

  Description:

  If a configuration consists of a large number of PDUs (>80), the automatic HOH assignment algorithm may take several seconds, a progress bar is being displayed to you while the algorithm is running. Moreover it can happen in large configurations, that the algorithm does not find a valid solution.

  Rationale:

  The total number of solutions for the HOH assignment problem grows exponentially with the number of processed PDUs, as does the processing time that would be necessary to examine all possible solutions. Therefore, it necessary to examine only a subset of all solutions. If no valid solution is contained in the subset, the algorithm fails to find a valid solution.
- Controller Network Assignments cannot be retrieved in AUTOSAR version 4.0

  Description:

  The CanIf does not contain information about the network in which a Can controller resides. In consequence, the Can Assistant cannot determine the network of a controller, leading to erroneous assignments in the Automatic Assignment when two or more controllers are assigned to the same network.

  Rationale:

  The information on how to retrieve the network of a Controller has not been fully clarified. As a fallback solution, the Can Assistant assumes that each controller is attached to a dedicated network in the case that two or more controllers exist.
- Only one element CanIfBufferHthRef element per CanIfBufferCfg container is supported in AUTOSAR 4.-0 CanIf configurations

  Description:

  When AUTOSAR 4.0 CanIf configurations are read in or written out, only the first CanIfBufferCfg element is read and written, i.e. it is not possible to assign more than one HTH to a CanIfBufferCfg, nor is it possible to properly read in such CanIf configurations.

3.3.1.6. Open-source software

CanAs does not use open-source software.

3.3.2. CanIf module release notes

- AUTOSAR R4.0 Rev 3

- AUTOSAR SWS document version: 5.0.0
- Module version: 6.10.19.B466224
- Supplier: Elektrobit Automotive GmbH

3.3.2.1. Change log

This chapter lists the changes between different versions.

Module version 6.10.19
2021-10-08

- Added support for distribution of CAN stack along network boudaries for decoupled processing.
- ASCCANIF-1548 Fixed known issue: CanIf does not notify the upper layer with TxConfirmation with a result E_NOT_OK

Module version 6.10.18
2021-09-17

- Extend Reliable TxConfirmation for CanTSyn.
- Added support for distribution of CAN stack along network boudaries.

Module version 6.10.17
2021-08-20

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

Module version 6.10.16
2021-07-28

- ASCCANIF-1528 Added support for reporting Can frames to non AUTOSAR mirroring concepts.

Module version 6.10.15
2021-06-25

- Added the capability of having Tx callback function with result (E_OK/E_NOT_OK) for CDD and J1939 upper layer only.

Module version 6.10.14
2021-04-30

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

Module version 6.10.13
2021-04-09

- ASCCANIF-1502 Fixed known issue: HTH offset is not considered during Tx buffering on multiple CAN drivers

Module version 6.10.12
2021-03-05

- Added support for queue size measurement for decoupled processing.
- ASCCANIF-1499 Fixed known issue: Duplicate CanObjectId assigns PDUs to wrong HOH

Module version 6.10.11
2021-02-12

- Added truncation options for a transmitting PDU.
- Changed wakeup behavior.

Module version 6.10.10
2021-01-22

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

Module version 6.10.9
2020-12-18

- ASCCANIF-1469 Fixed known issue: CanIf requires multiple transceivers in order to generate the CanIf_-GetTrcvMode() interface needed by Mirror in multi-core context
- ASCCANIF-1474 Fixed known issue: Mirror support related out-of-bounds access
- Added support for disabling/Enabling Software Filtering feature for optimization purpose

- Added support for CanIf custom hook on succesful reception

Module version 6.10.8
2020-10-23

- Added support for Defensive Programming.

Module version 6.10.7
2020-09-25

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

Module version 6.10.6
2020-08-28

- ASCCANIF-1449 Fixed known issue: Tx Confirmations can be lost with CanIfDecoupledProcessing turned ON

Module version 6.10.5
2020-07-31

- Tresos:Automatically calculation of settings of parameters

Module version 6.10.4
2020-06-19

- ASCCANIF-1240 Fixed known issue: Failing module initialization due to duplicate post-build configuration

Module version 6.10.3
2020-05-22

- Added support for J1939Nm and J1939Tp as standard upper layers.

Module version 6.10.2
2020-04-24

- Added support for Autosar HandleId concept for CDDs.

- ASCCANIF-1406 Fixed known issue: <UpperLayer>_RxIndication may not be called with software filtering and CanIdMask.

Module version 6.10.1
2020-03-25

- On the Tx part, frames that are larger than 64 bytes (FD)/8 bytes (nonFD), are now truncated to the maximum length and transmitted.

Module version 6.10.0
2020-02-21

- Provided support for multiple CAN drivers.
- ASCCANIF-1403 Fixed known issue: CanIf fails to compile if CanIfPublicTrcvWakeupSupport = TRUE and CanIfPublicTrcvSupport = FALSE.
- Provided ASR 4.3/4.4 MCAL Support.

Module version 6.9.24
2019-12-06

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

Module version 6.9.23
2019-10-11

- ASCCANIF-1379 Fixed known issue: Wrong CanTrcv vendor extension is generated in CanIf.

Module version 6.9.22
2019-09-06

- ASCCANIF-1372 Fixed known issue: <UpperLayer>_RxIndication() is not called with the newly translated Rx CanID.
- ASCCANIF-1374 Fixed known issue: Newly translated Rx CanID is not provided to Mirror.

Module version 6.9.21
2019-08-09

- ASCCANIF-1356 Fixed known issue: Incorrect loop variable type used in function CanIf_ResetMirrBuff.

Module version 6.9.20
2019-06-14

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

Module version 6.9.19
2019-05-17

- Added metadata support.

Module version 6.9.18
2019-04-18

- ASCCANIF-1356 Fixed known issue: Incompatibility between CanIf and 4.2.2 CanTrcv.

Module version 6.9.17
2019-03-22

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

Module version 6.9.16
2019-02-15

- Internal module improvement. This module version update does not affect module functionality.
- Added support for for Bus Mirroring

Module version 6.9.15
2019-01-24

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

Module version 6.9.14
2018-12-13

- ASCCANIF-1332 Fixed known issue: Incorrect configuration of PDU queues if PduQueueSize>255.
- ASCCANIF-1333 Fixed known issue: Interruption of CanIf_MainFunctionRx_Generic can lead to inconsistent queue status.

Module version 6.9.13
2018-11-23

- ASCCANIF-1329 Fixed known issue: Unconditional dependency on optional CanTrcv interfaces.

Module version 6.9.12
2018-10-26

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

Module version 6.9.11
2018-08-24

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

Module version 6.9.10
2018-06-22

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

Module version 6.9.9
2018-05-25

- Support for decoupled processing of reception and transmission confirmation events.

Module version 6.9.8
2018-04-20

- Add support for uint32 PduLengthType.

Module version 6.9.7
2018-02-16

- ASCCANIF-1286 Fixed known issue: Possible prolongation of Tx confirmation ISR when Tx buffering is enabled.

Module version 6.9.6
2018-01-19

- Added support of CanTSyn as upper layer.

Module version 6.9.5
2017-12-15

- ASCCANIF-1267 Fixed known issue: Compilation fails if DLC check is disabled for CAN 4.2 support.

Module version 6.9.4
2017-09-22

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

Module version 6.9.3
2017-07-28

- Post-build selectable support.

Module version 6.9.2
2017-05-05

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

Module version 6.9.1
2017-03-31

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

Module version 6.9.0
2017-03-10

- Added support for AUTOSAR 4.2.2 CAN drivers.

Module version 6.8.0
2017-03-03

- CAN and CANFD frames on the same CanId shall be received in two seperate RxPdus
- Move integration requirements to separate reqm file.

Module version 6.7.0
2017-02-03

- Do not enable partial networking TX filter after BusOff.
- Internal module improvement. This module version update does not affect module functionality.

Module version 6.6.2
2017-01-05

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

Module version 6.6.1
2016-11-04

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

Module version 6.6.0
2016-09-23

- Disable partial networking TX filter in case of ongoing transmission.

Module version 6.5.6
2016-05-25

- ASCCANIF-1169 Fixed known issue: CanIf may discard L-Pdu if CanIf_TxConfirmation() preempts CanIf_-Transmit()
- Added handle Id wizard support for configuration parameter CanIfCtrlId.

Module version 6.5.5
2016-02-05

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

Module version 6.5.4
2015-11-06

- Added a check to avoid multiple references to the same Can controller.

Module version 6.5.3
2015-06-19

- ASCCANIF-1139 Fixed known issue: CanIf passes the configured length value (DLC) to the upper layer modules
- Added support for 64 bytes CAN-FD frames
- ASCCANIF-1142 Fixed known issue: CanIf may include CanTrcv header file although CanTrcv support is disabled
- ASCCANIF-1145 Fixed known issue: CanIf does not disable PnTxFilter
- ASCCANIF-1147 Fixed known issue: CanIf does not include CanSM_Cbk.h

Module version 6.5.2
2015-02-20

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

Module version 6.5.1
2014-12-19

- Removed AUTOSAR 3.x compliant symbolic name value macros and updated the logic to only provide AUTOSAR 4.0.2 compliant macros if macro CANIF_PROVIDE_LEGACY_SYMBOLIC_NAMES is defined
- Added support for configurable mapping of CanIf_IsValidConfig function to dedicate memory section
- CanIf calls EcuM_ValidateWakeupEvent() only if wakeup has been successfully validated

Module version 6.5.0
2014-10-02

- ASCCANIF-1072 Fixed known issue: CanIfTxPduCfg containers cannot be added or removed at post-build time

- Introduced configuration container CanIfUpperLayerConfig to define upper layer modules at Pre-compile time
- Added proper name mangling for header files and API functions of CanTrcv
- Improved parameter description and error checks for configurations with multiple CanTrcv

Module version 6.4.4
2014-04-25

- ASCCANIF-1068 Fixed known issue: CanIf module reports an error if config time support is enabled
- ASCCANIF-1070 Fixed known issue: Build error due to missing file CanIf_PBcfg.c if code generation for CanIf is disabled and only post-build configuration is compiled

Module version 6.4.3
2013-11-15

- Removed warning from unused configuration parameter CanIfHrhCanHandleTypeRef
- Implemented support for CAN FD according to AUTOSAR R4.1 Rev 1

Module version 6.4.2
2013-10-11

- Changed MCG to introduce generation of XML code for Binary Code Generation

Module version 6.4.1
2013-06-28

- ASCCANIF-954 Fixed known issue: CanIf_SetControllerMode() does not report the DET error code CANIF_E_PARAM_CTRLMODE
- Implemented a configuration signature for consistency checks
- Updated memory sections of variables initialized during `CanIf_Init()` to NO_INIT
- Added a signature check to verify the precompile and link time configuration
- ASCCANIF-979 Fixed known issue: CanIf aborts compilation if AUTOSAR R4.0 Rev 3 compliant Can driver is used
- ASCCANIF-985 Fixed known issue: API services CanIf_ControllerBusOff() and CanIf_ControllerModeIndication() may report a DET error for a valid controller ID

- ASCCANIF-997 Fixed known issue: CanIf post-build-time configuration does not compile if used with PbcfgM
- ASCCANIF-999 Fixed known issue: CanIf calls Can_Write within a critical section causing system failure if interrupts are locked

Module version 6.4.0
2013-02-08

- ASCCANIF-934 Fixed known issue: A compiler error occurs if only a single HOH (sum of all HRHs and HTHs) is configured
- ASCCANIF-906 Fixed known issue: CanIfPublicHandleTypeEnum must be configured to UINT16 if more than 254 HOHs are used
- Updated CanIf to Can driver API according to AUTOSAR R4.0 Rev 3

Module version 6.3.0
2012-10-12

- Implementation of Handle-Id policy according to AUTOSAR R4.0 Rev 3
- The top-level structure of the software-component description in the ARXML files changed from `/AUTOSAR/CanIf`to `/AUTOSAR_CanIf`
- ASCCANIF-891 Fixed known issue: Condition for availability of configuration parameter CanIfDispatchUserConfirmPnAvailabilityName is incorrect

Module version 6.2.0
2012-06-20

- Update of config schema files according to AUTOSAR R4.0 Rev 3
- ASCCANIF-832 Fixed known issue: CanIf does not compile if wakeup validation and transmit confirmation in polling mode is disabled but partial network support is enabled
- Introduce post build configuration data structures
- Made config parameters CanIfHthCanHandleTypeRef and CanIfHrhCanHandleTypeRef optional
- ASCCANIF-856 Fixed known issue: HandleId wizard does not work for configuration parameter CanIfTxPduId

Module version 6.1.1
2012-04-20

- ASCCANIF-814 Fixed known issue: Eclipse framework crashes during execution of the handle ID generator

Module version 6.1.0
2012-03-16

- Update of Dem reporting to the AUTOSAR 4.0 CanIf specification
- Remove dependency from Can_CheckWakeup() and CanTrcv_CheckWakeup() if the functions are not used
- Update naming scheme for #defines for symbolic name values to AUTOSAR R4.0 Rev 3 naming scheme
- Add support of partial networking filters for Tx PDUs according to AUTOSAR R4.0 Rev 3
- Add API and call back functions related to partial networking according to AUTOSAR R4.0 Rev 3

Module version 6.0.0
2011-09-02

- Initial AUTOSAR 4.0 version.

3.3.2.2. New features

- Added support for Bus Mirroring.
- Added support for reporting Can frames to non AUTOSAR mirroring concepts.
- Added Metadata support.
- Added support for multiple CAN drivers.
- Added support for Autosar HandleId concept for CDDs.
- Added support for J1939Nm and J1939Tp as standard upper layers.
- Added support for automatically calculated values/settings of parameters.
- Added support for a custom notification to a CDD in case of a successful reception.
- Added truncation options for a transmitting PDU.
- Added support for queue size measurement for decoupled processing.
- Added the capability of having Tx callback function with result (E_OK/E_NOT_OK) for CDD and J1939 upper layer only.
- Added support for distribution of CAN stack along network boudaries.

3.3.2.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- [HisCanIf0001] Transmit buffering (extension to AUTOSAR specification)

  Description:

  If no transmit buffers are configured, the corresponding code and data are removed via the C preprocessor.

- [HisCanIf0002, HisCanIf0004] Single controller support (extension to AUTOSAR specification)

  Description:

  The configuration and the code were optimized for the use case of only one CAN controller. Unnecessary configuration data is removed and macros are used, to allow the compiler to better optimize loops.

  Please note however, that the controller configuration is not pre-compile-time and therefore the controller configuration structure is also used in this case.

- [HisCanIf0003, HisCanIf0005] Single driver support (extension to AUTOSAR specification)

  Description:

  This implementation only supports one CAN driver. It contains no code in form of loops, branches, etc. to support several CAN drivers.

  Please note however, that the driver configuration is not pre-compile-time and therefore there is a driver configuration structure in the C code.

- CAN ID parameter support for user specific upper layer callbacks

  Description:

  This implementation supports to provide the CAN ID as parameter to the receive indication callback function of user specific upper layers.

  The vendor specific parameter `CanIfUserUpperLayerConfig/CanIfUpperLayerUseCanId` was added to configure this feature.

- DLC check callout function support

  Description:

  This implementation supports the configuration of up to two DLC check callout functions for each upper layer. The first function is called, if a DLC check fails while the other is called in the case, that the DLC check succeeds.

  The vendor specific parameters `CanIfUserDlcErrorNotification` and `CanIfUserDlcPassedNotification` were added to the container `CanIfRxPduConfig` to configure the new callout functions.

- CAN ID translation callout support

  Description:

  This implementation supports the configuration of CAN ID translation callout functions. During run-time these functions can be used to change the CAN ID used on the CAN bus from the CAN ID that was originally configured (in parameter `CanIfTxPduCanId` and `CanIfRxPduCanId`).

  The vendor specific parameters `CanIfTranslateTxCanIdFunc` and `CanIfTranslateRxCanIdFunc` were added to the container `CanIfDispatchConfig` to configure the new callout functions.

- API and call back functions according to AUTOSAR 4.0 rev 3

  Description:

  The API functions `CanIf_ClearTrcvWufFlag`, `CanIf_CheckTrcvWakeFlag` and the call back functions `CanIf_ConfirmPnAvailability`, `CanIf_ClearTrcvWufFlagIndication`, `CanIf_CheckTrcvWakeFlagIndication` according to AUTOSAR 4.0 rev 3 were added.

- CanTrcv name mangling

  Description:

  This implementation supports to disable name mangling for header files and API functions of CanTrcv if a single CanTrcv driver is used.

  The vendor specific parameter `CanIfPublicCfg/CanIfSingleCanTrcvAPIInfixEnable` was added to configure this feature.

- Partial networking TX filter support (extension to AUTOSAR 4.2.2 specification)

  Description:

  According to AUTOSAR 4.2.2 a configured partial networking TX filter is enabled in the following cases: - at initialization - when CanIf_SetPduMode() is called with CANIF_TX_OFFLINE - when CanIf_SetControllerMode() is called with CANIF_CS_SLEEP The partial networking TX filter shall be deactivated after successful transmission of Wake-Up Frame. In case communication is ongoing and there is an successful reception of frame with PnTxFilter enabled, PnTxFilter shall be disabled. The PnTxFilter is in this case not needed since an Ack will be provided by an already active node.

  Please note however, that the transmit requests for other PDUs will be rejected until the configured PDU was sent. Only the very first PDU which initiates the Wake-up of the Network has to be the CanIfTxPduPnFilterPdu.

- TX Offline Active mode support

  Description:

The config parameter `CanIfTxOfflineActiveSupport` was introduced to determines whether TxOfflineActive feature is enabled or not. During CANIF_TX_OFFLINE_ACTIVE mode the upper layer has to handle the execution of the transmit confirmations immediately at the end of the transmit request.

- Support for decoupled processing of reception and transmission confirmation events

  Description:

  This implementation supports the configuration of decoupled processing of reception and transmission confirmation events on the MainFunction. This allows configuration of different PDUs to multiple CanIf_MainFunctionRx/Tx (with different period and priority) in order to limit the CPU load. During run-time these functions (CanIf_MainFunctionRx/Tx) can be used to process the reception/ transmission confirmation events received in interrupt context (CanIf_RxIndication/CanIf_TxConfirmation) using parameters `CanIfRxProcessing` and `CanIfTxProcessing`. The PDUs that are not assigned to a CanIf_MainFunctionRx/Tx() will be handled in interrupt context.

  The vendor specific parameter `CanIfDecoupledProcessingSupport` was added to enable this feature. The vendor specific containers `CanIfRxProcessing` and `CanIfTxProcessing` were added to the container `CanIfPublicCfg` to configure the new CanIf_MainFunctionRx and CanIf_MainFunctionTx functions. Each `CanIfRxProcessing` container contains the following parameters `CanIfRxPduQueueSize` and `CanIfPublicMaxPayloadQueueSize` which define the number of PDUs that can be added to the queue and the size of the data queue. Each `CanIfTxProcessing` container contains the parameter `CanIfTxPduQueueSize` which defines the number of PDUs that can be added to the queue. The vendor specific parameters `CanIfRxPduProcessingRef` and `CanIfTxPduProcessingRef` (added to the containers `CanIfRxProcessing` and `CanIfTxProcessing`)reference the PDUs that shall be processed on the MainFunction. When the queue is full, the incomming events will be ignored.

- MetaData support

  Description:

  Dynamic Tx and Rx PDUs are supported, where either parts or the whole CanId resides in the MetaData memory of the PDU which is managed by EcuC. A PDU is considered to have Metadata if a `MetaDataItem` of type `CAN_ID_32` is configured in EcuC. (see EcuC_Design.pdf)

  During transmission of a PDU with Metadata: CanIf will call `EcuC_GetMetaDataCanId()` (see EcuC_Design.pdf) in order to retrieve the Metadata information. `CanIfTxPduCanIdMask` defines which bits of the CanId reside in `CanIfTxPduCanId` and which are provided via Metadata. The resulting CanId is calculated in the following way: (`CanIfTxPduCanId` & `CanIfTxPduCanIdMask`) | (Metadata & ~`CanIfTxPduCanIdMask`). For a PDU with Metadata, `CanIfTxPduCanId` can be omitted. This means that the whole CanId is expected to be provided via the Metadata information.

  During reception of a PDU with Metadata: CanIf will perform a filtering which shall be done by comparing the incoming CanId with the stored `CanIfRxPduCanId` after applying the `CanIfRxPduCanIdMask` to both IDs. This filtering is applied on the stored `CanIfRxPduCanId` using linear search to get sure that

the matching ID can be found if exist. If the filtering is successful, CanIf shall place the CanId in the `MetaDataItem` of type `CAN_ID_32` by calling `EcuC_SetMetaDataCanId()` (see EcuC_Design.pdf).

- Multiple CAN driver support with one driver

  Description:

  It is possible to enable `CanIfPublicMultipleDrvSupport` with only one CAN driver configured. In this case, all the functions in relation to the driver are build using the `vendorId` and `vendorApiInfix` from that CAN driver, just like it would with multiple drivers configured.

- Support for different AR revisions of Can MCAL modules

  Description:

  The config parameter `CanIfCanDriverCompatibility` was added to support compatibility with different AR revisions of Can MCAL modules.

- Support for Autosar HandleId concept for CDDs

  Description:

  The config parameter `CanIfUseCddHandleIds` was added to support Autosar concept of Cdd handle id usage. If `CanIfUseCddHandleIds` is enabled, for Pdus that have a Cdd as an upper layer, CanIf will use handle ids from the Cdd that references that Pdu (CDD/CddComStackContribution/CddComIfUpperLayerContribution/CddComIfUpperLayerRx(or Tx)Pdu/CddComIfHandleId). If `CanIfUseCddHandleIds` is disabled, handle ids will be manually set using CanIfTxPduSourcePduID for Tx Pdu or CanIfRxPduTargetPduID for Rx Pdus.

- Support for Custom Hook on Reception

  Description:

  The container `CanIfHookOnRxSupport` was added to support Custom Hook on Reception. If `CanIfHookOnReceptionSupport` is enabled, a custom function call configured by CanIfHookOnReceptionFunctionName is enabled, which provides the Can ID, the PDU information and Controller ID of a successful reception and all configuration parameters from container CanIfHookOnRxSupport are configurable. If `CanIfHookOnReceptionSupport` is disabled, reporting a successful reception to a CDD module is disabled.

- Support for Bus Mirroring

  Description:

  The container `CanIfMirroringSupport` was added to support Autosar Bus Mirroring concept. If `CanIfBusMirroringSupport` is enabled, frame mirroring support through the Mirror module is enabled and all configuration parameters from container CanIfMirroringSupport are configurable. If `CanIfBusMirroringSupport` is disabled, frame mirroring support through the Mirror module is disabled.

- Software Filtering optimization

Description:

The Software Filtering can be disabled and consequently the related code is disabled in case all HRHs have parameter "CanIfHrhSoftwareFilter" set to OFF and no Rx Pdu has parameter "CanIfRxPduCanIdMask" enabled, i.e. the case where there is no need for Software Filtering.

The vendor specific parameter `CanIfSoftwareFilteringSupport` was added to disable Software filtering when it is not needed.

- Truncation options for a transmitted PDU

  Description:

  Truncation for TX PDUs is available as described in the AUTOSAR specification of CAN Interface, version R20-11.

  In addition, an over-sized TX Pdu can be truncated either to the size of a CAN frame (8 bytes for CAN 2.0, 64 bytes for CAN FD) or to the size of the configured length in EcuC. The option can be toggled based on the vendor specific parameter `CanIfTxPduTruncateToFrame`.

- Queue measurement support for decoupled processing

  Description:

  The decoupled processing of Rx and Tx Pdus can be measured by configuring the vendor specific parameters `CanIfRxDecoupledMeasurementSupport` and `CanIfTxDecoupledMeasurementSupport`. CanIf will forward to CDD the size of the enqueued Pdus in context of the MainFunction through the vendor specific parameters `CanIfNumberOfEnqueuedRxPdusApiName` and `CanIfNumberOfEnqueuedtxPdusApiName`.

  In addition, another configurable call can be forwarded to the CDD module through the vendor specific parameters `CanIfNumberOfRxPdusExceedingQueueApiName` and `CanIfNumberOfTxPdusExceedingQueueApiName` in case no Pdu can be enqueued anymore.

3.3.2.4. Deviations

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

- No inter-module consistency checks

  Description:

  The CanIf module does not perform the inter-module version checks as specified in the CanIf SWS.

  Rationale:

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

Requirements:

CANIF021

- Hrh Range Container (`CanIfHrhRangeCfg`) is not used

Description:

The software receive range filter as configured by configuration parameter CanIfHrhRangeCfg is not supported.

Rationale:

Software filtering is done by mapping received CanIDs to configured Rx LPdus. Putting an extra software range filter in front does not add any functional benefit.

Requirements:

CANIF645, CANIF646, CANIF628_Conf, CANIF629_Conf, CANIF644_Conf, CANIF630_Conf

- No Debug and Trace support

Description:

CanIf is not instrumented for the usage with Debug and Trace.

Requirements:

CANIF565, CANIF566, CANIF567, CANIF568

- `CanIf_Init()` has no default configuration if NULL_PTR is passed

Description:

If a NULL_PTR is passed to function CanIf_Init() differs from the SWS:

- Module PbCfgM used: Post-build-time configuration pointer is requested from PbCfgM module.
- Module PbCfgM not used: Module initialization is aborted and CANIF_E_PARAM_POINTER is reported to Det_ReportErrorStatus() if CanIfDevErrorDetect is true.

Requirements:

CANIF301

- `CanIf_ChangeBaudrate` and `CanIf_CheckBaudrate` API functions are not supported (reference to product description: ASCPD-98)

Description:

`CanIf_ChangeBaudrate` and `CanIf_CheckBaudrate` API functions are not implemented. Instead this feature is implemented according to AUTOSAR R4.1 Rev 1. where a new function `CanIf_SetBaudrate` has been introduced while `CanIf_ChangeBaudrate` and `CanIf_CheckBaudrate` are set to deprecated.

Requirements:

CANIF775, CANIF786, CANIF778, CANIF779, CANIF780, CANIF776, CANIF787, CANIF782, CANIF783, CANIF784, CANIF785_Conf, CANIF294

- Only post-build configuration is supported

Description:

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

Requirements:

CANIF460, CANIF461, CANIF377

- Only binary software filter algorithm is supported

Description:

The CanIf module only supports algorithm BINARY of configuration parameter CanIfPrivateSoftwareFilterType which configures the software filtering algorithm.

Requirements:

CANIF619_Conf

- No TTCan support

Description:

The CanIf implementation does not support TTCan.

Requirements:

CANIF675_Conf

- Parameter `CanIfTxPduDlc` is not used

Description:

The parameter `CanIfInitCfg/CanIfTxPduCfg/CanIfTxPduDlc` is not used. Instead, `CanIf_Transmit()` uses the DLC value given by the caller.

Requirements:

CANIF594_Conf

- Tx buffer handling according to AUTOSAR R4.0 Rev 2 (reference to product description: ASCPD-97)

Description:

If transmit buffering is enabled via parameter `CanIfPublicCfg/CanIfPublicTxBuffering`, the CanIf provides one buffer for each Tx PDU independent on the type of the associated HTH. I.e. the CanIf provides buffers for Tx PDUs assigned for BasicCAN as well as FullCAN transmission. Configuration parameter CanIfBufferSize is not used.

If a dynamic Tx PDU shall be buffered and the buffer for that PDU is already in use, the old message will only be overwritten, if the CAN ID of the new PDU is still the same as the one of the buffered message. If the CAN ID was changed in the meantime, the new message will be discarded and the old one remains in the buffer.

Rationale:

Resorting of pending Tx messages in case of an overwritten buffer is not implemented. Therefore overwriting buffers with messages that have a different CAN ID is rejected to prevent priority inversion in the CanIf Tx buffers. Also compare the requirement CANIF282 from previous AUTOSAR releases (e.g. R3.1).

Requirements:

CANIF837, CANIF833_Conf, CANIF834_Conf

- CanIf upper layer XCP must be configured as CDD module

Description:

The standard upper layer XCP is not contained in the enumeration range of the configuration parameters `CanIfRxPduUserRxIndicationUL` and `CanIfTxPduUserTxConfirmationUL`.

If this upper layer shall be used, configure it as complex device drivers (CDD).

Requirements:

CANIF556, CANIF555

- Symbolic names for CanIfCtrlIds and CanIfTrcvIds do not follow the AUTOSAR naming scheme

Description:

Some of the implemented symbolic name macros do not follow the AUTOSAR R4.0 Rev 3 naming scheme. AUTOSAR only specifies the inclusion of the name of direct parents in the symbolic macros but it is getting violated for CanIfCtrlIds and CanIfTrcvIds. CanIfCtrlIds: "CanIfConf_[CanIfCtrlDrvCfg]_[Controllername]" CanIfTrcvIds: "CanIfConf_[CanIfTrcvDrvCfg]_[CanIfTrcvCfg]"

Rationale:

CanIfCtrlIds short names are only distinct within the context of the superior CanIfCtrlDrv. CanIfTrcvIds short names are only distinct within the context of the superior CanIfTrcvDrv. The generation of SymbolicName macros as specified within the ECU configuration specification could lead to multiple macro redefinitions.

- Some configuration parameters have a lower config variant than specified

Description:

The following configuration parameters are specified to implement config variant link-time but actually implement config variant pre-compile:

- CanIfDispatchUserCtrlBusOffName
- CanIfDispatchUserCtrlModeIndicationName
- CanIfDispatchUserCheckTrcvWakeFlagIndicationName
- CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
- CanIfDispatchUserClearTrcvWufFlagIndicationName
- CanIfDispatchUserClearTrcvWufFlagIndicationUL
- CanIfDispatchUserConfirmPnAvailabilityName
- CanIfDispatchUserConfirmPnAvailabilityUL
- CanIfDispatchUserTrcvModeIndicationName
- CanIfDispatchUserValidateWakeupEventName
- CanIfDispatchUserCtrlBusOffUL
- CanIfDispatchUserCtrlModeIndicationUL
- CanIfDispatchUserValidateWakeupEventUL

The following configuration parameters are specified to implement config variant post-build but actually implement config variant link-time:

- CanIfRxPduUserRxIndicationUL
- CanIfTxPduUserTxConfirmationUL

- When CanIfValidateWakeupOnStartedCtrlOnly is disabled, wakeup validation is processed in CANIF_CS_SLEEP, CANIF_CS_STOPPED, and CANIF_CS_STARTED.

Description:

According to AUTOSAR R4.0 Rev 3 and later revisions of the specification successful wakeup validation shall be stored only if the Can controller is in state CANIF_CS_STARTED. Instead the current implemen-

tation validates wakeup successfully if the following conditions apply when the configuration parameter CanIfValidateWakeupOnStartedCtrlOnly is disabled:

- a) The Can controller is in state CANIF_CS_SLEEP.
- b) The Can controller is in state CANIF_CS_STARTED or CANIF_CS_STOPPED and a wakeup has been previously detected.

Requirements:

CANIF286

- Transmit confirmations always forwarded to upper layer module

Description:

Transmit confirmations are always forwarded to the upper layer regardless of the PDU mode.

The CanIf SWS prohibits calling the transmit confirmation callback service of the upper layer module if the PDU mode is equal to CANIF_SET_OFFLINE or CANIF_SET_TX_OFFLINE.

Rationale:

This mechanism ensures that pending confirmations are not blocked. A confirmation may be pending, if a transmission of LPDU occured before PDU mode was set to CANIF_SET_OFFLINE or CANIF_SET_TX_OFFLINE. Confirmations for transmissions triggered in PDU mode CANIF_SET_OFFLINE or CANIF_SET_TX_OFFLINE cannot occur, because transmission is prohibited in this case. Unexpected confirmations which may be triggered because of an erroneous CAN driver (hardware) are not taken into account.

Requirements:

CANIF073, CANIF489

- API service `CanIf_ReadRxPduData` always accepted

Description:

API service `CanIf_ReadRxPduData` accepts a request even if the corresponding CCMSM is not equal to CANIF_CS_STARTED or the corresponding PDU channel mode is not equal to CANIF_SET_ONLINE or CANIF_SET_TX_ONLINE.

Rationale:

The AUTOSAR configuration schema allows multiple HRHs and therefore multiple CanIf controller assigned to a single Rx-LPdu. This would force to merge the status of multiple CCMSM and PDU channel modes and is not specified. In addition it would increase the configuration complexity drastically.

Requirements:

CANIF324

- Reception of L-PDU is not limited to a single CAN controller

  Description:

  CanIf allows the configuration of a Rx-PDU with an assignment to multiple HRHs with different underlying CAN controller. The assignment of a Tx-PDUs and its corresponding Tx-Buffer is limited to a single HTH only.

  Rationale:

  The limitation, as described in the requirement, would prevent the reception of PDUs assigned to HRHs of a different CAN controller.

  Requirements:

  CANIF046

- DET error `CANIF_E_STOPPED` not supported

  Description:

  CanIf does not support the DET error `CANIF_E_STOPPED`. In detail this means:

  - `CanIf_Transmit` does not report `CANIF_E_STOPPED` if invoked in CCMSM `CANIF_CS_STOPPED` .
  - `CanIf_Transmit` does not report `CANIF_E_STOPPED` if invoked in PDU channel mode `CANIF_OFFLINE` .

  Rationale:

  In case of a bus-off, CanIf switches the CCMSM to `CANIF_E_STOPPED` and informs CanSM about the bus-off. But informing the upper layers does not happen in an atomic fashion. This leads to a small time slot, where a transmission by the upper layers is a regular use-case. CanIf handles this scenario by rejecting the transmit request, but without reporting the DET error.

  Requirements:

  CANIF382, CANIF723

- CanIf calls `EcuM_ValidateWakeupEvent()` only if wake-up has been successfully validated

  Description:

  CanIf calls `EcuM_ValidateWakeupEvent()` within the context of `CanIf_CheckValidation()` only if wake-up has been successfully validated.

  Rationale:

EcuM performs an action triggered by `EcuM_ValidateWakeupEvent()` no matter of the value of the parameter `sources`. According to Bugzilla (https://www.autosar.org/bugzilla/show_bug.cgi?id=57883), CanIf must NOT call `<User_ValidateWakeupEvent>(sources)` if wakeup has not been validated.

Requirements:

CANIF681

- CanIf does not support the API service `CanIf_TriggerTransmit()`.

Description:

CanIf does not support the API service `CanIf_TriggerTransmit()` introduced by AUTOSAR 4.2.1 along with CAN-FD.

Requirements:

SWS_CANIF_00883

- Due to "https://www.autosar.org/bugzilla/show_bug.cgi?id=52225" and "https://www.autosar.org/bugzilla/show_bug.cgi?id=61651" RfCs, CanIf_PduModeType was introduced, according with AUTOSAR 4.2.2 specification.

Description:

Due to "https://www.autosar.org/bugzilla/show_bug.cgi?id=52225" and "https://www.autosar.org/bugzilla/show_bug.cgi?id=61651" RfCs, CanIf_PduGetModeType and CanIf_PduSetModeType shall no longer be used. CanIf_PduModeType was introduced and shall be used accordingly: -CANIF_SET_TX_OFFLINE and CANIF_SET_TX_OFFLINE are not used anymore -CANIF_SET_TX_ONLINE and CANIF_SET_RX_ONLINE are not used anymore -CANIF_ONLINE shall be used instead of CANIF_SET_ONLINE and CANIF_GET_ONLINE -CANIF_TX_OFFLINE_ACTIVE shall be used instead of CANIF_SET_TX_OFFLINE_ACTIVE, CANIF_GET_OFFLINE_ACTIVE -CANIF_TX_OFFLINE shall be used instead of CANIF_SET_TX_OFFLINE -CANIF_GET_RX_ONLINE and CANIF_GET_TX_ONLINE are not used anymore -CANIF_OFFLINE shall be used instead of CANIF_GET_OFFLINE -CANIF_GET_OFFLINE_ACTIVE_RX_ONLINE is not used anymore PnTxFilter will be enabled if CanIf_SetControllerMode(controllerId, CANIF_CS_CLEEP) will be called. CanIf_PduModeType was introduced, according with AUTOSAR 4.2.2 specification.

Requirements:

CANIF490, CANIF491, CANIF492, SWS_CANIF_00073, SWS_CANIF_00483

- CanIf does not include the header Mirror_Cbk.h.

Description:

According to the SWS of the Mirror module the file does not exist. All the interfaces are exposed through the Mirror.h header.

Requirements:

SWS_CANIF_00903

- The signature of the CanIf_GetControllerMode API depends on the configuration

Description:

With the introduction of the BusMirroring feature (CONC_634), respecting the signature change of the API according to the AUTOSAR 4.4.0 SWS became mandatory.

Requirements:

CANIF_229

- The signature of the Tx Confirmation callback <User_TxConfirmation> depends on the configuration

Description:

With the introduction of the requirement SWS_CANIF_00739 for returning the result within the Tx Confirmation, respecting the signature change of the callback according to the AUTOSAR R20-11 SWS became mandatory.

Requirements:

CANIF011

- The signature of the CanIf_GetTrcvMode API depends on the configuration

Description:

With the introduction of the BusMirroring feature (CONC_634), respecting the signature change of the API according to the AUTOSAR 4.4.0 SWS became mandatory.

Requirements:

CANIF_288 SWS_CANIF_00288

- Wakeup event referenced in CanIfCtrlWakeupSourceOutRef shall be validated by CanIf with reception indications from that Controller

Description:

CanIfCtrlWakeupSourceOutRef can be configured even if that particular controller does not support wake-up and can be used for validation (e.g. the transceiver supports wake-up but the controller does not). The validation will be done on a reception on that controller.

Rationale:

Even though starting with CanIf ASR 4.0.3 CanIfCtrlWakeupSourceOutRef was not used anymore, it will be used to allow transceiver wake-up events to be validated through a controller that does not support wake-up.

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.

- No support of TTCan (reference to product description: ASCPD-3)

  Description:

  TTCan controllers are not supported by this implementation of CanIf.
- Number of supported CAN controllers

  Description:

  The CanIf only supports configurations with CAN controllers that have a controller ID less than 255.

  Rationale:

  The number 255 is used by the CanIf to classify a controller ID as invalid.
- Number of supported CAN Transceiver Drivers

  Description:

  Up to 255 CAN Transceiver Drivers are supported.

  Rationale:

  This restriction allows to use smaller types (uint8 instead of uint16) for internal calculations.
- No support for multiple configurations

  Description:

  Only one configuration is supported. Multiple configurations are not allowed.
- Supported software filtering mechanisms

  Description:

  Binary search is the only supported software filtering method.
- Only one upper layer callback function is supported (reference to product description: ASCPD-2)

  Description:

Only one RX indication and one TX confirmation callback function is supported for each upper layer.

- Code optimizations may result in compiler warnings

  Description:

  The `CanIf` implicitly uses code optimizations for certain configurations. This may, however, lead to compiler warnings such as "if-statement always evaluates to true".

  Rationale:

  Excluding all potentially redundant control flow statements would clutter code.

- Reconfiguration of CAN identifier limited for Flexible Data-Rate

  Description:

  The reconfiguration of a CAN identifier via the API service `CanIf_SetDynamicTxId` is restricted to a separate range for CAN 2.0 and CAN Flexible Data-Rate messages.

  A Tx-PDU with a CAN 2.0 identifier (`STANDARD_CAN` or `EXTENDED_CAN`) can not be reconfigured to a CAN Flexible Data-Rate identifier (`STANDARD_FD_CAN` or `EXTENDED_FD_CAN`) and vice versa.

  Rationale:

  The `CanIf` reserves a transmit buffer of 8 bytes for CAN 2.0 messages. A reconfiguration to a CAN Flexible Data-Rate message offers the possibility to send up to 64 bytes without having an underlying buffer of sufficient size.

- ASR versions of CAN drivers with Multiple CAN driver support enabled

  Description:

  When Multiple CAN driver support is enabled, the configured CAN drivers must have the same AUTOSAR SW release version and must be one which is supported by the CanIf module.

  Rationale:

  The behavior in relationship to the CAN drivers is treated in an unitary manner.

- J1939 PDUs mapping to different cores not supported

  Description:

  All J1939 PDUs must be mapped on the same core (the same one on which the J1939 stack is located) if the CAN stack is multicore distributed along network boundaries.

- Multicore not supported when Bus Mirroring enabled

  Description:

These features cannot be used together since the Multicore support for the Mirror module needs to be extended/adapted.

3.3.2.6. Open-source software

CanIf does not use open-source software.

3.3.3. CanNm module release notes

- AUTOSAR R4.0 Rev 3
- AUTOSAR SWS document version: 3.3.0
- Module version: 6.19.8.B466224
- Supplier: Elektrobit Automotive GmbH

3.3.3.1. Change log

This chapter lists the changes between different versions.

Module version 6.19.8
2021-10-27

- Improved support for calling CanNm APIs that rely on the current state from the context of the state change notification

Module version 6.19.7
2021-06-25

- ASCCANNM-1236 Fixed known issue: CanNm might unexpectedly not transmit NM messages after (re)entering Normal Operation State or Repeat Message State.

Module version 6.19.6
2021-03-05

- Added support for postbuild selectable config of CanNmMsgCycleOffset

Module version 6.19.5
2020-10-23

- Improved Active wakeup Bit functionality

Module version 6.19.4
2020-06-19

- ASCCANNM-1194 Fixed known issue: First NM message sent on the bus carries outdated state change information and additional messages are sent on the same main function.

Module version 6.19.3
2020-02-21

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

Module version 6.19.2
2019-10-11

- ASCCANNM-1142 Fixed known issue: Compilation error for CanNm module if CanNmStateChangeIndEnabled is set to true
- Changed maximum value for CanNmPnInfoOffset to 31 and minimum value for CanNmPnInfoLength to 1
- ASCCANNM-1160 Fixed known issue: Linker errors are reported due to incorrect memory mapping
- ASCCANNM-1159 Fixed known issue: Active wake-up bit in CBV is incorrectly set when Network Mode is re-entered.

Module version 6.19.1
2019-06-14

- Support transition from PrepareBusSleepMode to NetworkMode when CanNm_PassiveStartUp is triggered

Module version 6.19.0
2019-02-15

- Changed usage of message cycle offset
- Improved robustness check for references, optional parameters property and enable parameters property

- ASCCANNM-1107 Fixed known issue: First NM message sent on the bus carries outdated state change information
- ASCCANNM-1116 Fixed known issue: State change notification can be sent repeatedly from RepeatMessage state
- Changed structure of generated templates for post-build selectable support

Module version 6.18.0
2018-10-26

- Implemented Multi-core support
- ASCCANNM-1096 Fixed known issue: CanNm generates an invalid basic software module description if no configuration set is provided

Module version 6.17.3
2018-06-22

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

Module version 6.17.2
2018-02-16

- Implemented Post-build selectable support
- Removed AUTOSAR 3.x compliant symbolic name value macros and updated the logic to only provide AUTOSAR 4.0.2 compliant macros

Module version 6.17.1
2017-09-22

- Implemented support for car wake up
- Changed number of calls for Nm_StateChangeNotification during Prepare Bus Sleep Mode to Repeat Message State. Reverted ASCCANNM-923 Wrong state change information sent on CanBus on transition from Prepare Bus Sleep Mode to Repeat Message State
- ASCCANNM-943 Fixed known issue: Compilation error occurs if all CanNmPnFilterMaskByteValues are set to zero
- Added CanNmNodeDetectionEnabled, CanNmNodeIdEnabled and CanNmRepeatMsgIndEnabled as per channel configurable.
- ASCCANNM-944 Fixed known issue: Out of bounds access in case at post-build more PNCs are configured than at precompile time

- Implemented Support for PDU Length greater than 8 Bytes
- ASCCANNM-959 Fixed known issue: Source code does not compile if CanNmPnEiraCalcEnabled is set to false and for at least one channel CanNmPnEraCalcEnabled is set to true
- ASCCANNM-972 Fixed known issue: Code generation error for user data with length zero
- ASCCANNM-973 Fixed known issue: API CanNm_CheckRemoteSleepIndication indicates wrong state if the API interrupts the execution of the main function
- ASCCANNM-961-973 Fixed known issue: Existence of PDU referenced by CanNmPnEraRxNSduRef in PduR is not checked
- ASCCANNM-960 Fixed known issue: CanNmPnEiraRxNSduRef shall be available based on parameter CanNmPnEiraCalcEnabled
- ASCCANNM-963 Fixed known issue: Tx timeout exception is generated for a channel which works correctly

Module version 6.17.0
2017-03-31

- ASCCANNM-900 Fixed known issue: CanNm causes the ECU to become asynchronous with other ECUs on the network
- Added/corrected missing memory sections and compiler abstraction
- ASCCANNM-885 Fixed known issue: Incorrect consistency check of CanNmComUserDataSupport against CanNmUserDataEnabled
- ASCCANNM-881 Fixed known issue: The user data transmitted in the NM PDU could be inconsistent
- Removed CanNmRepeatMessageTime - CanNmMsgCycleTime multiplicity constrain
- Changed UserTxConfPduId member in the CanNm_ChannelDataType structure
- ASCCANNM-915 Fixed known issue: Wrong depedency CanNmStateChangeIndEnabled - CanNmPassiveModeEnabled. Improved CanNmStateChangeIndEnabled parameter description.
- ASCCANNM-923 Fixed known issue: Wrong state change information sent on CanBus on transition from Prepare Bus Sleep Mode to Repeat Message State

Module version 6.16.0
2016-10-31

- Added aggregation of internal and external requested partial networks also if the NM-PDU filter algorithm is disabled
- Implement CanNm_Transmit(...) for sponaneous transmission of NM frames
- ASCCANNM-870 Fixed known issue: Tx timeout timer called incorrectly

Module version 6.15.0
2016-05-23

- ASCCANNM-871 Fixed known issue: Repeat Message time is not editable if Passive mode is enabled

Module version 6.14.0
2016-02-10

- ASCCANNM-842 Fixed known issue: Handle ID wizard error computing CanNmRxPduIds when multiple CanNmRxPdu per channel are configured
- Added support for Debug & Trace with custom header file configurable via parameter `BaseDbgHeaderFile`

Module version 6.13.0
2015-11-06

- Fixed In case RMS timer is zero parameter NmRetryFirstMessageRequest should not be active
- Fixed incorrect handling of RSI timer in case of disabled communication
- Implemented NmRetryFirstMessageRequest functionality to Normal operation state if Repeat Message Time is zero
- Fixed the NM-Timeout and ReducedTime timer restart if Communication is disabled

Module version 6.12.0
2015-07-28

- ASCCANNM-807 Fixed known issue: Generation error when CanNmNodeIdEnabled = TRUE and CanNmPassiveModeEnabled = TRUE
- ASCCANNM-818 Fixed known issue: Changed implementation class of CanNmNodeId to PostBuild
- ASCCANNM-780 Fixed known issue: RMS time can be 1 cycle shorter if Immediate transmission is set to true
- ASCCANNM-812 Fixed known issue: Unintended trigger of immediate messages when CanNmImmediateNmTransmissions > 0 but CanNmImmediateNmCycleTime is disabled.
- ASCCANNM-798 Fixed known issue: In case of passive mode CanNmUserDataTxPdu should not be forced to exist

Module version 6.11.0
2015-06-24

- ASCCANNM-771 Fixed known issue: The Nm Message Tx Timeout Timer is not started when entering Network Mode from Prepare Bus Sleep Mode
- ASCCANNM-770 Fixed known issue: The Nm_RemoteSleepIndication() can be triggered by a transition from Ready Sleep to Normal Operation
- ASCCANNM-751 Fixed known issue: TX Pdu is still transmitted after API CanNm_DisableCommunication is called
- ASCCANNM-765 Fixed known issue: CanNm_CheckRemoteSleepIndication has an incorrect return value

Module version 6.10.0
2015-02-20

- ASCCANNM-776 Fixed known issue: Statemachine remains in RepeatMessageState if CanNmRepeatMessageTime is 0
- CanNm_Checks.m misleading error message

Module version 6.9.0
2015-01-07

- Fixed MISRA warning
- Update multiplicity of CanNmRxPdu elements according to resultion in bugzilla 54555.
- Add the CanNmNodeIdEnabled parameter.

Module version 6.8.0
2014-10-02

- Removed dependency that if CanNmRemoteSleepIndEnabled is false, then CanNmRemoteSleepIndTime needs to be 0
- ASCCANNM-686 Fixed known issue: Incomplete initialization of EIRA timer array
- ASCCANNM-674 Fixed known issue: NM messages are lost in case of external wake-ups
- Loosen the dependency between parameters `CanNmRemoteSleepIndTime` and `CanNmMsgCycleTime`
- Implemented support for Side Allocation
- Refactor CanNm.h to CanNm_Api.h
- Add a test to prove that the module compiles without any post build information
- ASCCANNM-744 Fixed known issue: CanNmRemoteSleepIndTime cannot be 0 even if CanNmRemoteSleepIndEnabled is false
- Implement support for the aggregation of external requested partial networks (ERA)

Module version 6.7.0
2014-04-25

- ASCCANNM-648 Fixed known issue: Wrong memory section for an array of type `NetworkHandleType`
- ASCCANNM-663 Fixed known issue: `PreCompile` parameters referencing `Postbuild` parameters in invalid conditions in CanNm's XDM schema
- ASCCANNM-670 Fixed known issue: ECU may be prevented from entering SLEEP mode due to incorrect handling of EIRA
- ASCCANNM-675 Fixed known issue: Compile error in `CanNm_HandleTimerTick` function if Debug and Trace module is enabled
- ASCCANNM-668 Fixed known issue: In case of active wake-ups the last immediate and first cyclic message shall not be transmitted in the same cycle if `CanNmMsgCycleOffset` is configured to 0

Module version 6.6.0
2013-12-11

- ASCCANNM-624 Fixed known issue: TX timeout exception is not reported if `CanIf_Transmit()` returns `E_NOT_OK`
- ASCCANNM-623 Fixed known issue: User data is initialized incorrectly to 0x00U instead of 0xFFU if partial networking is enabled
- ASCCANNM-608 Fixed known issue: An incorrect DET error is reported when an invalid PDU ID is passed to `CanNm_TxConfirmation()` or `CanNm_RxIndication()`
- Implemented support of `VARIANT-POST-BUILD` for CanNm

Module version 6.5.0
2013-10-18

- ASCCANNM-587 Fixed known issue: NM Messages are stopped after the transmission of immediate NM messages
- ASCCANNM-586 Fixed known issue: Unexpected behavior when `CanNmPduCbvPosition` is configured to `CANNM_PDU_OFF`
- Implemented support of spontaneous transmission of NM PDUs via calls of API function `CanNm_NetworkRequest()`
- Changed functionality to release a network even if transmission of NM messages is disabled
- Implemented support for notification of transmission timeouts to CanSM
- ASCCANNM-609 Fixed known issue: CanNm erroneously reports TX timeout exception if feature *Bus Load Reduction* is enabled

- Changed the default value of parameter `CanNmRemoteSleepIndEnabled` to false
- Added support for function tracing via AUTOSAR Debugging

Module version 6.4.0
2013-06-25

- Improved the robustness of the finite state machine design by revising the event handling; removed configuration parameter `CanNmEventQueueSize`
- ASCCANNM-548 Fixed known issue: A compiler errors when `CanNmNodeDetectionEnabled` is set to false and '`CanNmPduNidPosition` is not set to `CANNM_PDU_OFF`
- ASCCANNM-439 Fixed known issue: CanNm does not switch to repeat message state if communication is disabled

Module version 6.3.0
2013-02-15

- ASCCANNM-504 Fixed known issue: Error occurs during code generation when `CanNmTxPd` is disabled and `CanNmPassiveMode` is set to false
- Changed the reference path of `ComMChannel` in parameter `CanNmComMNetworkHandleRef` to `/AUTOSAR/EcucDefs/ComM/ComMConfigSet/ComMChannel`
- Memory allocation keywords were implemented in compliance to ASR 4.0.3
- ASCCANNM-474 Fixed known issue: The API functions `CanNm_GetNodeIdentifier()` /`CanNm_GetLocalNodeIdentifier()` are also available when parameter `CanNmPduNidPosition` is set to off
- ASCCANNM-507 Fixed known issue: Compiler errors when symbolic names according to AR4.0.3 are used
- ASCCANNM-512 Fixed known issue: `PduR_CanNmTxConfirmation()` and `PduR_CanNmTriggerTransmit()` are called with the wrong handle ID

Module version 6.2.0
2012-10-12

- ASCCANNM-463 Fixed known issue: Immediate Nm messages are not sent when `CanNmComControlEnabled` is set to false
- ASCCANNM-464 Fixed known issue: Incorrect number of immediate Nm messages
- ASCCANNM-461 Fixed known issue: Extra NM message is sent while leaving Repeat Message State when `CanNmMsgCycleOffset` is zero
- ASCCANNM-465 Fixed known issue: Compiler warning `statement not reached`

- ASCCANNM-467 Fixed known issue: CanNm schema does not prevent an invalid configuration of `CanNmRepeatMessageTime`
- Migrated to ASR 4.0 ComStack HandleId Policy
- Support for `ActiveWakeUp` bit in CBV added
- The top-level structure of the software-component description in the ARXML files changed from `/AUTOSAR/CanNm` to `/AUTOSAR_CanNm`
- ASCCANNM-486 Fixed known issue: Error during the PduR code generation when Com User Data support is enabled

Module version 6.1.1
2012-06-27

- Update to AUTOSAR 4.0.3 version
- ASCCANNM-413 Fixed known issue: No configuration constant for `CanNm_Init` is available
- ASCCANNM-422 Fixed known issue: CanNm may ignore valid PN messages
- ASCCANNM-397 Fixed known issue: Transition from Ready Sleep State to Prepare Bus-Sleep Mode takes longer than `CanNmTimeoutTime`
- ASCCANNM-425 Fixed known issue: EIRA contains PN requests which are not relevant for the ECU
- Corrected the invalid MemMap usage in CanNm_HsmCanNmFnct.c

Module version 6.1.0
2012-03-15

- ASCCANNM-364 Fixed known issue: Node never goes to Sleep state if `CanNm_DisableCommunication` is called in Repeat message state
- ASCCANNM-339 Fixed known issue: Event queue overflow
- ASCCANNM-385 Fixed known issue: Production error `CANNM_E_NETWORK_TIMEOUT` is erroneously reported

Module version 6.0.2
2011-12-08

- COM User Data Support added

Module version 6.0.1
2011-09-28

- EBACANNM-219 Fixed known issue: Bus Load Reduction Mechanism might increase the bus load

Module version 6.0.0
2011-09-02

- Initial AUTOSAR 4.0 version

3.3.3.2. New features

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

3.3.3.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- COM Rx user data

  Description:

  COM Support for Rx user data is added.

  - New container `CanNmUserDataRxPdu` is added to configure the Rx Pdu of received user data.
  - User can enable or disable this container.

  When this feature is enabled, then user must configure the respective Pdu in EcuC and provide correct routing path in PduR. When this feature is disabled, the user can still receive data using `CanNm_GetUserData()` API.

  Rationale:

  User has freedom of receiving the user data over COM.

- Channel with no user data with user data support enabled

  Description:

  As per AUTOSAR requirement CANNM086, when `CanNmUserDataEnabled` is enabled, the `CanNmUserDataLength` should not be zero.

  The module deviates from this requirement. The module allows a user to configure a mixture of channels where some channels support user data and some channels doesn't support user data.

  If CanNm_GetUserData or CanNm_SetUserData API is called for a channel with user data length as 0, DET error CANNM_E_INVALID_FUNCTION_ARG will be registered.

  Rationale:

More flexibility and freedom of configuration for user is achieved.

- Support for Side Allocation

  Description:

  The Side Allocation feature allow flashing of two different ECUs with the same software. The behaviour of each ECU will differ at runtime based on a flag(eg: stored in EEPROM or the level of a pin).

  The following parameter differ between the two variants: `CanNmNodeId` CanNm supports configurating a callout function to be called everytime the CanNm module needs to retrieve a NodeId for an ECU.

- Extended the range for the offset of the PN request information in the NM message

  Description:

  Starting from AUTOSAR requirement CANNM060_Conf, CanNmPnInfoOffset range has changed from 1..7 to 1..31.

  Rationale:

  More flexibility and freedom of configuration for user is achieved.

3.3.3.4. Deviations

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

- Initialization check in main function

  Description:

  According to requirements CANNM235 and CANNM236, if the main function (`CanNm_MainFunction()`) is called before the module is initialized (`CanNm_Init()`), the main function shall report an error to DET with an instance ID of `CanNm_MainFunction()`. In contrast to these requirements, `CanNm_MainFunction()` does not report any error to DET if it is called before initialization. However it returns immediately without performing any functionality.

  Rationale:

  According to AUTOSAR 4.0 General SRS requirement BSW00450, if a main function of an un-initialized module is called from the BSW Scheduler, it shall return immediately without performing any functionality and without raising any errors.

  Requirements:

  CANNM235, CANNM236

- COM user data zero length

Description:

As per the requirement CANNM086, if `CANNM_USER_DATA_ENABLED` is enabled, `CANNM_USER_DATA_LENGTH` should not be zero. In contrast to this, the user is allowed to configure a channel with `CANNM_USER_DATA_LENGTH` as zero.

Rationale:

In reality we may have a use case, where in a `CanNm` channel user data is present and another channel is not having user data. This requirement puts restriction on such use case.

Requirements:

CANNM086

- Coordinator Synchronization Support

Description:

The requirement CANNM341 describes that if the CanNm receives a NM message with the `NmCoordinatorSleepReady` bit (see CBV) set it shall indicate this to the Nm by calling `Nm_CoordReadyToSleepIndication()`.

The requirement CANNM342 describes that the `NmCoordinatorSleepReady` bit in the CBV shall be set by the API `CanNm_SetSleepReadyBit()`.

In contrast to the above requirements, `Nm_CoordReadyToSleepIndication()` is never called and the API `CanNm_SetSleepReadyBit()` is not provided.

Requirements:

CANNM341, CANNM342, CANNM343, CANNM338, CANNM340, CANNM080_Conf

- Dependency between `CanNmTimeoutTime` and `CanNmMsgCycleTime`

Description:

The requirement CANNM020_Conf states that the configuration parameter `CanNmTimeoutTime` must be a multiple of the value of the configuration parameter `CanNmMsgCycleTime`.

This dependency shall be removed since this dependency does not exist and therefore is wrong.

Rationale:

The system requirements of at least one customer requires timing settings where `CanNmTimeoutTime` is not a multiple of `CanNmMsgCycleTime`.

Additional information:

http://www.autosar.org/bugzilla/show_bug.cgi?id=54115

Requirements:

CANNM020_Conf

- Changes in symbolic name references

Description:

If the attribute `SHORT-NAME` is not specified for the container `CanNmRxPdu` the symbolic name macros for `CanNmRxPduId` are generated not according to the requirement ecuc_sws_2108 but according to the naming pattern `CanNmConf_CanNmChannelConfig_<ChannelName>_CanNmRxPdu`. where <ChannelName> is the name of the channel containing `CanNmRxPdu`.

The above behavior is also applicable for the generation of symbolic name for the parameter `CanNmTxConfirmationPduId` which is located inside the container `CanNmTxPdu` and `CanNmTxUserDataPduId` which is located inside the container `CanNmUserDataTxPdu` whose symbolic name macros are generated in the following pattern: `CanNmConf_CanNmChannelConfig_<ChannelName>_CanNmTxPdu` and `CanNmConf_CanNmChannelConfig_<ChannelName>_CanNmUserDataTxPdu`.

Rationale:

If no short-name is specified, EB tresos Studio assumes the name of the corresponding schema node as a default. Thus, the symbolic name macros generated according to the requirement ecuc_sws_2108 are not unique.

- Dependency of `CanNmNodeId` on `CanNmNodeDetectionEnabled`

Description:

The description of the parameter `CanNmNodeId` contains the following dependency: "This parameter is only valid if `CanNmPassiveModeEnabled = false` and `CanNmNodeDetectionEnabled = true`.

In contrast to this, the parameter `CanNmNodeId` can be configured with no dependency on `CanNmNodeDetectionEnabled`.

Rationale:

The dependency that `CanNmNodeId` is valid only if `CanNmNodeDetectionEnabled = true` is incorrect.

Additional information:

http://www.autosar.org/bugzilla/show_bug.cgi?id=57577

Requirements:

CANNM031_Conf

- Tracing of variables is not supported via AUTOSAR Debugging

  Description:

  CanNm module does not provide support for tracing global variables.

  Requirements:

  CANNM287, CANNM288, CANNM289, CANNM290

- Allow release of network if transmission of Nm messages is disabled

  Description:

  The function `CanNm_NetworkRelease()` returns `E_OK` and releases the network also if transmission of NM messages has been disabled by calling the function `CanNm_DisableCommunication()`.

  Rationale:

  According to requirement CANNM294 the function `CanNm_NetworkRelease()` shall have no effect and return `E_NOT_OK` if transmission of Nm messages is disabled. This behavior may cause problems when the network should be released. In case a call to `CanNm_NetworkRelease()` does not release the network the respective communication channel might not shut down and keep the ECU awake.

  Additional information:

  http://www.autosar.org/bugzilla/show_bug.cgi?id=60589

  Requirements:

  CANNM294

- No support for link-time configuration parameters

  Description:

  The following parameters are treated as pre-compile time parameters instead of as link-time parameters:

  - `CanNmGlobalConfig/CanNmMainFunctionPeriod`
  - `CanNmGlobalConfig/CanNmPnResetTime`
  - `CanNmGlobalConfig/CanNmPnEiraRxNSduRef`
  - `CanNmGlobalConfig/CanNmChannelConfig/CanNmAllNmMessagesKeepAwake`
  - `CanNmGlobalConfig/CanNmChannelConfig/CanNmBusLoadReductionActive`
  - `CanNmGlobalConfig/CanNmChannelConfig/CanNmCarWakeUpBitPosition`
  - `CanNmGlobalConfig/CanNmChannelConfig/CanNmCarWakeUpBytePosition`
  - `CanNmGlobalConfig/CanNmChannelConfig/CanNmCarWakeUpFilterEnabled`

- CanNmGlobalConfig/CanNmChannelConfig/CanNmCarWakeUpFilterNodeId
- CanNmGlobalConfig/CanNmChannelConfig/CanNmCarWakeUpRxEnabled
- CanNmGlobalConfig/CanNmChannelConfig/CanNmImmediateNmCycleTime
- CanNmGlobalConfig/CanNmChannelConfig/CanNmImmediateNmTransmissions
- CanNmGlobalConfig/CanNmChannelConfig/CanNmMsgCycleTime
- CanNmGlobalConfig/CanNmChannelConfig/CanNmMsgReducedTime
- CanNmGlobalConfig/CanNmChannelConfig/CanNmMsgTimeoutTime
- CanNmGlobalConfig/CanNmChannelConfig/CanNmNodeId
- CanNmGlobalConfig/CanNmChannelConfig/CanNmPduCbvPosition
- CanNmGlobalConfig/CanNmChannelConfig/CanNmPduNidPosition
- CanNmGlobalConfig/CanNmChannelConfig/CanNmPnEraCalcEnabled
- CanNmGlobalConfig/CanNmChannelConfig/CanNmPnHandleMultipleNetworkRequests
- CanNmGlobalConfig/CanNmChannelConfig/CanNmRemoteSleepIndTime
- CanNmGlobalConfig/CanNmChannelConfig/CanNmRepeatMessageTime
- CanNmGlobalConfig/CanNmChannelConfig/CanNmTimeoutTime
- CanNmGlobalConfig/CanNmChannelConfig/CanNmUserDataLength
- CanNmGlobalConfig/CanNmChannelConfig/CanNmWaitBusSleepTime
- CanNmGlobalConfig/CanNmChannelConfig/CanNmPnEraRxNSduRef
- CanNmGlobalConfig/CanNmChannelConfig/CanNmComMNetworkHandleRef
- CanNmGlobalConfig/CanNmChannelConfig/CanNmRxPdu/CanNmRxPduId
- CanNmGlobalConfig/CanNmChannelConfig/CanNmTxPdu/CanNmTxConfirmationPduId
- CanNmGlobalConfig/CanNmChannelConfig/CanNmUserDataTxPdu/CanNmTxUserDataPduId
- CanNmGlobalConfig/CanNmPnInfo/CanNmPnInfoLength
- CanNmGlobalConfig/CanNmPnInfo/CanNmPnInfoOffset
- CanNmGlobalConfig/CanNmPnInfo/CanNmPnFilterMaskByte/CanNmPnFilterMaskByteIndex

Requirements:

CANNM300

- Description:

The container CanNmRxPdu according to AUTOSAR R4.0.3 must have 1 element, but according to AUTOSAR R4.1.3 the multiplicity for the container CanNmRxPdu is 1..*

This deviation has been introduced due to bugzilla ticket #54555.

Requirements:

CANNM038_Conf

- Changes in symbolic name references

  Description:

  If the attribute `SHORT-NAME` is specified for the container `CanNmRxPdu` the symbolic name macros for `CanNmRxPduId` are generated not according to the requirement ecuc_sws_2108 but according to the naming pattern `CanNmConf_CanNmChannelConfig_<ChannelName>_SHORT-NAME`. where <ChannelName> is the name of the channel containing `CanNmRxPdu`.

- Changes regarding CanNmImmediateNmCycleTime

  Description:

  Parameter `CanNmImmediateNmCycleTime` has a default value of 0.001 and the multiplicity is 1 and not 0..1 as in the Autosar 4.0.3 SWS.

  Requirements:

  CANNM057_Conf

- Changes regarding CanNmMsgCycleOffset

  Description:

  Parameter `CanNmMsgCycleOffset` is treated as post-build selectable parameter instead of as link-time parameter.

  Requirements:

  CANNM300, CANNM029_Conf

3.3.3.5. 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.

- For this module no limitations are known.

3.3.3.6. Open-source software

CanNm does not use open-source software.

3.3.4. CanSM module release notes

- AUTOSAR R4.0 Rev 3
- AUTOSAR SWS document version: 2.2.0
- Module version: 3.7.7.B466224
- Supplier: Elektrobit Automotive GmbH

3.3.4.1. Change log

This chapter lists the changes between different versions.

Module version 3.7.7
2021-10-08

- Support distribution along network boundaries.

Module version 3.7.6
2021-09-17

- Make configuration variant editable.

Module version 3.7.5
2021-06-25

- ASCCANSM-610 Fixed known issue: CanSM can treat all CAN network channels as having a transceiver and cause a deadlock

Module version 3.7.4
2021-05-28

- Internal module improvement. This module version update does not affect module functionality.
- ASCCANSM-607 Fixed known issue: CanSM is missing in the Create ECU Configuration wizard

Module version 3.7.3
2021-04-30

- Internal module improvement. This module version update does not affect module functionality.
- ASCCANSM-602 Fixed known issue: ModeRequestRepetitionTime could underflow if configured on zero

Module version 3.7.2
2021-04-09

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

Module version 3.7.1
2021-03-05

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

Module version 3.7.0
2021-02-12

- Added post build selectable support.

Module version 3.6.25
2020-10-23

- ASCCANSM-580 Fixed known issue: Remove limitation for single driver support

Module version 3.6.24
2020-08-28

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

Module version 3.6.23
2020-07-31

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

Module version 3.6.22
2020-06-19

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

Module version 3.6.21
2020-02-21

- ASCCANSM-564 Fixed known issue: BusOff recovery switches from L1 to L2 faster than expected

Module version 3.6.20
2019-12-06

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

Module version 3.6.19
2019-08-09

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

Module version 3.6.18
2019-06-14

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

Module version 3.6.17
2019-05-17

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

Module version 3.6.16
2019-04-18

- ASCCANSM-554 Fixed known issue: Polyspace test - CanSM_Cfg.c could cause out of index access on array.

Module version 3.6.15
2018-11-23

- ASCCANSM-551 Fixed known issue: Async server calls for bus indication are only generated for single channel.

Module version 3.6.14
2018-10-26

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

Module version 3.6.13
2018-09-28

- Added multicore support.

Module version 3.6.12
2018-06-22

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

Module version 3.6.11
2018-05-25

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

Module version 3.6.10
2018-04-20

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

Module version 3.6.9
2018-02-16

- ASCCANSM-502 Fixed known issue: Missing include in source files when TS_MERGED_COMPILE is disabled

Module version 3.6.8
2017-11-17

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

Module version 3.6.7
2017-09-22

- ASCCANSM-481 Fixed known issue: Call CanIf_SetPduMode() with CANIF_ONLINE after CanSM_TxTimeoutException()

Module version 3.6.6
2017-08-25

- Inverted logic for AUTOSAR 4.0.2 and remove AUTOSAR 3.x legacy support for symbolic names

Module version 3.6.5
2017-07-28

- Support of specific Bus-Off handling.

Module version 3.6.4
2017-06-02

- ASCCANSM-469 Fixed known issue: Wrong error message and inconsistency in check of providing legacy defines

Module version 3.6.3
2017-03-31

- ASCCANSM-467 Fixed known issue: Memory mapping is unbalanced if merged compilation is used and partial networking enabled

Module version 3.6.2
2017-03-10

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

Module version 3.6.1
2017-03-03

- Move integration requirements to separate reqm file.

Module version 3.6.0
2017-02-03

- Internal module improvement. This module version update does not affect module functionality
- Do not enable partial networking TX filter after BusOff

Module version 3.5.1
2017-01-05

- ASCCANSM-425 Fixed known issue: Changed the way ServiceNeedsWizard configures CanSM DemEvents

Module version 3.5.0
2016-11-04

- Usage of partial networking functionality on transceivers without selective wakeup capability

Module version 3.4.6
2016-05-25

- ASCCANSM-429 Fixed known issue: CanSM aborts compilation if validation of memory section enabled

Module version 3.4.5
2016-02-05

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

Module version 3.4.4
2015-11-06

- Source code restructuring in order to improve maintainability.

Module version 3.4.3
2015-09-18

- ASCCANSM-408 Fixed known issue: Bus-off recovery results in switch to OFFLINE if multiple Can controllers are connected to a single Can network

Module version 3.4.2
2015-06-18

- ASCCANSM-399 Fixed known issue: CanSM calls CanNm_ConfirmPnAvailability() with a wrong value for parameter nmChannelHandle

Module version 3.4.1
2014-12-19

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

Module version 3.4.0
2014-10-02

- ASCCANSM-366 Fixed known issue: CanSM does not handle subsequent bus-off events
- Updated state machine to harmonize with a ComM module according to AUTOSAR SWS 4.1 rev3
- Updated Tx timeout exception behavior according to AUTOSAR SWS 4.1 rev3
- Added possibility to enable partial networking without a transceiver configuration

Module version 3.3.4
2014-04-25

- ASCCANSM-356 Fixed known issue: CanSM includes DEM interface header file even if Dem module is not used
- Added support for enhanced BswM bus-off reporting

Module version 3.3.3
2013-11-15

- ASCCANSM-348 Fixed known issue: CanSM makes unattended calls to `BswM_CanSM_CurrentState()` and `CanIf_SetPduMode()`
- Added support for new baudrate switching API function `CanSM_SetBaudrate()`

Module version 3.3.2
2013-10-11

- Implemented support for handling of transmission timeouts

Module version 3.3.1
2013-06-14

- Added support for EB Debug and Trace solution
- Implemented non-functional code improvements in order to meet mass production criteria

Module version 3.3.0
2013-02-08

- ASCCANSM-275 Fixed known issue: Wrong header file included for usage of API function `CanNm_ConfirmPnAvailability()`
- ASCCANSM-284 Fixed known issue: CanSM uses wrong symbolic name for DEM event referenced by configuration parameter `CANSM_E_BUS_OFF`
- ASCCANSM-301 Fixed known issue: Compiler errors occur when partial networking is enabled but no transceiver is configured
- Updated reference paths of CanSM-ComMChannel reference for the introduction of `ComMConfigSet` container in ComM
- Updated state machine according to AUTOSAR 4.0 Rev 3 SWS

Module version 3.2.0
2012-10-19

- ASCCANSM-248 Fixed known issue: The CanSM prevents the shutdown of the ECU in case the `ComMNmVariant` of the corresponding `ComMChannel` is configured to NONE or LIGHT
- Changed the top-level structure of the software-component description in the ARXML files from `/AUTOSAR/CanSM` to `/AUTOSAR_CanSM`
- Implemented Dem handling according to AUTOSAR 4.0 Rev 3 SWS

Module version 3.1.1
2012-06-20

- ASCCANSM-221 Fixed known issue: Wrong header file included for usage of API function `CanNm_ConfirmPnAvailability()`

Module version 3.1.0
2012-03-16

- Partial networking support

Module version 3.0.0
2011-09-02

- Initial AUTOSAR 4.0 version.

3.3.4.2. New features

- Added support for distribution of CAN stack along network boudaries.

3.3.4.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- Enhanced production error reporting

  Description:

  An enhanced production error reporting mechanism has been introduced. This allows to configure the following options independently for each Dem event:

  - Report production errors to the `Diagnostics Event Manager` (`Dem`).
  - Report production errors to the `Development Error Tracer` (`Det`) as development errors.
  - Do not report production errors at all.

  If a production error is redirected towards the `Det`, you may configure the reported `Det` error-ID.

  Rationale:

  This enhancement implements the HIS requirements concerning fault operation and error detection: HisGen0007, HisGen0008 and HisGen0009.

- DET error `CANSM_E_INVALID_BUSOFF_INDICATION` introduced

  Description:

  If the function `CanSM_ControllerBusOff()` is indicated and `CanSM` is not in state `COMM_FULL_COMMUNICATION`, `CanSM_ControllerBusOff()` calls the function `Det_ReportError()` with the ErrorId parameter `CANSM_E_INVALID_BUSOFF_INDICATION (value 0x0B)`.

  Rationale:

  Bus-off recovery is only applicable if `CanSM` is in full communication.

- DET error `CANSM_E_TXTIMEOUT_RECOVERY_ACTIVE` introduced

  Description:

  If the function `CanSM_RequestComMode()` is indicated and `CanSM` perform Tx timeout recovery (refer to function `CanSM_TxTimeoutException()`), `CanSM_RequestComMode()` calls the function `Det_ReportError()` with the ErrorId parameter `CANSM_E_TXTIMEOUT_RECOVERY_ACTIVE (value 0x0C)`.

  Rationale:

  Mode transition is only applicable if no Tx timeout recovery is active.

- Enhanced bus-off reporting

  Description:

  Refer to the EB specific configuration parameter `CanSMEnhancedBusOffReporting` for enabling enhanced bus-off reporting. If enhanced bus-off reporting is enabled, the `CanSM` reports the following two additional states to the `BswM`:

  - `CANSM_BSWM_BUS_OFF_L1`: To report a bus-off when the bus-off counter is lower than `CanSMBorCounterL1ToL2`.
  - `CANSM_BSWM_BUS_OFF_L2`: To report a bus-off when the bus-off counter is greater than or equal to `CanSMBorCounterL1ToL2`.

  Rationale:

  This enhancement allows the `BswM` and its extensions to distinguish between the bus-off recovery levels L1 and L2 when a bus-off is notified.

- Partial networking usage

  Description:

  `CanSM` allows to use the feature `partial networking` without any CAN transceiver driver configuration.

  Rationale:

  This enhancement gives `CanNm` the possibility to enable its `partial networking` support without using an AUTOSAR conform CAN Transceiver driver module.

  This might be a valid use-case for an ECU initiating wakeups of partial networks without needing the capability to wakeup on its own.

- Partial networking Tx filter not enabled after Bus-Off

  Description:

  `CanSM` allows to use the feature `partial networking` without delaying the startup through TxPnFilter (needed to handle first the NM messages) after a Bus-Off.

  Rationale:

  This enhancement implements the PDU mode state machine according to ASR 4.2.2. Please refer to Bugzilla RFC 52225 and 61651 for more details.

- Support of specific Bus-Off handling

  Description:

`CanSM` supports a specific bus-off handling which is achieved by a pre-compile time config parameter named `CanSMBusDeactivatedBussOff`. This new feature will deactivate the CAN controller, during the bus-off recovery time.

Rationale:

After a Bus-Off, `CanSM` module shall - deactivate Tx and also Rx path, setting Pdu modes to CANIF_-OFFLINE, instead of CANIF_TX_OFFLINE(AUTOSAR-like bahavior) - start CAN controller

- Support of Multicore handling

  Description:

  `CanSM` supports a multicore handling which is achieved by a pre-compile time config parameter named `CanSMMultiCoreSupport`.

  Rationale:

  In a multicore system, `CanSM` module shall - be able to send notification for mode change on a different core.

- Support of Multicore Distribution along network boundaries

  Description:

  `CanSM` supports a multicore distribution and should be able to be allocated and work on different cores/ partitions which is achieved by a pre-compile time config parameter named `CanSMDistributedChan-nelProcessingSupport`.

  `CanSMDistributedChannelProcessingSupport` is not enabled, until `CanSMMultiCoreSupport` is configured to true.

  A CanSM_MainFunction_-"SN"() shall be created (where "SN" is the shortName of the EcucPartition) for each EcucPartition referenced for ComMChannelPartitionRef, and this function will be responsible for processing each CAN network that reference this EcucPartition.

3.3.4.4. Deviations

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

- On bus-off event, the order of actions differs from the AUTOSAR SWS

  Description:

  In case of a bus-off event, AUTOSAR specifies the following actions to be performed:

  1. `Call BswM_CanSM_CurrentState(network,CANSM_BSWM_BUS_OFF)` [CANSM508]

2. `Call ComM_BusSM_ModeIndication(network, COMM_SILENT_COMMUNICATION)` [CANSM521]
3. `Call Dem_ReportErrorStatus(eventId, DEM_EVENT_STATUS_PREFAILED)` [CANSM522]
4. `Call CanIf_SetControllerMode(ctrlId, CANIF_CS_STARTED)` [CANSM509]

This implementations performs the required action in the following order:

1. `Call BswM_CanSM_CurrentState(network,CANSM_BSWM_BUS_OFF)`
2. `Call Dem_ReportErrorStatus(eventId, DEM_EVENT_STATUS_PREFAILED)`
3. `Call CanIf_SetControllerMode(ctrlId, CANIF_CS_STARTED)`
4. `Call ComM_BusSM_ModeIndication(network, COMM_SILENT_COMMUNICATION)`

Rationale:

Action order follows the AUTOSAR 4.2 Rev 2 SWS.

Requirements:

CANSM508 CANSM521 CANSM522 CANSM509

- Only post-build configuration is supported

Description:

The `CanSM` module only supports configuration variant `VARIANT-POST-BUILD`. `VARIANT-PRE-COMPILE` (CANSM250) and `VARIANT-LINK-TIME` (CANSM251) are not supported. The source files `CanSM_Lcfg.c` (CANSM361) is not provided.

Requirements:

CANSM250 CANSM251 CANSM361 CANSM010

- `CanSM` does not provide APIs to change or check baud rate of a CAN network (reference to product description: ASCPD-98)

Description:

`CanSM` does not support the following API services:

- `CanSM_CheckBaudrate`
- `CanSM_ChangeBaudrate`

Requirements:

CANSM501 CANSM564 CANSM565 CANSM562 CANSM571 CANSM563 CANSM566 CANSM561 CANSM569 CANSM570 CANSM502 CANSM504 CANSM505 CANSM530 CANSM506 CANSM573 CANSM574 CANSM503 CANSM567 CANSM568 CANSM572 CANSM432 CANSM433 CANSM524

CANSM525 CANSM526 CANSM527 CANSM529 CANSM531 CANSM532 CANSM533 CANSM534 CANSM535 CANSM536 CANSM542 CANSM543 CANSM342_Conf CANSM507

- No consistency check between code files and header files

Description:

The inter-module version checks as specified in the CanSM SWS are not implemented.

Rationale:

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

Requirements:

CANSM025

- Communication mode may change in context of `CanSM_RequestComMode`

Description:

Transitions of the `CanSM` state machine, which are triggered by the API function `CanSM_RequestComMode`, are fully operated in the context of `CanSM_RequestComMode` if expected mode indications are performed synchronously.

Rationale:

Reduction of time necessary for a state transition. In case of synchronous mode indications the transition is also synchronous, instead of lasting until the next `CanSM` MainFunction invocation.

Requirements:

CANSM428

- No AUTOSAR debug and trace support

Description:

`CanSM` does not support AUTOSAR debug and trace support.

Requirements:

CANSM310 CANSM309

- `CanSM` does not call `ComM` during initial transition

Description:

`CanSM` does not call `ComM` notification `ComM_BusSM_ModeIndication()` during the initial transition to No Communication.

Rationale:

The initial transition takes place in the initialization of CanSM. ComM may not be initialized at this moment.

Requirements:

CANSM430

- CanSM always accepts a mode request

Description:

The CanSM always accepts a mode request by ComM and reports a Development Error only, if a mode request is done with invalid parameters.

Rationale:

CanSM must not lose a mode request by ComM. A mode request done during an ongoing transition shall be stored and processed afterwards. Please refer to Bugzilla Rfc 55033 for more details.

Requirements:

CANSM375 CANSM376 CANSM377 CANSM395 CANSM555 CANSM402

- Partial networking TX filter is not enabled after BusOff

Description:

The TX_OFFLINE transition as a E_FULL_TO_SILENT_COM effect is now done directly without passing through ONLINE first. The effect E_TX_OFF of the sub state machine CANSM_BSM_S_FULLCOM is not needed anymore (no PDU mode request to CANIF_TX_OFFLINE needed). CanIf sets the CANIF_TX_-OFFLINE mode when CanIf_ControllerBusOff() is called.

Rationale:

The PDU mode state machine is updated to ASR 4.2.2 and now the transition to OFFLINE is done automatically by CanIf. CanIf_PduSetModeType and CanIf_PduGetModeType were replaced by CanIf_PduModeType containing the following PDU Channel Modes: CANIF_ONLINE, CANIF_OF-FLINE, CANIF_TX_OFFLINE and CANIF_TX_OFFLINE_ACTIVE. Please refer to Bugzilla RFC 52225 and 61651 for more details.

Requirements:

CANSM537 CANSM513

- Inclusion of CanSM.h

Description:

CanSM.c will no longer include CanSM.h

Rationale:

Since 4.1 the requirement has ceased to exist, due to changes of inclusion in BswM.

Requirements:

CANSM013

3.3.4.5. 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.

- Code optimizations may result in compiler warnings

  Description:

  The `CanSM` implicitly uses code optimizations for certain configurations. This may, however, lead to compiler warnings such as "if-statement always evaluates to true".

  Rationale:

  Excluding all potentially redundant control flow statements would clutter code.

3.3.4.6. Open-source software

CanSM does not use open-source software.

3.3.5. CanTp module release notes

- AUTOSAR R4.0 Rev 3
- AUTOSAR SWS document version: 4.0.0
- Module version: 6.8.45.B466224
- Supplier: Elektrobit Automotive GmbH

3.3.5.1. Change log

This chapter lists the changes between different versions.

Module version 6.8.45
2021-10-08

- Memory section symbol "CANTP_START_SEC_JUMP_TABLE_SHARED_VAR_INIT_UNSPECIFIED" deviates from recommended notation.
- Reconcile memory and main function declarations with respect to Rte.

Module version 6.8.44
2021-09-17

- Distribution of CAN stack along network boundaries
- ASCCANTP-1419 Fixed known issue: Unexpected counter decrease if MainFunction is interrupted by Tx Confirmation.

Module version 6.8.43
2021-07-28

- ASCCANTP-1412 Fixed known issue: Unreachable code assert occurs when an N-PDU with MetaData and invalid N_PCI type is received

Module version 6.8.42
2021-06-25

- ASCCANTP-1397 Fixed known issue: Transmission is aborted when N_Cs is smaller than STmin

Module version 6.8.41
2021-05-28

- Added metadata support for Normal Fixed addressing mode.
- ASCCANTP-1398 Fixed known issue: CanTp rejects flow control with additional bytes when padding is off

Module version 6.8.40
2021-04-30

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

Module version 6.8.39
2021-04-09

- Make the postpone/stallhandling feature of received messages optional.

Module version 6.8.38
2021-03-05

- Enable FC and Data Pdu of one channel to use a single CanIf Tx Pdu.
- ASCCANTP-1376 Fixed known issue: Multi-frame transmission aborted when the Flow Control received previously to the First Frame confirmation.

Module version 6.8.37
2021-01-22

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

Module version 6.8.36
2020-12-18

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

Module version 6.8.35
2020-09-25

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

Module version 6.8.34
2020-08-28

- ASCCANTP-1365 Fixed known issue: Received padded single frames with wrong length can stop ongoing connections.

Module version 6.8.33
2020-07-31

- Tresos:Automatically calculation of settings of parameters

Module version 6.8.32
2020-04-24

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

Module version 6.8.31
2020-03-25

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

Module version 6.8.30
2020-02-21

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

Module version 6.8.29
2019-12-06

- ASCCANTP-1337 Fixed known issue: CAN FD flow control is not accepted when normal padding is active.

Module version 6.8.28
2019-11-08

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

Module version 6.8.27
2019-10-11

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

Module version 6.8.26
2019-09-06

- ASCCANTP-1327 Fixed known issue: STmin not equal to ‘127ms’ when reserved value is used and CanTpChangeTxParameterApi is enabled.

Module version 6.8.25
2019-08-09

- ASCCANTP-1321 Fixed known issue: Wrong channel index type is used.

Module version 6.8.24
2019-07-12

- ASCCANTP-1319 Fixed known issue: Incorrect data type used for channel Id, in case CANTP_MODE_-FULL_DUPLEX is used.

Module version 6.8.23
2019-06-14

- ASCCANTP-1309 Fixed known issue: Incorrect handling of a single frame PDU might cause unexpected behavior of the ECU.

Module version 6.8.22
2019-04-18

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

Module version 6.8.21
2019-03-22

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

Module version 6.8.20
2019-02-15

- ASCCANTP-1298 Fixed known issue: Generation error occurs in CanTp if N:1 routing is used in PduR.

Module version 6.8.19
2019-01-25

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

Module version 6.8.18
2018-12-13

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

Module version 6.8.17
2018-10-26

- Added check for NULL_PTR in CanTp_ReadParameter().
- ASCCANTP-1286 Fixed known issue: Wrong Tx locked channel used.

Module version 6.8.16
2018-09-28

- ASCCANTP-1263 Fixed known issue: Receive message might be lost in case of pending TxConfirmation on the same channel .
- ASCCANTP-1277 Fixed known issue: Invalid CanTp_Bswmd.arxml generated when no CanTpConfig configured.

Module version 6.8.15
2018-08-24

- Added support for CanTp_ChangeTxParameter() for TX N-SDUs.

Module version 6.8.14
2018-07-27

- Improving CanTp_MainFunction Performance.

Module version 6.8.13
2018-06-22

- Allow TxConfirmation() to request another transmission of the same PDU.
- ASCCANTP-1228 Fixed known issue: The dynamic STmin value change during segmented transmission is not allowed.

- ASCCANTP-1253 Fixed known issue: Invalid N_PCI of Single Frame with FD accepted.

Module version 6.8.12
2018-05-25

- ASCCANTP-1209 Fixed known issue: Wrong padding value for small CAN-FD frames.

Module version 6.8.11
2018-04-20

- Add support for uint32 PduLengthType

Module version 6.8.10
2018-03-16

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

Module version 6.8.9
2018-02-16

- ASCCANTP-1206 Fixed known issue: CAN-FD Flow Control length erroneously computed for CAN-FD messages.

Module version 6.8.8
2018-01-19

- ASCCANTP-1174 Fixed known issue: Configuration of CAN 2.0 frames with length less than 8 bytes is not allowed.

Module version 6.8.7
2017-12-15

- Reduce memory consumption by introducing parallel channels.

Module version 6.8.6
2017-09-22

- Improve usage of critical section for idle channels.

Module version 6.8.5
2017-08-25

- ASCCANTP-1157 Fixed known issue: Usage of mixed CAN 2.0 and CAN-FD PDUs not allowed.

Module version 6.8.4
2017-07-28

- ASCCANTP-1150 Fixed known issue: Incorrect compiler abstraction used in CanTp_RxIndication.
- Post-build selectable support.
- ASCCANTP-1148 Fixed known issue: Wrong data type is used for the struture member NumberOfChannels.

Module version 6.8.3
2017-06-30

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

Module version 6.8.2
2017-06-02

- Provide CanTp_ReadParameter() API.

Module version 6.8.1
2017-05-05

- Report Det error when a postponed Rx frame is overwritten.

Module version 6.8.0
2017-03-31

- The number of CanTpRx/TxChannels as well as CanTpRx/TxNSdus that are supported by the implementation is extended.

Module version 6.7.1
2017-03-10

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

Module version 6.7.0
2017-03-03

- Configurable CAN-FD PDUs padding length to 64 bytes.
- Move integration requirements to separate reqm file.

Module version 6.6.1
2017-02-03

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

Module version 6.6.0
2017-01-05

- Different padding byte values for CAN 2.0 and CAN FD PDUs.
- Different configurable timeout values for repeated FC WAIT and other FC PDUs

Module version 6.5.12
2016-12-02

- ASCCANTP-1078 Fixed known issue: Behavior upon reception of unexpected PDUs deviates from ISO/CD 15765-2:2014 and AUTOSAR 4.1.x/4.2.x

Module version 6.5.11
2016-11-04

- Adapted resource file for the scheduling of main functions to the split of `IpduM_MainFunction()` into `IpduM_MainFunctionRx()` and `IpduM_MainFunctionTx()`.
- Improve the description of CanTpNSa and CanTpNTa
- Remove compiler warnings with GHS multi C Compiler v2014.1.6

Module version 6.5.10
2016-07-01

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

Module version 6.5.9
2016-04-29

- ASCCANTP-1059 Fixed known issue: CanTp sends flow control frame with status overflow as response on discarded single frame

Module version 6.5.8
2016-02-05

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

Module version 6.5.7
2015-11-06

- Removed the usage EcuC PduLength as maximum for I-PDUs
- ASCCANTP-1045 Fixed known issue: If there are multiple CanTpRxNPduId or CanTpRxFcNPduId instances configured to value 0, CanTp_SetNSa() fails
- Updated memory section naming

Module version 6.5.6
2015-06-19

- Added support to transmit and receive segmented messages with more than 4095 bytes
- Added CAN FD support to transmit and receive N-PDUs with length up to 64 bytes
- ASCCANTP-1037 Fixed known issue: CanTp_SetNSa()/CanTp_GetNSa() APIs write and read source addresses for wrong N-SDUs

Module version 6.5.5
2015-02-20

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

Module version 6.5.4
2015-01-07

- Added support for configurable mapping of CanTp_IsValidConfig function to dedicate memory section
- Removed AUTOSAR 3.x compliant symbolic name value macros and updated the logic to only provide AUTOSAR 4.0.2 compliant macros if macro CANTP_PROVIDE_LEGACY_SYMBOLIC_NAMES is defined

Module version 6.5.3
2014-10-02

- Improved state machine to allow expected incoming frames (CTS, CF) before outgoing frames (CF, CTS) are confirmed
- Update range check of `CanTpGptChannelResolution` to prevent that it is configured to zero

Module version 6.5.2
2014-04-24

- Updated address space to allow incoming N-PDU and FC N-PDU with same address
- ASCCANTP-964 Fixed known issue: The service needs assistant tries to generate (non existent) CanTp Dem events and reports a warning that shall be ignored
- ASCCANTP-983 Fixed known issue: Compilation aborts and reports an error if memory mapping is used for memory sections `CANTP_START_CONFIG_DATA_UNSPECIFIED` and `CANTP_START_SEC_CODE`
- Introduced memory section for jump table shared variables
- ASCCANTP-985 Fixed known issue: Compilation aborts if Dbg function call tracing is enabled for internal CanTp function `CanTp_RequestTxFrameData()`
- ASCCANTP-986 Fixed known issue: Build error due to missing file CanTp_PBcfg.c if code generation for CanTp is disabled and only post-build configuration is compiled
- Updated block size value for segmented frame reception

Module version 6.5.1
2013-10-10

- ASCCANTP-913 Fixed known issue: CanTp expects the upper layer to provide data sufficient to fill a CF which may lead to a N_Cr timeout

- Updated symbolic name value naming schema according to AUTOSAR 4.0 rev3
- Updated MCG to generate XML code for Binary Code Generation

Module version 6.5.0
2013-06-18

- ASCCANTP-804 Fixed known issue: The functions `CanTp_CancelReceive()` and `CanTp_CancelTransmit()` incorrectly report a DET error
- ASCCANTP-836 Fixed known issue: `CanTp_Transmit` does not check the N-Sdu data size for functional addressing properly
- 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
- ASCCANTP-862 Fixed known issue: Post-build configuration does not work if jumptables are enabled
- Implemented `CanTpTc`
- Updated handle ID wizard to set the configuration parameters `CanTpRxNPduId` and `CanTpRxFcNPduId` also for extended and mixed addressing format

Module version 6.4.0
2013-02-08

- Add relocatability to post build configuration
- ASCCANTP-729 Fixed known issue: If `CanTpNcs` is equal to `CanTpMainFunctionPeriod`, a timeout always occurs
- ASCCANTP-598 Fixed known issue: If a timeout occurs, CanTp might report `NTFRSLT_E_NOT_OK` instead of the timeout specific error code
- Update block size calculation to AUTOSAR 4.0 rev3

Module version 6.3.0
2012-10-16

- Update BSW to AUTOSAR 4.0 rev3 TP API
- Migration to ASR 4.0 ComStack HandleId Policy
- ASCCANTP-738 Fixed known issue: Automatic assignment of `CanTp_MainFunction()` to a task for periodic execution works only if the `CanTpConfig` is named `CanTpConfig_0`

- The top-level structure of the software-component description in the ARXML files changed from `/AUTOSAR/CanTp` to `/AUTOSAR_CanTp`

Module version 6.2.0
2012-06-20

- Updated config according to AUTOSAR 4.0 rev3
- Introduce post build data structure

Module version 6.1.0
2012-03-16

- ASCCANTP-612 Fixed known issue: CanTp uses default N_Cs value for buffer handling timeouts if no physical Tx N-SDU is configured
- Updated naming scheme for #defines for symbolic name values to AUTOSAR 4.0 rev3 naming scheme

Module version 6.0.1
2011-09-30

- ASCCANTP-597 Fixed known issue: Compilation of CanTp generates a compiler warning or fails with an error due to the use of invalid preprocessor directive in the source code
- Updated Dem handling (except the configuration) to AUTOSAR 4.0

Module version 6.0.0
2011-09-02

- Initial AUTOSAR 4.0 version

3.3.5.2. New features

- Allow TxConfirmation() to request another transmission of the same PDU.
- Added support for CanTp_ChangeTxParameter() for TX N-SDUs.
- Added support for automatically calculated values/settings of parameters.
- Enable FC and Data Pdu of one channel to use a single CanIf Tx Pdu.
- Added metadata support for Normal Fixed addressing mode.
- Added support for distribution of CAN stack along network boudaries.

3.3.5.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- The parameter `CanTpGeneral/CanTpDynamicNSaEnabled` was added to configure the handling of N_SA values. It allows to configure the following handling of N_SA values:
  - TRUE: N_SA values can be set and get via API interface functions.
  - FALSE: Use of N_SA values as configured in ROM (default).
- The module can be used as jump table module from several applications.

  In this case, one application must be the jump table server, that implements the jump table and all functionality. The other applications can then configure the CanTp as jump table client which means that the functionality is reduced to wrapper functions or macros, that call the jump table server functions.
- The module can recover from lower layer transmit errors.

  If the call to `CanIf_Transmit()` fails, the module does not return an error immediately. Instead, it tries to transmit the frame until the N_As timeout has elapsed and then it notifies the upper layer upon failure.
- The module tolerates received padded frames even if padding is disabled.

  If the CanTp module of the sender is configured with padding enabled and the CanTp module of the receiver is configured with padding disabled, the CanTp module of the receiver tolerates received padded frames and processes them.
- The module is able to receive and transmit segmented frames with up to 65535 bytes of payload.

  The module is able to transmit and receive segmented frames to transport up to 65535 bytes of data. For a payload greater than 4095 the module will use the first frame format as specified in AUTOSAR 4.2.1.
- The module supports CAN FD to transmit and receive PDUs up to 64 byte.

  The module is also able to transmit and receive frames with PDU length of 12, 16, 20, 24, 32, 48, or 64. For single frames with PDU size greater 8 the module will use the single frame format as specified in AUTOSAR 4.2.1. This feature can be enabled by the configuration parameter `CanTpGeneral/CanTpFlexibleDataRateSupport`. The maximum allowed PDU size can be configured for every EcuC PDU which is referred by `CanTpRxNPduRef` or `CanTpTxNPduRef`.
- The CanTp module supports different configurable padding values for CAN-FD frames.

  The module is able to transmit different padding values for CAN-FD and CAN 2.0 PDUs. This feature can be enabled by the configuration parameter `CanTpGeneral/CanTpPaddingByteCanFD` allowing a maximum padding value up to 255.
- The module supports different configurable timeout values for repeated FC WAIT PDUs.

  The module is able to transmit Flow Control frames with WAIT status using a different timeout value for repeated Flow Control WAIT PDUs. This feature can be enabled by the configuration parameter `CanTpGen-`

eral/CanTpNbrWaitRepeatedSupport. The Flow Control PDU timeout can be configured for every RxNSdu via CanTpNbrWaitRepeated.

- The CanTp module supports mandatory padding of CAN FD PDUs to 64 bytes and CAN 2.0 PDUs to 8 bytes, if CANTP_ON_CAN_CAN_FD is configured.

  The module is able to transmit CAN FD PDUs with the maximum value of 64 bytes. This feature can be enabled by the configuration parameter CanTpTxPaddingActivation (on transmission) and CanTpRxPaddingActivation (on reception). The possible enumeration literals of the existing config parameters CanTpTxPaddingActivation and CanTpRxPaddingActivation are extended by the literal CANTP_ON_CAN_CAN_FD:

  - CANTP_OFF:
    - No padding needed for CAN 2.0 PDUs
    - Enable mandatory padding with variable lengths (8, 12, 16, 20, 24, 32, 48, 64 based on configured EcuC maximum length) for CAN FD PDUs
  - CANTP_ON:
    - Enable mandatory padding to 8 bytes for CAN 2.0 PDUs only
    - Enable mandatory padding with variable lengths (8, 12, 16, 20, 24, 32, 48, 64 based on configured EcuC maximum length) for CAN FD PDUs
  - CANTP_ON_CAN_CAN_FD: Enable mandatory padding to 8 bytes for CAN 2.0 PDUs and 64 bytes for CAN FD PDUs
- The module supports an enlarged upper limit of NSdus and channels.

  The CanTp module supports up to 65535 half duplex or 32767 full duplex connection channels. The maximum number of TxNSdus, as well as RxNSdus was extended to 32767.
- The module reports an error when a queued CF is overwritten.

  The module reports a Det error when a postpones Rx frame is overwritten.
- The module provides CanTp_ReadParameter() API.

  To get a high performance link between a tester and an ECU during the network, the TP has to be speed up by changing FlowControl parameters like STmin and Blocksize. After changing STmin and Blocksize parameters, CanTp module provides an interface to read the current values for STmin and Blocksize from the CAN-TP. The use case for reading the TP parameters is to have the possibility to check the values of the parameters after writing them.
- The module supports parallel channels.

  Parallel channels are an efficient and fast way to reduce RAM consumption. Information during run-time can be stored using parallel channels. When CanTpMaxParallelChannels are configured the major amount of required global RAM is given by the array CanTp_Channel which dimension is equal with the maximum number or parallel channels.

- CanTp_MainFunction better performance when all channels are idle.

  If all channels are Idle, CanTp_MainFunction shall exit immediately without executing any functionality or entering unwanted critical sections. That improves efficiently the execution time of CanTp_MainFunction API.

- The module provides CanTp_ChangeTxParameter() API.

  The module provides the CanTp_ChangeTxParameter() API support to change the transmit parameter STmin.

- The module provides CanTp_ResetTxParameter() API.

  The module provides the CanTp_ResetTxParameter() API support to reset the STmin parameter value.

- The module provides configuration parameter CANTP_CHANGE_TX_PARAMETER_REQ_API.

  This feature can be enabled by the configuration parameter CANTP_CHANGE_TX_PARAMETER_REQ_-API API .

- The module provides CanTp_ChangeRxParameter() and CanTp_ChangeParameter() APIs.

  This two APIs can be used at a time, but never together.

- The module provides support for MetaData and for Normal fixed addressing format.

  A PDU is considered to have Metadata if at least one MetaDataItem of type `SOURCE_ADDRESS_16` or `TARGET_ADDRESS_16` is configured in EcuC. (see EcuC_Design.pdf) Currently, CanTp supports Meta-Data only for `CANTP_NORMALFIXED` addressing format. Normal fixed addressing is a subformat of normal addressing in which the mapping of the address information is in the CAN identifier.

  During transmission of a N-SDU with Metadata (if `CanTpDynIdSupport` is enabled), CanTp will fill in all the bytes from the CanId according to ISO 15765-2:2016 as follows:

  - `EcuC_GetMetaDataSourceAddr()` API (see EcuC_Design.pdf) is called in order to get the source address information representing bits 0..7 of the Metadata information. This value must match the configured `CanTpNTa` value.

  - if `CanTpGenericConnectionSupport` is disabled, CanTp will use as target address information the configured value of `CanTpNTa` representing bits 8..15 of the Metadata information.

  - if `CanTpGenericConnectionSupport` is enabled, `EcuC_GetMetaDataTargetAddr()` API (see EcuC_Design.pdf) is called in order to retrieve the target address information representing bits 8..15 of the Metadata information.

  - bits 16..28 are filled with static values as defined by ISO 15765-2:2016 (bit 16 represents uni-/multicast message based on the `CanTpTxTaType`).

  - CanTp sets the CanId for CanIf through the `EcuC_SetMetaDataCanId()` API call.

CanTp will calculate the whole CanId when Normal fixed addressing is used and parameters `CanIfTxPduCanId` and `CanIfTxPduCanIdMask` can be omitted for the respective Pdus in CanIf.

During FC reception, CanTp will check that the address information (N_Sa and N_Ta) matches the address information from the FF transmission.

During reception of a PDU with Metadata (if `CanTpDynIdSupport` is enabled), the following steps will be performed:

- CanTp will use the CanId at the FF or SF reception by calling `EcuC_GetMetaDataCanId()` API to obtain the N_Ta and N_Sa (if `CanTpGenericConnectionSupport` is disabled). The N_Ta must match the configured value of `CanTpNSa` to be accepted.

- if `CanTpGenericConnectionSupport` is disabled, CanTp will use the configured source address value of `CanTpNSa` representing bits 0..7 of the Metadata information (the CanId) to call `EcuC_SetMetaDataSourceAddr()` API.

- if `CanTpGenericConnectionSupport` is enabled, `EcuC_SetMetaDataSourceAddr()` API (see EcuC_Design.pdf) is called in order to set the source address information representing bits 0..7 of the CanId.

- CanTp will call `EcuC_SetMetaDataTargetAddr()` API (see EcuC_Design.pdf) in order to set the target address information representing bits 8..15 of the Metadata information obtained from the CanId.

Before sending a FC, CanTp will call `EcuC_GetMetaDataSourceAddr()` and `EcuC_GetMetaDataTargetAddr()` APIs to get the N_Ta and N_Sa set through the previous frame reception (FF or CF).

- The module provides support for Multicore and for Dedicated Channels Processing.

- If `CanTpDedicatedChannelProcessingSupport` is enabled, CanTpChannelProcessing configuration will be enabled and a CanTp_MainFunction will be generated for each of its subcontainers.

- For each of those CanTp_MainFunctions the following naming scheme will be used: CanTp_MainFunction_`shortName`(): Where `shortName` is the short name of the respective configuration container in the ECU configuration.

- Each CanTp_MainFunction generated will handle one or several CanTp Channels that are referenced by respective CanTpChannelProcessing container.

- The CanTp Channels that are not referecend by any CanTpChannelProcessing will be handled by the normal CanTp_MainFunction.

- If `CanTpMultiCoreSupport` is enabled, CanTpEcucPartitionRef configuration will be enabled.

3.3.5.4. Deviations

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

- Flow control frames are sent immediately without respecting timeout N_Br.

  Description:

  During reception of a segmented message, ISO 15765-2 chapter 6.7.1 mandates to wait for N_Br to elapse before sending a flow control (FC) frame. For the CanTp implementation, the flow control messages FC(CTS) and FC(OVFLW) are sent immediately when the corresponding conditions (buffer available, buffer request failed permanently) are met. The flow control message FC(WT) is sent after N_Br has elapsed.

  Rationale:

  To improve bus performance, feedback is provided immediately when it is known. FC(WT) is only sent if needed.

- 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 module may schedule the modules main function before the module is initialized. This would result in lots of Det errors during start up. 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.

- CanTp does not report `CANTP_E_INVALID_TX_BUFFER` and `CANTP_E_INVALID_RX_BUFFER`.

  Description:

  CanTp does not provide any DET checks which reports the error `CANTP_E_INVALID_TX_BUFFER` or `CANTP_E_INVALID_RX_BUFFER`.

  Rationale:

  With the change of the AUTOSAR Tp API in AUTOSAR 4.0, the CanTp DET errors `CANTP_E_INVALID_TX_BUFFER` and `CANTP_E_INVALID_RX_BUFFER` are obsolete. See Bugzilla http://www.autosar.org/bugzilla/show_bug.cgi?id=56264.

  Requirements:

  CANTP293

- CanTp does not provide the API function `CanTp_Shutdown()` (reference to product description: ASCPD-96).

  Description:

  The API function `CanTp_Shutdown()` is not implemented in the CanTp module.

  Rationale:

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

  Requirements:

  CANTP010, CANTP211, CANTP202, CANTP200

- `PduR_CanTpChangeParameterConfirmation()` must not be called.

  Description:

  The callback function `PduR_CanTpChangeParameterConfirmation()` is not used to notify the upper layer about the result of the `CanTp_ChangeParameter()` function call.

  Rationale:

  Since `CanTp_ChangeParameter()` is specified as synchronous, there is no need to use the callback function `PduR_CanTpChangeParameterConfirmation()` to notify the upper layer. The return value is sufficient. Also see Bugzilla http://www.autosar.org/bugzilla/show_bug.cgi?id=46227.

  Requirements:

  CANTP304, CANTP305, CANTP306

- Notification result `NTFRSLT_E_CANCELATION_OK` and `NTFRSLT_E_CANCELATION_NOT_OK` not used

  Description:

  CanTp reports a successful cancellation with `PduR_CanTpRxIndication()`/`PduR_CanTpTxConfirmation()` using the notification result `NTFRSLT_E_NOT_OK`. In case that the cancellation was not successful, `PduR_CanTpRxIndication()`/`PduR_CanTpTxConfirmation()` is not called.

  Rationale:

  Due to the decision in the Bugzilla issue http://www.autosar.org/bugzilla/show_bug.cgi?id=52106 the notification result `NTFRSLT_E_CANCELATION_OK` and `NTFRSLT_E_CANCELATION_NOT_OK` shall not be used. Instead a successful cancellation shall be reported with `PduR_CanTpRxIndication()`/`PduR_CanTpTxConfirmation()` using the notification result NTFRSLT_E_NOT_OK.

Requirements:

CANTP244, CANTP255, CANTP263

- No AUTOSAR Debug and Trace support

Description:

CanTp is not instrumented for the usage with AUTOSAR Debug and Trace.

Requirements:

CANTP249, CANTP250, CANTP251, CANTP252, CANTP253

- `CanTpRxDl` and `CanTpTxDl` are not used

Description:

The configuration parameters `CanTpRxDl` and `CanTpTxDl` are not used.

Rationale:

Based on RFC53101 the `CanTpRxDl` and `CanTpTxDl` are deprecated and shall not be used in the future. http://www.autosar.org/bugzilla/show_bug.cgi?id=53101

Requirements:

CANTP280_Conf, CANTP267_Conf

- Det errors `CANTP_E_TX_COM`, `CANTP_E_RX_COM` and `CANTP_E_COM` are not reported

Description:

In case that a connection is aborted due to a timeout or other connection related issues, the module does not report a Det error.

Rationale:

These Det reports are no real development error information but additional runtime information in case that a connection problem occurs. However, in this case the upper layer is informed about the reason of the aborted connection anyway. The Det report does not provide any additional or relevant information if this happens. Find the discussion to this topic at http://www.autosar.org/bugzilla/show_bug.cgi?id=52569

Requirements:

CANTP229, CANTP293

- No consistency check between code files and header files

Description:

The inter-module version checks as specified by the SWS are not implemented.

Rationale:

- The required compile-time version checks would result in an inflexible basic software stack hardly to integrate.
- EB tresos AutoCore is an already integrated product.
- The project handling of EB tresos Studio provides means to enforce that only modules with the same AUTOSAR release version can be added to the project.

Requirements:

CANTP307, CANTP308

3.3.5.5. 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 on the number of connection channels

  Description:

  The CanTp supports up to 65535 half duplex or 32767 full duplex connection channels. If full and half duplex channels are mixed twice the number of full duplex channels plus the number of half duplex channels must be lower than or equal to 65535.

  Rationale:

  This limitation allows to use 2 byte to identify channels and therefore reduces the ROM size of the configuration.
- Limitation on the number of N-SDUs

  Description:

  The maximum number of `CanTpRxNSdus` and `CanTpTxNSdus` are implementation dependent and limited to 32767 each.
- Limitation on parameter `CanTpRxWftMax`

  Description:

  The CanTp supports a maximum value of 255 for parameter `CanTpRxWftMax`.

  Rationale:

This limitation allows to use 1 byte for the `N_WFTmax` parameter and therefore reduces the ROM size of the configuration.

- Maximum data size of segmented frames is limited to 65535

  Description:

  CanTp only supports to transmit N-SDUs and receive I-PDUs which do not exceed 65535.

  Rationale:

  The `PduLengthTypeEnum` in `EcuC` is limited to 16 bit.

- The CanTp supports MetaData only for Normal fixed addressing mode

  Description:

  CanTp allows support for MetaData only for `CanTpRxAddressingFormat` and `CanTpTxAddressingFormat` set to `CanTpNormalFixed`. MetaData for `CanTpExtended`, `CanTpMixed` and `CanTpMixed29Bit` addressing format is currently not supported.

- The CanTp doesn't support Dedicated Channels Processing and Max Parallel Channels at the same time

  Description:

  CanTp doesn't support `CanTpDedicatedChannelProcessingSupport` enabled and `CanTpMaxParallelTxChannels` or `CanTpMaxParallelRxChannels`. enabled at the same time. An error will be displayed if both features are enabled.

- The CanTp Dedicated Channels Processing cannot be used implicitly with Postbuild support

  Description:

  `CanTpDedicatedChannelProcessingSupport` cannot be used implicitly with Postbuild support.

- The CanTp Dedicated Channels Processing Main Functions limitation

  Description:

  The maximum number of `CanTpChannelProcessing` is 255.

3.3.5.6. Open-source software

CanTp does not use open-source software.

4. ACG8 CAN Stack user guide

4.1. Overview

The ACG8 CAN Stack user guide describes the concepts of the CAN stack in the AUTOSAR context and provides configuration advice for the ACG8 CAN Stack modules. 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.

This user guide contains the following sections:

- Section 4.2, “Background information” describes the concepts of CAN communication in the AUTOSAR context.
- Section 4.3, “Configuring the ACG8 CAN Stack” provides information on how to configure the ACG8 CAN Stack.

This user guide is intended for readers who have good knowledge of AUTOSAR and about the purpose of the ACG8 CAN Stack modules.

4.2. Background information

This chapter provides general information about the CAN 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 CAN

The CAN communication stack provides a one-to-one mapping between I/N-PDUs and L-PDUs. This means each I/N-PDU is packed into exactly one CAN frame and each CAN frame carries exactly one I/N-PDU.

The transmission of an L-PDU in the CAN communication works as follows:

1. The `Com` module calls `PduR_ComTransmit()`.
2. The `PduR` calls `CanIf_Transmit()`.

3. The `CanIf` calls the `Can_Write()` function of the `Can` module.

This way, the timing of the `Com` schedule drives a periodic transmission of L-PDUs on the CAN bus.

The fact that the CAN bus receives an L-PDU causes the call of the reception indication. Whenever the `Can` module detects that is has received an L-PDU that is configured to be received by the ECU, the `Can` module calls the callback function `CanIf_RxIndication()` of the `CanIf` module.

In addition, the `Can` module provides support for the following events issued by the CAN controller:

- bus-off events
- wake-up events
- transmission confirmations
- transmission cancellation confirmations.

Each of these events can be serviced as follows:

- in the context of an interrupt service routine provided by the `Can` module, or
- via polling functions, which have to be scheduled cyclically. These are in particular the functions
  - `Can_MainFunction_Read()` for handling reception indications,
  - `Can_MainFunction_Write()` for handling transmission confirmations and transmission cancellation confirmations,
  - `Can_MainFunction_BusOff()` for handling bus-off indications,
  - and `Can_MainFunction_Wakeup()` for handling wake-up indications.

4.2.1.1. Module dependencies

For information on the dependencies of the CAN communication stack modules, see the EB tresos AutoCore Generic documentation.

In addition to the dependencies described in the EB tresos AutoCore Generic documentation, the `CanTp` module exhibits a dependency on the General Purpose Timer (`Gpt`) module. The `CanTp` uses a timer of the `Gpt` module for the measuring of the `CanTp` specific time-outs.

4.2.2. CAN FD

CAN FD addresses the demand of higher bandwidth without changing the technology. Compared to CAN 2.0, the feature CAN FD has the following advantages:

- CAN FD increases the payload of CAN frames from 8 up to 64 bytes.
- CAN FD increases the transmission rate of the data phase. The baud rate is increased for the payload and CRC of a CAN FD frame.

Each listed functionality increases the bandwidth on the CAN network.

For information on how to configure CAN FD, see Section 4.3.1, “Configuring CAN FD”.

4.2.3. CANStack MetaData

On transmission of a PDU with MetaData, `CanIf` retrieves the MetaData information set by the upper layer. In the case of `CanTp`, the whole CAN-ID is constructed by `CanTp`, including the source and destination address for normal fixed addressing. MetaData is needed to compose the CAN-ID for a PDU with MetaData.

Upon reception of a PDU with MetaData, `CanIf` places the CAN-ID in the `MetaDataItem` of type CAN_ID_32. Afterwards, `CanTp` gets the target address and source address from the CAN-ID and compares them with the configured addresses. If they do not match, reception is aborted.

`CanIf` supports the ability to filter incoming messages using the `CanIfRxPduCanIdMask`. The filtering is done by comparing the incoming CAN-ID with the configured `CanIfRxPduCanIdMask`.

For information on how to configure CAN MetaData, see Section 4.3.4, “Configuring CAN MetaData”.

4.2.4. (Prototype) Multicore distribution along network boundaries

The communication stack can consume a lot of CPU load and projects with heavy CAN communication usage should be able to take full advantage of a multi-core architecture.

In order to achieve this, the possibility to distribute the same CAN stack on different cores along network boundaries has been introduced. The feature encapsulates the following:

- Separation of Pdu processing
- Mechanism for state management handling in the context of multicore systems
- Reduce the amount of cross-core communication (and thus, potentially blocking synchronization)
- Ensure memory access and data consistency

For information on how to configure the multicore distribution along network boundaries, see Section 4.3.5, “(Prototype) Configuring the multicore distribution along network boundaries”.

4.3. Configuring the ACG8 CAN Stack

This section contains configuration instructions that involve several modules of ACG8 CAN Stack.

4.3.1. Configuring CAN FD

This section provides a starting point on how to configure the CAN FD feature. For background information, see Section 4.2.2, “CAN FD”.

Configuring CAN FD

Step 1
In `Can`, to use the CAN FD functionality, enable `CanFDSupport`.

Step 2
Add a `Can` controller reference to the configuration container `CanControllerBaudrateConfig` with a configured subcontainer `CanControllerFdBaudrateConfig`.

Step 3
In `CanIf`, define the frame type to be used in `CanIfRxPduCanIdType` and `CanIfTxPduCanIdType`.

In addition to CAN 2.0 frame types, you can also choose CAN FD.

Step 4
In `CanIfRxPduDlc`, set the lower limit for the received CAN FD frame payload. You can set the limit up to 64.

WARNING

Usage of padded CAN FD frames

A CAN FD frame shall match one of the specified FD lengths. If this is not the case, it is automatically padded by the CAN driver. `CanIf` does not perform padding and cannot detect padding bytes within a CAN FD frame. `CanIf` uses the length of the `EcuC` PDU that is referenced in the parameter `CanIfRxPduRef` for the maximum length. If the number of data bytes is static, you can use this configuration parameter to cut off padding bytes. Otherwise the application must take care to remove the padding bytes.

TIP

Padded CAN FD frames with `CanTp`

`CanTp` automatically performs padding of unaligned data. The module detects incoming padded CAN FD frames and only passes the data to the upper layer.

Step 5
In `CanTp`, in configuration parameter `CanTpFlexibleDataRateSupport`, enable the CAN FD functionality globally for `CanTp`.

NOTE **Usage of `CanTp` is optional**

The module `CanTp` is still optional. You can use CAN FD without `CanTp`.

Step 6

In `CanTpRxNPduRef` and `CanTpTxNPduRef`, reference the `EcuC` PDUs. The configured PDU lengths of the references define the maximum payload size of the CAN FD frames up to 64 bytes.

4.3.2. Configuring HOH assignment and bit timing in Can and CanIf

4.3.2.1. Overview

The CAN Buffer Assignment Editor and the CAN Bit Timing Editor are editors that are contained in the CAN Assistant. The CAN Assistant is an EB tresos Studio-based Eclipse plug-in for configuring the `Can` and `CanIf` modules in your configuration project.

Use the CAN Buffer Assignment Editor to configure the `Can` and `CanIf` modules in the area of PDU to hardware object handle (HOH) assignment. Use the CAN Bit Timing Editor to set up bit timing parameters in the `Can` module.

The CAN Buffer Assignment Editor and CAN Bit Timing Editor are referred to as CAN wizards in the remainder of this documentation.

Section 4.3.2.2, “Background information” gives you necessary background information about the CAN wizards and their user interfaces.

Section 4.3.2.3, “Using the CAN Buffer Assignment Editor” outlines how you can execute specific configuration steps for PDU to HOH assignment.

Section 4.3.2.4, “Using the CAN Bit Timing Editor” outlines how you can execute specific configuration steps for setting up the bit timing parameters.

4.3.2.2. Background information

The following sections describe basic concepts about the CAN wizards and their user interfaces:

- Section 4.3.2.2.1, “System and local configuration parameters” explains the *local* and *system* configuration parameters of the `Can` and `CanIf` configurations.
- Section 4.3.2.2.2, “Necessary parameters” outlines which kind of parameters the CAN wizards depend on.

- Section 4.3.2.2.3, “Resource files” explains how the CAN wizards are tailored to specific hardware platforms.
- Section 4.3.2.2.4, “The CAN Buffer Assignment Editor GUI” describes the GUI of the CAN Buffer Assignment Editor.
- Section 4.3.2.2.5, “Auto assignment” explains the auto assignment algorithm.
- Section 4.3.2.2.6, “The CAN Bit Timing Editor GUI” describes the GUI of the CAN Bit Timing Editor.

4.3.2.2.1. System and local configuration parameters

Configuration parameters of communication stack modules can be grouped into

- System configuration parameters and
- Local configuration parameters.

System configuration parameters must be configured consistently among all nodes of the same communication network for proper operation.

System configuration parameters for CAN networks are, for example:

- The CAN network bit rate
- The set of communication controllers that participate in the CAN network
- The set of PDUs that each communication controller sends and receives

System configuration parameters are usually imported into a configuration project. File types that can be imported into EB tresos Studio configuration projects include DBC, Fibex, or AUTOSAR system description files. For detailed information about these file formats, see the EB tresos Studio user's guide.

The CAN wizards configure `Can` and `CanIf` local configuration parameters that take `Can` and `CanIf` system and local configuration parameters as input.

TIP

Leave the importer option Buffer Assignment in Can/CanIf at the predefined value

When you configure the import, leave the option **Buffer Assignment in Can/CanIf** at the predefined value **Create default buffer assignment**. There is no direct link between PDUs and communication controllers in the `CanIf` configuration. This relationship is implicitly given by HOHs that define links to controllers and PDUs. If you choose **Do not create Buffer Assignment** for **Buffer Assignment in Can/CanIf**, the relationship between PDUs and controllers is lost because no HOHs are created with this setting.

Local configuration parameters are parameters that you can configure freely as long as they do not become inconsistent with the system configuration parameters. The properties of the shared communication network do not directly depend on local parameters.

The local configuration parameters include:

- The hardware object handles (HOHs) that are used to send and receive PDUs
- The point in time when the communication controller samples the value of a CAN bit, i.e. the sample point position

If your `Can` and `CanIf` parameters are already configured by a configuration import, you can use the CAN Buffer Assignment Editor without further configuration steps.

To use the CAN Bit Timing Editor, configure the source clock rate that your communication controllers use by configuring the parameter `CanCpuClockRef`. For more information on `CanCpuClockRef`, see Section 4.3.2.2.2.2, “Necessary parameters for bit timing configuration”. For information on how to configure `CanCpuClockRef`, see Section 4.3.2.4.1, “Configuring CanCpuClockRef”.

For information on how to configure the `CanIf` and `Can` system configuration parameters completely by hand, see Section 4.3.2.2.2, “Necessary parameters”.

4.3.2.2.2. Necessary parameters

This section outlines the system level configuration elements that must exist before you can use the CAN Buffer Assignment Editor and the CAN Bit Timing Editor.

WARNING 

Manual configuration is prone to errors and therefore not recommended

If you configure the required configuration parameters manually, do not omit any parameter and enter the parameters in the form in which the `Can` and `CanIf` modules expect them. The `Can` and `CanIf` modules cannot detect all resulting errors, e.g. incorrectly entered CAN-IDs or filter masks. Incorrectly entered CAN-IDs may lead to incorrectly sent or received PDUs in the configured ECUs.

To avoid configuration errors, see the documentation of the modules for information on their configuration.

If your `Can` and `CanIf` modules are configured via a configuration import and you only use the CAN Buffer Assignment Editor, you can skip this chapter. To use the CAN Bit Timing Editor, configure the source clock rate of your communication controllers by setting `CanCpuClockRef`. For information on how to configure this parameter, see Section 4.3.2.4.1, “Configuring CanCpuClockRef”.

If you configure the `CanIf` and `Can` system configuration parameters manually, this chapter outlines the elements you need to add. See the documentation of the `Can` and `CanIf` modules for detailed information. This chapter refers to the AUTOSAR 4.0 and 4.2/4.3/4.4 configuration parameters. Information about the AUTOSAR 4.0 and 4.2/4.3/4.4 release and the `Can` and `CanIf` SWS documents can be obtained from [1].

4.3.2.2.2.1. Necessary parameters for PDU to HOH assignment

`CanIf/CanIfInitCfg`

The CAN Buffer Assignment Editor supports the `CanIfInitCfg` container, encompassing their sent and received PDUs in the subcontainer lists `CanIfRxPduCfg` and `CanIfTxPduCfg`. If you want to define HOHs for PDU transmission, these must reside in `CanIfInitHohCfg/CanIfHrhCfg` and `CanIfInitHohCfg/CanIfHthCfg`.

`CanIf/CanIfCtrlDrvCfg/CanIfCtrlCfg`

For every communication controller that connects the configured ECU to a CAN network, one configuration container must exist as subcontainer. The parameter `CanIfCtrlCanCtrlRef` must refer to the representation of the communication controller in the `Can` configuration module.

`CanIf/CanIfInitCfg/CanIfRxPduCfg`

The containers for received PDUs must have the following parameters defined:

- `CanIfRxPduCanId` if the PDU is received via one specific CAN-ID.
- If the PDU owns a CAN-ID range, i.e. it can be received by any CAN-ID within a given range, the parameters of the subcontainer `CanIfRxPduCanIdRange` must be configured. `CanIfRxPduCanIdRangeLowerCanId` and `CanIfRxPduCanIdRangeUpperCanId` define the lower and upper bounds of the range.
- `CanIfRxPduCanIdType`.
- One or more references to assigned HOHs within `CanIfRxPduHrhIdRef` (only if the PDU has already been assigned to one or more HOHs).

`CanIf/CanIfInitCfg/CanIfTxPduCfg`

The containers for sent PDUs must have the following parameters defined:

- `CanIfTxPduCanId`.
- `CanIfTxPduCanIdType`.
- `CanIfTxPduType`.
- `CanIfTxPduBufferRef` (only if the PDU has already been assigned to an HOH).

If HOHs for sending/receiving PDUs are configured, the CAN Buffer Assignment Editor requires the correct setting for several parameters.

`CanIf/CanIfInitCfg/CanIfInitHohCfg/CanIfHrhCfg`

The containers for HOHs used for receiving PDUs must have the following parameters defined:

- `CanIfHrhCanCtrlIdRef`.
- `CanIfHrhIdSymRef`.

`CanIf/CanIfInitCfg/CanIfInitHohCfg/CanIfHthCfg`

The containers for HOHs used to send PDUs must have the following parameters defined:

- `CanIfHthCanCtrlIdRef`.

- CanIfHthIdSymRef.

CanIf/CanIfInitCfg/CanIfBufferCfg

CanIfBufferCfg containers, which are referenced by sending PDUs, must have exactly one CanIfBufferHthRef defined. More than one CanIfBufferHthRef results in a warning and only the first CanIfBufferHthRef is used.

Can/CanConfigSet/CanController

For every controller instance in the CanIf configuration, a corresponding instance must exist in the Can configuration. The CanIf instance must reference the Can instance by the parameter CanIfCtrlCanCtrlRef. If any of the controller's HOHs requires a filter mask, it must be configured in the CanFilterMask subcontainer of Can/CanConfigSet/CanController.

Can/CanConfigSet/CanHardwareObject

For every HOH instance in the CanIf configuration, a corresponding instance must exist in the Can configuration. The CanIf instance must reference the Can instance by the parameter CanIfHrhIdSymRef or CanIfHthIdSymRef. If the HOH requires a filter mask, AUTOSAR 4.0 Can module configurations must reference the corresponding container in Can/CanConfigSet/CanHardwareObject/CanFilterMaskRef. AUTOSAR 4.2/4.3/4.4 Can module configurations must provide the filter mask in Can/CanConfigSet/CanHardwareObject/CanHwFilter/CanHwFilterMask.

The containers for HOHs must have the following parameters defined:

- CanHandleType
- AUTOSAR 4.0 Can module configurations must provide CanIdValue
- AUTOSAR 4.2/4.3/4.4 Can module configurations must provide CanHwFilter/CanHwFilterCode

4.3.2.2.2.2. Necessary parameters for bit timing configuration

Can/CanConfigSet/CanController

For every communication controller that is attached to a CAN network, a corresponding configuration container must exist in the Can configuration. The CAN Bit Timing Editor requires that the container in turn contains at least one CanControllerBaudrateConfig subcontainer.

Can/CanConfigSet/CanController/CanCpuClockRef

Every communication controller that is attached to a CAN network requires a source clock from which it derives its own clock rate. The source clock of the communication controller is referenced in Can/CanConfigSet/CanController/CanCpuClockRef. The reference target is /Mcu/McuModuleConfiguration/McuClockSettingConfig/McuClockReferencePoint. McuClockReferencePointFrequency defines the rate of the clock. It is required that the clock rate is an integral multiple of the bit rate on the CAN bus and (if used) of the CAN-FD bit rate.

Since AUTOSAR system models do not provide the hardware properties of a given ECU, a configuration import cannot configure this parameter. This reference must therefore be configured by hand. For more information, see Section 4.3.2.4.1, “Configuring CanCpuClockRef”.

`Can/CanConfigSet/CanController/CanControllerDefaultBaudrate`
: The `CanControllerDefaultBaudrate` must refer to one of the `CanControllerBaudrateConfig` subcontainers in `Can/CanConfigSet/CanController`. The CAN Bit Timing Editor uses the referenced parameter for bit timing configuration.

`Can/CanConfigSet/CanController/CanControllerBaudrateConfig`
: The `CanControllerBaudrateConfig` configuration container provides the CAN bit rate of the attached network in `CanControllerBaudRate` in units of kilobits per second and the bit timing parameters that define the sample point for bits transmitted in the CAN bit rate. These parameters are given in multiples of a *time quantum/Tq*. The CAN Bit Timing Editor reads and writes these parameters:

- `CanControllerPropSeg` defines the propagation time segment
- `CanControllerSeg1` defines the phase buffer segment 1
- `CanControllerSeg2` defines the phase buffer segment 2
- `CanControllerSyncJumpWidth` defines the resynchronization jump width

`Can/CanConfigSet/CanController/CanControllerBaudrateConfig/CanControllerFdBaudrateConfig`
: The optional `CanControllerFdBaudrateConfig` configuration container is enabled if the CAN configuration controller supports CAN FD. `CanControllerFdBaudRate` provides the CAN FD bit rate of the attached network in units of kilobits per second and the bit timing parameters that define the sample point for bits transmitted in the CAN FD bit rate. These parameters are given in multiples of a *time quantum/Tq*. The CAN Bit Timing Editor reads and writes these parameters:

  - `CanControllerPropSeg` defines the propagation time segment
  - `CanControllerSeg1` defines the phase buffer segment 1
  - `CanControllerSeg2` defines the phase buffer segment 2
  - `CanControllerSyncJumpWidth` defines the resynchronization jump width

  In addition to that, the CAN Bit Timing Editor also reads and writes the parameters `CanControllerTrcvDelayCompensationOffset` and `CanControllerTxBitRateSwitch`. For a detailed description of these parameters, see the related `Can` SWS document, [1].

4.3.2.2.3. Resource files

The CAN wizards are already shipped with a resource file tailored to the peculiarities of your hardware platform. If you are not interested in how the CAN Buffer Assignment Editor determines which HOHs are available or how the CAN Bit Timing Editor determines the bit timing peculiarities for your platform, you can skip this section.

HOH groups
: Every hardware platform defines one or more `HOH groups`. These groups represent one or more of the platforms HOHs that exhibit the following properties:

- `Direction`: Defines whether the HOHs that belong to this group can be used to send, to receive, or both.
- `Filter mask`: In order to receive more than one PDU, a filter mask must be assigned to an HOH. This property defines whether the HOHs that belong to this group:
  - define their own individual filter mask,
  - refer to a shared filter mask, or
  - may not define a filter mask at all.
- `MCU/CC level`: This property defines whether the HOHs of the group are:
  - directly available at each of the CAN cells communication controllers (CC)s, or
  - whether the HOHs are defined on MCU level that implies that the HOHs are shared and can only be assigned to one of the CAN cells CCs.

Filter masks

Several hardware platforms define filter mask registers which can be shared among several HOHs. These shared filter masks can either be defined at the CC level, or at the the MCU level. CC level filter masks can be used within each CC individually. MCU level filter masks must be shared among all communication controllers of the node.

Don't-care-bit value

Although AUTOSAR `Can` defines the semantics of the filter mask value by defining "0" as a don't-care-bit, `Can` implementations exist that assume that a logical "1" in a filter mask identifies a don't-care-bit. To configure filter masks correctly, the resource file therefore also has to provide the don't-care-bit value for the current `Can` implementation.

TIP

CAN Buffer Assignment Editor displays don't-care-bits as logical zeros

The CAN Buffer Assignment Editor displays filter mask values as if the don't-care-bit is defined as a logical zero in the **Filter Mask** field within the **CanIf Harwdware Object Handles** grid. If an AUTOSAR `Can` implementation requires don't-care-bits to be logical ones, the CAN Buffer Assignment Editor:

1. Reads the filter masks in.
2. Inverts the filter masks.
3. Writes the filter masks back out.

You can see the inverted filter mask value in the **Inv. Filter Mask** field.

Shift FilterMask of standard CAN identifiers

If an MCAL configuration generator tool requires, for instance, that the 11 bits of a standard CAN identifier (and of the filter mask) reside in the 11 most significant bits of a 29-bit value, the value must be shifted left by 18 before it is written to the configuration. This value indicates by how many bit positions the filter mask is shifted.

Shift FilterMask of extended CAN identifiers

Same as above for extended (29-bit) CAN identifiers.

Controller types

Multiple controller types that differ by the number of available HOHs and/or filter masks can be defined for hardware platforms that support two or more kinds of controller. Each of the controllers in your configuration is assigned to one controller type. The assigned controller type can be changed in the GUI, see Figure 4.2, “CAN Buffer Assignment Editor”.

bit timing register to values offset

Some MCAL `Can` modules require that bit timing related parameters are configured in the representation that the corresponding register of the underlying hardware expects. The value range `[1..8]` for instance can be represented in a three bit hardware register that contains register values in the range of `[0..7]`. If the MCAL configuration generator module is not able to do the translation from the original value range to the register value range itself, the offset of the two value ranges can be provided in the resource file. The values that are affected by the offset are `CanControllerPropSeg`, `CanControllerSeg1`, `CanControllerSeg2`, and `CanControllerSyncJumpWidth` for the CAN and CAN-FD bit timing configurations.

bit timing register constraints

Some MCAL `Can` modules constrain the values of two or more bit timing parameters in relation to each other. For instance, an MCAL `Can` module might require that one parameter must always be less or equal to another parameter, or it might require that the sum of several parameters must not exceed a certain value. For instance `PropSeg + Seg1 + Seg2 <= 24` implies that the sum of the parameters `CanControllerPropSeg`, `CanControllerSeg1`, and `CanControllerSeg2` must not be larger than `24`. Such constraints can be specified independently for the CAN and the CAN-FD bit timing register set.

4.3.2.2.3.1. Missing resource file

The CAN Buffer Assignment Editor and the CAN Bit Timing Editor stop with the error message displayed in Figure 4.1, “Missing resource file error message” if no resource file could be found for the target and derivative of your `Can` module.

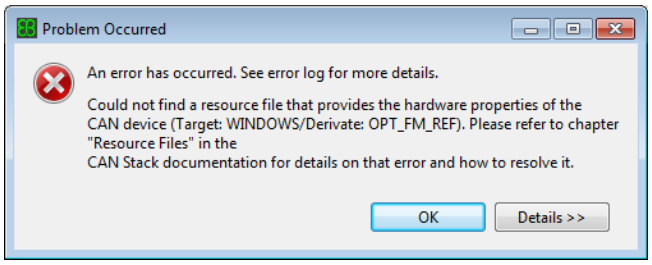


Figure 4.1. Missing resource file error message

This error occurs under the following scenarios:

- *Sub-derivative not configured correctly*: If your `Can` module supports sub-derivatives, verify that you correctly configured the sub-derivative of your hardware. Refer to your MCAL manual for information on how to correctly configure the sub-derivative.

  If the error occurs although the sub-derivative is configured correctly, the resource file for your sub-derivative is missing.
- *No resource file available for the derivative*: If your `Can` module only supports one derivative (i.e. it does not support sub-derivatives) and the error message in Figure 4.1, “Missing resource file error message” is displayed, this means that your `Can` module was shipped without any resource file.

The resource file might be missing for the following reasons:

- The peculiarities of the underlying hardware cannot be modeled by means of a resource file for your `Can` module or for your specific sub-derivative of your `Can` module.
- EB did not integrate your `Can` module into the ACG communication stack yet. Usually, a resource file is created and tested in the course of such an integration.

Contact https://www.elektrobit.com/support/ if you think that your `Can` module was accidentally delivered without resource file, or if you want EB to create a resource file for your `Can` module.

4.3.2.2.4. The CAN Buffer Assignment Editor GUI

This chapter describes the GUI of the CAN Buffer Assignment Editor. Figure 4.2, “CAN Buffer Assignment Editor” shows the various fields of the main window.

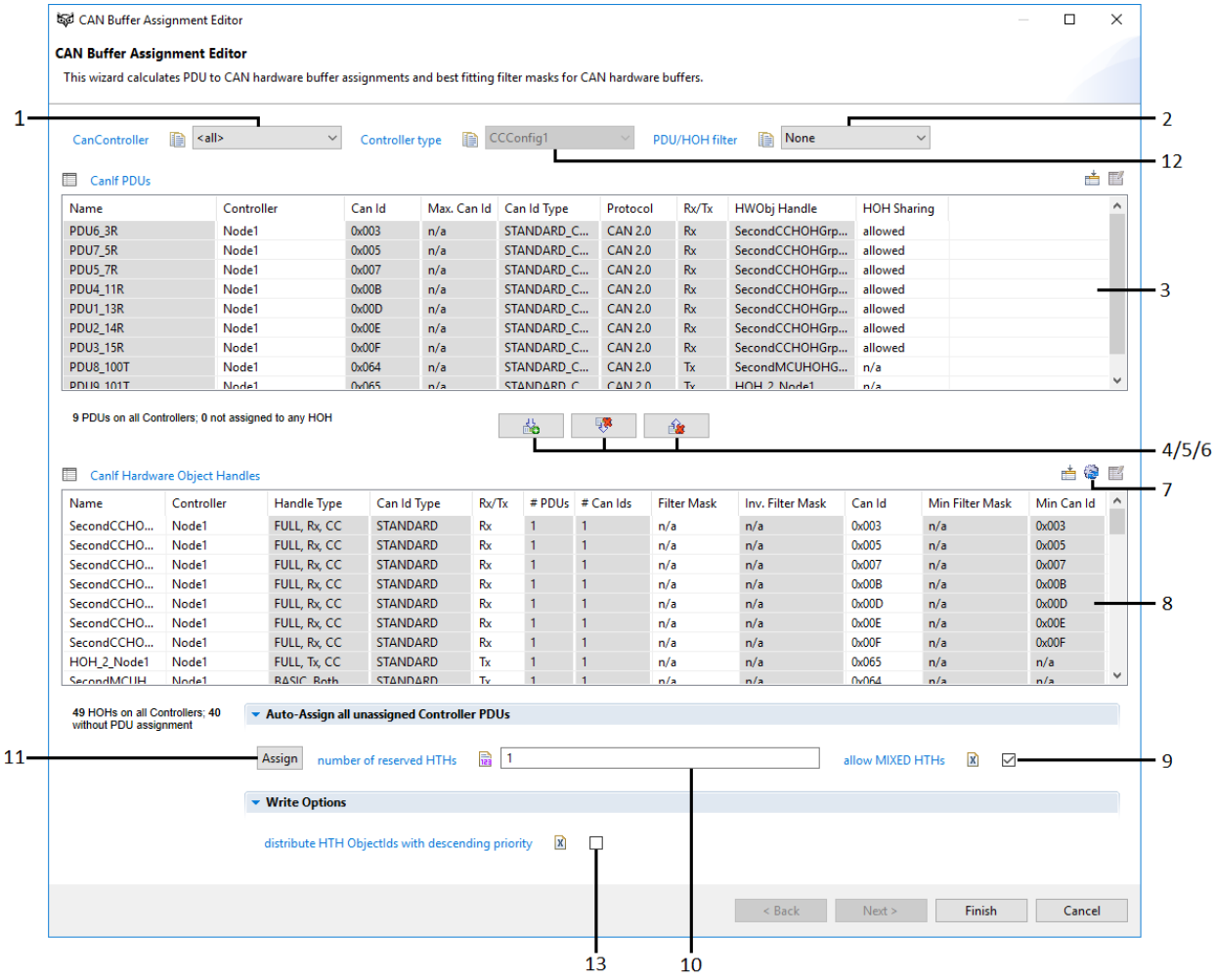


Figure 4.2. CAN Buffer Assignment Editor

Number on image	Item	Description
1	**Controller selector**	The Controller selector is used to filter PDU and HOH data of a specific communication controller. It contains one entry for each CAN communication controller found in the `CanIf` configuration and one additional entry, **<all>**, which is selected per default. If this entry is selected, PDUs and HOHs of all communication controllers are displayed.
2	**PDU/HOH filter**	The PDU/HOH filter provides the values: ▶ **None** ▶ **Filter HOHs of selected PDUs** ▶ **Filter PDUs of selected HOHs**

Number on image	Item	Description
		It is used to filter PDUs which are assigned to a specific HOH and vice versa. If you select **None**, no filter is applied.
3	**CanIf PDUs** grid	This grid displays all PDUs that the node either sends or receives. Section 4.3.2.2.4.1, “The PDU grid” describes the columns of the PDU grid in detail.
4	**Clear selected HOHs** button	If you click this button, you can remove the PDU assignment from all currently selected HOHs.
5	**Clear selected PDUs** button	If you click this button, you can remove the HOH assignment of all currently selected PDUs.
6	**Assign PDUs to HOH** button	If you click this button, you can assign all currently selected PDUs to the currently selected HOH.
7	**Optimize** button	If you click this button, you can remove unnecessary don't-care-bits from the **Filter Mask** and **Can Id** parameters of the currently selected Rx HOHs. All assigned PDUs can still be received.
8	**CanIf Hardware Object Handles** grid	This grid shows the HOHs that can be used to send or receive PDUs. Section 4.3.2.2.4.2, “The Hardware Object Handles grid” describes the columns of the **CanIf Hardware Object Handles** grid in detail.
9	**allow MIXED HTHs** check box	If you activate this check box, the CAN Buffer Assignment Editor auto assignment algorithm may assign sent standard and extended CAN identifier PDUs to the same HOHs, to configure MIXED HTHs. Deactivate this option, if the `Can` driver in use does not allow MIXED HTHs.
10	**Reserved HTHs** field	This field is used to configure the number of HOHs the auto assignment algorithm tries to reserve for PDU transmission. For details, see Section 4.3.2.3.5, “Automatically assigning PDUs to HOHs”.
11	**Assign** button	If you click this button, the CAN Buffer Assignment Editor tries to find a valid assignment for all currently unassigned PDUs. The CAN Buffer Assignment Editor considers the available hardware resources and several other side constraints. The auto assignment algorithm is described in detail in Section 4.3.2.3.5, “Automatically assigning PDUs to HOHs”.

Number on image	Item	Description
12	**Controller type selector**	The controller type selector is used to assign a different controller type to your CAN communication controller as provided in the resource file (see Section 4.3.2.2.3, “Resource files”). The selector is only enabled if a specific controller is selected, more than one controller type is available and a license for this feature is present.
13	**Distribute HTH ObjectIds with descending priority**	This check box controls how HTH ObjectIds are distributed when the configuration is written after the **Finish** button was selected. If this checkbox is selected, HTHs which are sending PDUs that have a lower CAN-ID assigned will obtain a lower ObjectId than HTHs which are sending PDUs that have a higher CAN-ID assigned. Activate this check box if you use a CAN driver that requires this specific HTH ordering to avoid priority inversion, i.e. to avoid that sending a lower priority/higher CAN-ID PDU blocks the sending of a higher priority/lower CAN-ID PDU.

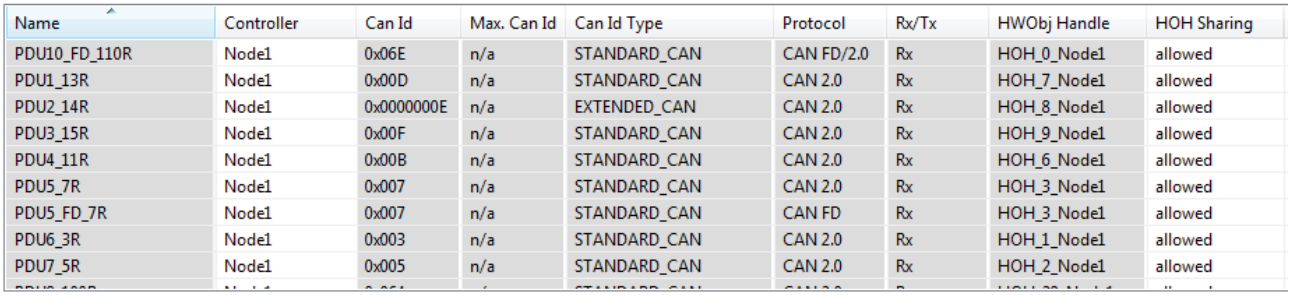
NOTE

Controller type re-assignment will delete existing HOH assignments

If you assign a new controller type to your CAN communication controller, the CAN Buffer Assignment Editor deletes all previously existing HOH assignments of all CAN communication controllers in your configuration. Therefore, if you assign a new controller type to one or more CAN communication controllers of your configuration you have to set up the HOH assignments again.

4.3.2.2.4.1. The PDU grid

The PDU grid allows to select one or more PDUs, and to change the read-writable properties of these one by one. Click one or more times onto the column labels to sort the elements in the grid according to the column content either in ascending or descending order. The PDU grid contains the following columns:



Name	Controller	Can Id	Max. Can Id	Can Id Type	Protocol	Rx/Tx	HWObj Handle	HOH Sharing
PDU10_FD_110R	Node1	0x06E	n/a	STANDARD_CAN	CAN FD/2.0	Rx	HOH_0_Node1	allowed
PDU1_13R	Node1	0x00D	n/a	STANDARD_CAN	CAN 2.0	Rx	HOH_7_Node1	allowed
PDU2_14R	Node1	0x0000000E	n/a	EXTENDED_CAN	CAN 2.0	Rx	HOH_8_Node1	allowed
PDU3_15R	Node1	0x00F	n/a	STANDARD_CAN	CAN 2.0	Rx	HOH_9_Node1	allowed
PDU4_11R	Node1	0x00B	n/a	STANDARD_CAN	CAN 2.0	Rx	HOH_6_Node1	allowed
PDU5_7R	Node1	0x007	n/a	STANDARD_CAN	CAN 2.0	Rx	HOH_3_Node1	allowed
PDU5_FD_7R	Node1	0x007	n/a	STANDARD_CAN	CAN FD	Rx	HOH_3_Node1	allowed
PDU6_3R	Node1	0x003	n/a	STANDARD_CAN	CAN 2.0	Rx	HOH_1_Node1	allowed
PDU7_5R	Node1	0x005	n/a	STANDARD_CAN	CAN 2.0	Rx	HOH_2_Node1	allowed

Figure 4.3. The PDU grid

Column	Description
Name	Displays the PDUs short name as retrieved from the `CanIf` configuration
Controller	Displays the name of the CAN communication controller which sends or receives the PDU. If the CAN Buffer Assignment Editor is not able to determine the controller that processes one or more PDUs, this column displays **<undefined>**. In this case, assign the controller to the unassigned PDU as follows: 1. Select the PDU in the column **Name**. 2. Click the **Controller** column of the selected element. 3. Select the desired controller from the displayed controller list. To assign a controller to multiple PDUs at once, you can use the bulk change dialog.
Can Id	Displays in hexadecimal format the CAN-ID of the PDU as it is retrieved from the `CanIf` configuration. If the PDU owns a CAN-ID range, this value represents the minimum CAN-ID of the range.
Max. Can Id	Displays in hexadecimal format the maximum CAN-ID, if the PDU owns a CAN-ID range.
Can Id Type	Displays the CAN-ID type of the PDU as retrieved from the `CanIf` configuration. **STANDARD_CAN** indicates that the PDU is assigned an 11-bit CAN identifier. **EXTENDED_CAN** indicates a 29-bit extended frame CAN identifier
Protocol	Displays information whether the PDU is transmitted on the bus as **CAN 2.0** PDU, as **CAN FD** PDU, or whether it may appear on the bus in both **CAN FD/2.-0** formats.
Rx/Tx	Indicates whether the PDU is sent (**Tx**), or received (**Rx**).
HWObjHandle	If the PDU is currently assigned to an HOH, the name of the HOH is displayed in this column. An empty value indicates that the PDU currently has not been assigned an HOH.
HOH Sharing	This column is only applied to Rx PDUs; If the current value is **allowed**, this implies that during auto assignment the PDU may be assigned to an HOH that al-

Column	Description
	so receives other PDUs. If the value is set to **forbidden**, the auto assignment reserves a dedicated Rx HOH for the PDU. See Section 4.3.2.3.5, “Automatically assigning PDUs to HOHs” for a detailed description of auto assignment. To set the HOH Sharing property for multiple PDUs at once, you can use the bulk change dialog.

NOTE

Received PDUs with identical CAN identifiers must have their HOH Sharing set to allowed

If one communication controller receives one CAN FD PDU and one CAN 2.0 PDU via the same CAN identifier, both PDUs must be assigned to the same HOH. Therefore, the **HOH Sharing** field must be left at **allowed** for both PDUs.

4.3.2.2.4.2. The Hardware Object Handles grid

The Hardware Object Handles grid allows you to select one or more HOHs and to change the read-writable properties of these one by one. Click the column labels to sort the elements in the grid according to the column content either in ascending or descending order.

The elements in the Hardware Object Handles grid are derived from HOHs that are actually present in the `Can` and `CanIf` configurations. Additionally the CAN Buffer Assignment Editor detects whether the hardware platform allows the configuration of additional HOHs. If the hardware platform allows this, the additional HOHs are also added to the grid.

The Hardware Object Handles grid contains the following columns:

Name	Controller	Handle Type	Can Id Type	Rx/Tx	# PDUs	# Can Ids	Filter Mask	Inv. Filter Mask	Can Id	Min Filter Mask	Min Can Id
HOH_3_Node1	Node1	BASIC, Both, CC	STANDARD	Rx	2	1	n/a	n/a	0x007	n/a	0x007
HOH_0_Node1	Node1	BASIC, Both, CC	STANDARD	Rx	1	1	n/a	n/a	0x06E	n/a	0x06E
HOH_31_Node1	Node1	BASIC, Both, CC	STANDARD	Rx	1	1	n/a	n/a	0x065	n/a	0x065
HOH_29_Node1	Node1	BASIC, Both, CC	STANDARD	Rx	1	1	n/a	n/a	0x064	n/a	0x064
HOH_9_Node1	Node1	BASIC, Both, CC	STANDARD	Rx	1	1	n/a	n/a	0x00F	n/a	0x00F
HOH_8_Node1	Node1	BASIC, Both, CC	EXTENDED	Rx	1	1	n/a	n/a	0x0000000E	n/a	0x0000000E
HOH_7_Node1	Node1	BASIC, Both, CC	STANDARD	Rx	1	1	n/a	n/a	0x00D	n/a	0x00D
HOH_6_Node1	Node1	BASIC, Both, CC	STANDARD	Rx	1	1	n/a	n/a	0x00B	n/a	0x00B
HOH_2_Node1	Node1	BASIC, Both, CC	STANDARD	Rx	1	1	n/a	n/a	0x005	n/a	0x005

Figure 4.4. The Hardware Object Handles grid

Column	Description
Name	Displays the HOHs short name as retrieved from the `CanIf` configuration. If the HOH does not yet exist in the `Can` and `CanIf` configurations, the name is derived from the corresponding HOH group listed in the resource file. For details, see Section 4.3.2.2.3, “Resource files”. You may enter an arbitrary AUTOSAR short name as long as it is unique among all HOHs.

Column	Description
Controller	Displays the name of the CAN communication controller that currently uses the HOH. If the `CanIf` and `Can` configurations contain more than one communication controller and the HOH has been derived from an HOH group defined on the MCU level, the column displays **<undefined>**. You have to assign the HOHs to a communication controller to use them for automatic assignment as follows: 1. Select the HOH. 2. Click the **Controller** column of the selected element. 3. Select the desired controller from the displayed controller list. To assign a controller to multiple HOHs at once, you can use the bulk change dialog.
Handle Type	Displays the properties of the HOH. This column consists of three components: 1. The first component indicates whether the HOH is a *Full Can* (**FULL**) or *Basic Can* (**BASIC**) HOH. See Section 4.3.2.2.5, “Auto assignment” for a detailed description of *Full Can* and *Basic Can* HOHs. 2. The second component indicates whether the HOH can be solely used to ▶ send (**Tx**), ▶ receive (**Rx**), ▶ or whether it allows the configuration to either send or receive (**Both**). 3. The last component indicates whether ▶ the HOH is owned by an individual controller (**CC**), ▶ or the HOH is defined on the MCU level and must therefore be assigned to one of the present controllers (**MCU**) before you can use it.
Can Id Type	Displays the CAN-ID type of the HOH. The type: ▶ **STANDARD** indicates that the HOH only processes PDUs that provide an 11-bit CAN identifier. ▶ **EXTENDED** indicates that the HOH only processes CAN PDUs that provide a 29-bit CAN identifier. ▶ **MIXED** means that PDUs of both CAN identifier types can be processed.
Rx/Tx	Indicates whether the HOH is currently used to transmit (**Tx**), or receive (**Rx**).
# PDUs	Indicates the number of PDUs the HOH processes.
# Can Ids	For Tx HOHs, this field displays the number of distinct CAN identifiers of the PDUs that are sent on this HOH. For Rx HOHs, a CAN identifier and filter mask parameter pair defines the set of CAN identifiers that can be received. The

Column	Description
	number of don't-care-bits in the filter mask defines the number of CAN identifiers which can potentially be processed by the HOH, which is `2 ^ (#dontCareBits)`. Usually a filter mask is chosen in such a way that the number of totally received CAN identifiers is the same as the number of distinct CAN identifiers of processed PDUs, i.e. that no unwanted PDUs are received.
Filter Mask	If an HOH is a **BASIC Rx** HOH, it may define a filter mask to receive several PDUs. See Section 4.3.2.2.5, “Auto assignment” for a detailed description of *Full Can* and *Basic Can* HOHs. The don't-care-bits are always displayed as logical zeros in this field. You may enter the desired value either in decimal or in hexadecimal format. The value is always displayed in hexadecimal format.
Inv. Filter Mask	Displays the inverted filter mask, which displays don't-care-bits as logical ones.
Can Id	Defines the CAN-ID used by **Rx** HOHs to receive PDUs. If used in combination with the **Filter Mask** value, the CAN-ID bit value at the filter mask's don't-care-positions is not considered for PDU reception. You may enter the desired value either in decimal or in hexadecimal format. The value is always displayed in hexadecimal format.
Min Filter Mask	Displays the filter mask which contains the least don't-care-bits necessary to receive the HOHs Rx PDUs. This parameter is only relevant for BASIC Rx HOHs.
Min Can Id	Displays the CAN-ID which, together with the **Min Filter Mask**, can be used to receive all the assigned Rx PDUs. This parameter is only relevant for BASIC Rx HOHs.

4.3.2.2.5. Auto assignment

This chapter explains the auto assignment algorithm in detail. If you are not interested in the details of auto assignment, you may skip this section and proceed at Section 4.3.2.3.5, “Automatically assigning PDUs to HOHs”.

In the following, PDUs are grouped according to their properties:

- `Tx/Rx PDUs` Tx and Rx PDUs are PDUs which are sent or received by the node.
- `Assigned/Unassigned PDUs` Assigned PDUs are assigned to exactly one `HOH`. Unassigned PDUs are not yet assigned.
- `Shared/Individual Rx PDUs` Individual Rx PDUs require to be assigned to one dedicated HOH for reception. However, a set of shared Rx PDUs may be assigned to one common HOH. Shared Rx PDUs have their `HOH Sharing` parameter set to `allowed`, whereas individual Rx PDUs set the parameter to `forbidden`.

HOHs are grouped according to the following properties:

- `Assigned/Unassigned HOHs` Assigned HOHs have one or more PDUs assigned for reception/transmission. Unassigned HOHs are not assigned to any PDU.
- `Rx/Tx HOHs, HOHs for both directions` Rx HOHs are HOHs which may only receive PDUs, whereas Tx HOHs may only send PDUs. HOHs for both directions can be used to either receive or send PDUs.
- `BASIC/FULL Rx HOHs` FULL Rx HOHs only define the `CAN Id` property and therefore only receive the CAN PDU with this specific CAN-ID. BASIC Rx HOHs additionally define a `Filter Mask` to specify the bits in the `CAN Id` property that have to match in order to receive a CAN PDU. The bit positions in the `CAN Id` that may have an arbitrary value are called don't-care-bits.
- `BASIC Rx HOHs with shared/own Filter Mask` BASIC Rx HOHs with their own filter mask may define their filter mask value freely. BASIC Rx HOHs with shared filter mask have to share their filter mask value with other BASIC Rx HOHs. Since there are usually less filter masks than BASIC Rx HOHs available, the assignment algorithm must take care that the number of used different filter mask values does not exceed the number of available shared filter masks.

1. In the first step, the auto assignment algorithm determines the number of Tx HOHs that are already assigned to PDUs. Then the algorithm tries to reserve additional Tx HOHs until the desired number of Tx HOHs (GUI parameter) is reached or no more Tx HOHs are available. If at least one Tx HOH is reserved, the algorithm continues. If no Tx HOH is present after the reservation process, the algorithm terminates and issues an error message.
2. Then, all unassigned Tx PDUs are sorted according to their CAN-ID. All available Tx HOHs are sorted according to the number of PDUs they currently process. In the next step, the Tx PDU with the lowest CAN-ID is assigned to the Tx HOH with the fewest assigned Tx PDUs. After the assignment, PDUs and HOHs are sorted again and the assignment is repeated until no unassigned Tx PDUs remain.
3. In the next step, the algorithm retrieves all unassigned shared Rx PDUs and all basic Rx HOHs which have already assigned at least one PDU. The algorithm checks for every PDU/HOH combination, whether the CAN-ID of the PDU matches the HOHs filter mask and CAN-ID value. If the algorithm detects a match, the PDU is assigned to the HOH.
4. Thereafter the algorithm retrieves all unassigned individual Rx PDUs. For each of these PDUs the algorithm tries to reserve an Rx HOH. If the algorithm detects that not enough HOHs are available, the assignment process is terminated and an error message is issued. If enough HOHs are available, the individual Rx PDUs are assigned to their HOHs.
5. If the previous assignment steps are successful, only unassigned shared Rx PDUs remain to be assigned. These shared Rx PDUs remain for assignment. They cannot be processed by BASIC RX HOHs which already exist. The shared Rx PDUs are distributed among the remaining free BASIC and FULL Rx HOHs. The distribution uses already allocated and free shared filter masks.

4.3.2.2.6. The CAN Bit Timing Editor GUI

This chapter describes the GUI of the CAN Bit Timing Editor. To launch the CAN Bit Timing Editor, click the **Can Assistant: Edit CAN Bit Timing** menu item in the **Sidebar** view. The following prerequisites must be fulfilled:

- You provided the necessary system level parameters for `Can`, either by configuration import or by defining them by hand. For more information, see Section 4.3.2.2.2.2, “Necessary parameters for bit timing configuration”.
- You configured `CanCpuClockRef` manually. For more information, see Section 4.3.2.4.1, “Configuring CanCpuClockRef”.

For information on how to retrieve the **Sidebar** view, see Section 4.3.2.3, “Using the CAN Buffer Assignment Editor”.

After the CAN Bit Timing Editor has started, its main window is displayed as shown in Figure 4.5, “CAN Bit Timing Editor Main Window”.

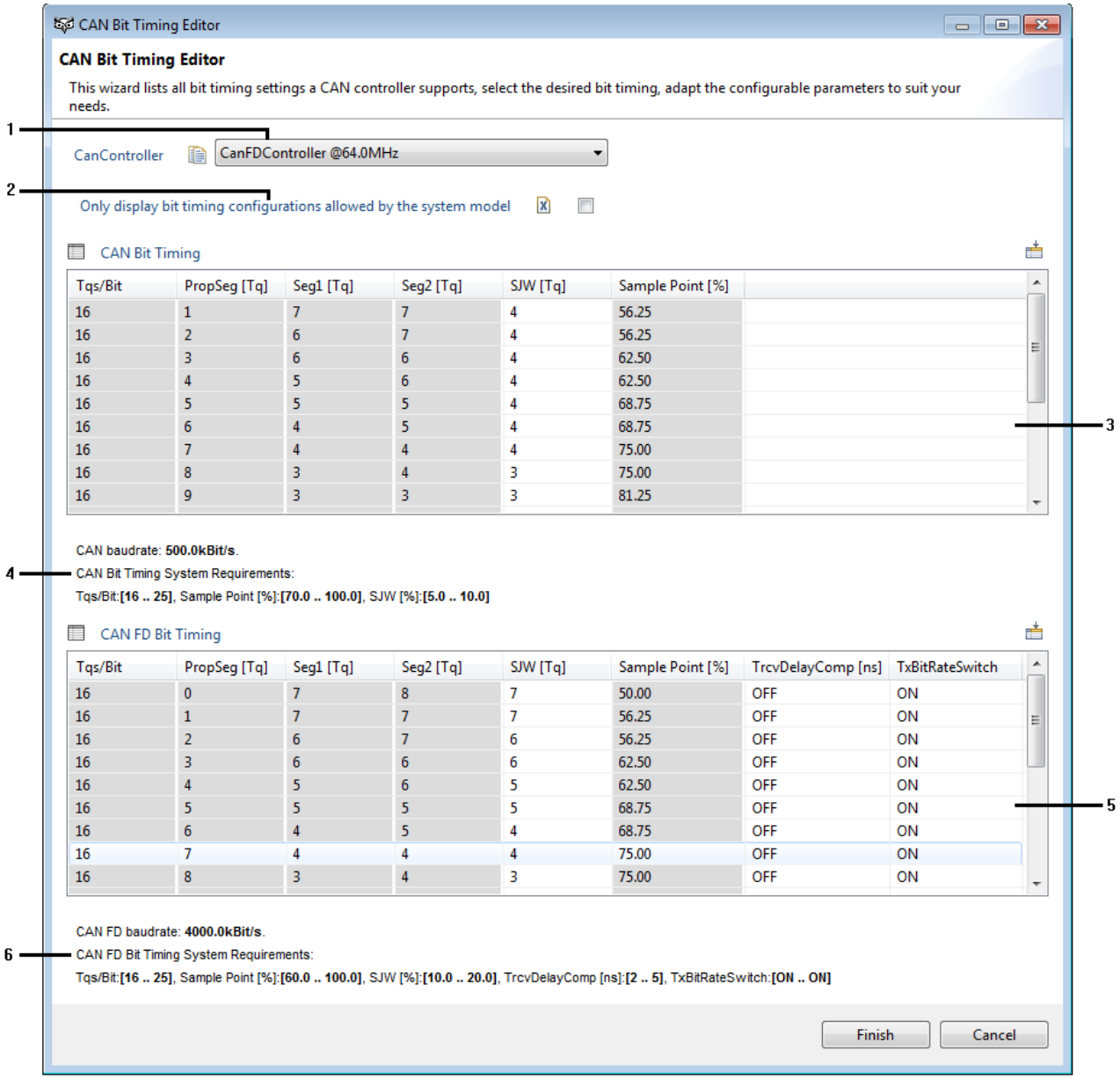


Figure 4.5. CAN Bit Timing Editor Main Window

Number on image	Item	Description
1	**CanController** drop-down list box	Use the **CanController** drop-down list box to select the CAN communication controller for which the bit timing registers shall be displayed and/or configured. The drop-down list box contains one entry for each `CanController` container found in the `Can` configuration. The entry consists of the container name and its configured source clock rate.

Number on image	Item	Description
2	System requirements filter	An AUTOSAR system model may contain requirements on the allowed range of bit timing registers. If you activate this check box, the bit timing grids for CAN and CAN FD only display configurations that adhere to these requirements. If you deactivate this check box, the bit timing grids display all configurations that the hardware supports.
3 and 5	**CAN Bit Timing** and **CAN FD Bit Timing** grids	The CAN bit timing and CAN FD bit timing grids display the bit timing configurations that can be configured for the currently selected CAN controller. The CAN FD bit timing grid is empty if the currently selected CAN Controller does not support CAN FD. For more information, see Section 4.3.2.2.6.1, “CAN Bit Timing grid” and Section 4.3.2.2.6.2, “CAN FD Bit Timing grid”.
4 and 6	**CAN Bit Timing** and **CAN FD Bit Timing** property fields	The CAN bit timing and CAN FD bit timing property fields display the bit rate of the connected network in CAN and CAN FD mode as well as the system model requirements related to the bit timing configurations in CAN and CAN FD mode.

4.3.2.2.6.1. CAN Bit Timing grid

The **CAN Bit Timing** grid displays all bit timing configurations that you can configure for the currently selected CAN communication controller. Click one or more times onto the column labels to sort the elements in the grid according to the column content either in ascending or descending order. The **CAN Bit Timing** grid contains the following columns:

Column	Description
Tqs/Bit	Displays the number of Tqs that make up the duration of one bit (1/(CanControllerBaudRate * 1000)). The number of Tqs per bit is also given by `SyncSeg + PropSeg + Seg1 + Seg2`. `SyncSeg` is not displayed in a dedicated column since its value is always `1`.
PropSeg [Tq]	Displays the number of Tqs that define the length of the propagation time segment/`CanControllerPropSeg`.
Seg1 [Tq]	Displays the number of Tqs that define the length of the phase buffer segment 1/`CanControllerSeg1`.
Seg2 [Tq]	Displays the number of Tqs that define the length of the phase buffer segment 2/`CanControllerSeg2`.

Column	Description
SJW [Tq]	Displays the number of Tqs that define the length of the resynchronization jump width/`CanControllerSyncJumpWidth`. You can change this value in the column.
Sample Point [%]	Displays the sample point in percent of the whole bit time.

NOTE

Value range of SJW [Tq] is restricted if the system requirements filter is activated

An activated system requirements filter may restrict the allowed values for `CanControllerSyncJumpwWidth`. If you want to configure a value that does not appear in the list of allowed values, first deactivate the system requirements filter check box, then select the desired value.

4.3.2.2.6.2. CAN FD Bit Timing grid

The **CAN FD Bit Timing** grid displays all bit timing configurations that can be configured for the currently selected CAN communication controller in CAN FD mode. Click one or more times onto the column labels to sort the elements in the grid according to the column content either in ascending or descending order. The **CAN FD Bit Timing** grid contains the following columns in addition to the columns listed in Section 4.3.2.2.6.1, “CAN Bit Timing grid”:

Column	Description
TrcvDelayComp [ns]	Displays the value of the optional parameter `CanControllerTrcvDelayCompensationOffset` given in nanoseconds. A value of `OFF` indicates that the parameter is currently disabled. To enable the parameter, enter a positive integral value. To set the parameter to `OFF` again, delete the value.
TxBitRateSwitch	Displays the value of the boolean parameter `TxBitRateSwitch`. You can set it to `ON` or `OFF`.

NOTE

Value range of TrcvDelayComp [ns] is restricted if the system requirements filter is activated

If the system requirements filter check box is activated and a system model requirement restricts the value range of **TrcvDelayComp**, the CAN Bit Timing Editor adapts the input value if it does not adhere to the system requirement. If you want to configure a value that does not adhere to the system model requirement, deactivate the system requirements filter check box before you can set the value.

NOTE

Value range of TxBitRateSwitch is restricted if the system requirements filter is activated

If the system requirements filter check box is activated and a system model requirement restricts the value of **TxBitRateSwitch** either to `ON` or to `OFF`, the CAN Bit Timing Editor does not allow to set **TxBitRateSwitch** to a different value. If you want to configure a value that does not adhere to the system model requirement, deactivate the system requirements filter check box before you can set the value.

4.3.2.3. Using the CAN Buffer Assignment Editor

This chapter provides instructions on how to assign hardware object handles (HOHs) to PDUs with the CAN Buffer Assignment Editor.

You may launch the CAN Buffer Assignment Editor, when you provide the necessary system level parameters for `Can` and `CanIf`. You can import the configuration or define it manually.

The CAN Buffer Assignment Editor is launched via the **Sidebar** view of EB tresos Studio. If the **Sidebar** view is not displayed, you can activate it as follows:

1. Select **Window** from the tool bar.
2. Select **Show View**.
3. Select **Sidebar**.

To launch the CAN Buffer Assignment Editor, click **Edit CAN Buffer Assignment** in the **Sidebar** view.

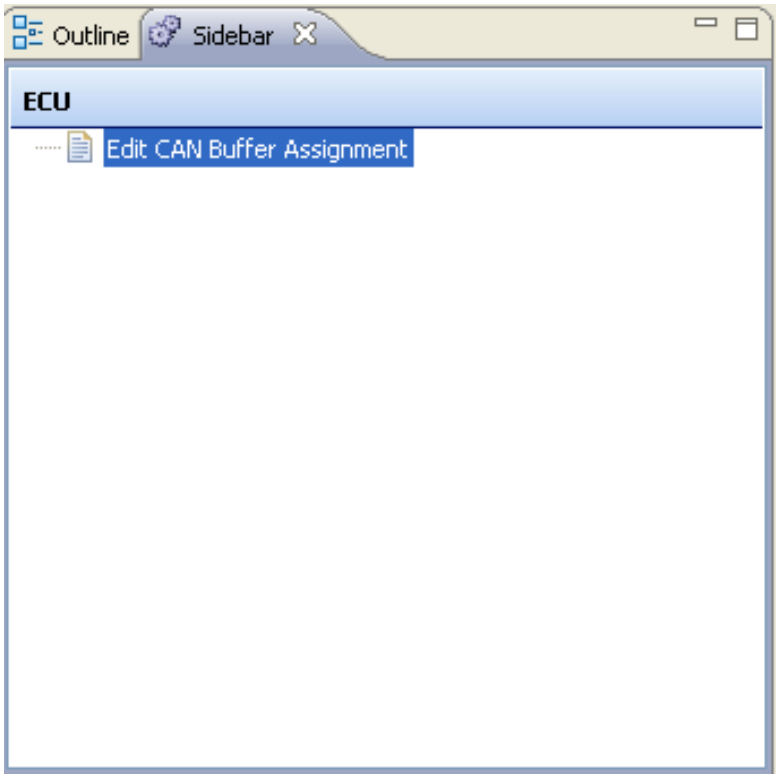


Figure 4.6. The CAN Buffer Assignment Editor in the **Sidebar** view

TIP

If you do not see the CAN Buffer Assignment Editor entry in the Sidebar view

If your **Sidebar** view does not contain any entries:

1. Check if the `Can` and `CanIf` configuration is loaded.
2. Check if the project or any project subelement is selected in the **Project Explorer** view.
3. Right-click the configuration in the **Project Explorer** view to load a configuration.
4. Select **Load Configuration**.

On starting the CAN Buffer Assignment Editor GUI, a popup dialog appears if there are MCU level HOHs without controller assignment. Select **OK** if you want these HOHs to be assigned automatically.

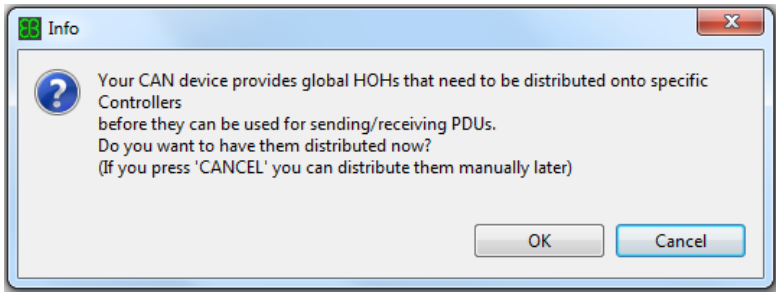


Figure 4.7. Popup dialog asking for automatic assignment of unassigned **MCU level HOHs**

After the CAN Buffer Assignment Editor has successfully started, its main window is displayed. Figure 4.2, “CAN Buffer Assignment Editor” shows the main window.

To use the CAN Buffer Assignment Editor, see the following instructions:

- Section 4.3.2.3.1, “Clearing HOH assignment of PDUs”
- Section 4.3.2.3.2, “Clearing PDU assignment of HOHs”
- Section 4.3.2.3.3, “Optimizing filter mask and CAN-ID parameters”
- Section 4.3.2.3.4, “Assigning PDUs to an HOH”
- Section 4.3.2.3.5, “Automatically assigning PDUs to HOHs”
- Section 4.3.2.3.6, “Writing the assignment back to Can and CanIf”

4.3.2.3.1. Clearing HOH assignment of PDUs

To clear the HOH assignment of one or more PDUs:

1. Select the PDUs in the PDU grid.
2. Click the **Clear selected PDUs** button.

4.3.2.3.2. Clearing PDU assignment of HOHs

To clear the PDU assignment of one or more HOHs:

1. Select the HOHs in the Hardware Object Handles grid
2. Click the **Clear selected HOHs** button.

NOTE	**If you remove all PDU assignments, the Filter Mask and the Can Id fields are reset** The **Filter Mask** and the **Can Id** fields of a BASIC Rx HOH in the Hardware Object Handles grid are reset if all of its PDU assignments are removed.

4.3.2.3.3. Optimizing filter mask and CAN-ID parameters

NOTE

Optimizing the filter mask might increase the number of required filter mask registers

If your platform only provides a limited number of shared filter mask registers, you have to use the following feature carefully. When you click the **Optimize...** button, it usually leads to a new filter mask value. This new value requires an additional filter mask register on the hardware. If the number of configured filter masks exceeds the number of filter masks that are actually available in the hardware:

- the `Can` module usually reports an error,
- no `Can` configuration code is generated.

Click the **Optimize FilterMask and CanId to assigned PDUs** button to reduce the number of unwanted PDUs. This allows you to make the hardware filter mask of one or more BASIC HOHs as restrictive as possible.

4.3.2.3.4. Assigning PDUs to an HOH

To assign one or more PDUs to an HOH:

1. Select the PDUs in the PDU grid.
2. Select one HOH in the HOH grid.
3. Then click the **Assign PDUs to HOH** button.

There are several constraints if you want to assign one or more PDUs to an HOH manually.

- *Same direction of PDUs, HOH:* Only assignments of PDUs with the same Rx/Tx direction to an HOH that supports this direction are allowed.
- *Multiple Rx PDUs to BASIC HOH:* If you want to assign more than one Rx PDU to an HOH, ensure that the HOH is a BASIC HOH. Assignment of more than one Rx PDU to a FULL HOH is not allowed.
- *Same controller:* Make sure that the PDUs and the HOH are assigned to the same controller. Assignment of PDUs to HOHs across different controllers is not allowed.
- *PDUs with forbidden HOH Sharing:* Make sure that you assign PDUs with the **HOH Sharing** option set to **forbidden** only to HOHs that are not assigned other PDUs.

If the assignment is successful, the **Filter Mask** and **Can Id** fields of **BASIC** Rx HOHs are updated. All CAN-IDs of the assigned PDUs are accepted for reception.

NOTE

Implicit update of filter mask and Can-ID

If you move an Rx PDU from one BASIC Rx HOH to another, the **Filter Mask** and the **Can Id** fields of both HOHs are implicitly updated.

NOTE

Assign received PDUs with the same CAN identifier to the same HOH

If one communication controller receives one CAN FD PDU and one CAN 2.0 PDU via the same CAN identifier, you must assign both PDUs to the same Rx HOH to be able to receive both PDUs.

4.3.2.3.5. Automatically assigning PDUs to HOHs

You may use the auto assignment to assign all of your configurations PDUs that are not yet assigned to an HOH. PDUs with already assigned HOHs are left untouched from auto assignment.

TIP

Repeating the automatic assignment

If you want to repeat the automatic assignment for one or more than one PDU of your configuration:

1. Clear the HOH assignment for the PDUs you want to assign automatically to HOHs again.
2. Leave the HOH assignments of all other PDUs as they are.
3. Restart the auto assignment.

Similar steps are required if you want to repeat the automatic assignment for one or more HOHs.

To launch the auto assignment:

1. Enter the number of HOHs that the algorithm shall try to reserve for PDU transmission into the **reserved HTHs** field.
2. Click the **Auto-Assign all unassigned Controller PDUs** button to the left.

Depending on the number of PDUs to assign, the auto assignment algorithm may take several seconds to find an assignment.

TIP

Performing the automatic assignment for specific controllers

If you want to perform the automatic assignment function for all of your controllers, select **<all>** in the **CanController** combo box. If you want to perform the automatic assignment function for one specific controller, select the controller in the combo box.

4.3.2.3.6. Writing the assignment back to Can and CanIf

TIP

Check for problems first

Before you click the **Finish** button, check whether the CAN Buffer Assignment Editor detected any problems in your PDU to HOH assignment. Warnings and errors are displayed in the top part of the HOH Assignment Window. If there are any errors the **Finish** button is disabled.

To write the PDU to HOH assignment back to the `Can` and `CanIf` configurations, click the **Finish** button. The PDU to HOH assignment is written back provided there is no error in the configuration.

Every HOH with at least one PDU assigned is written back to the configuration. HOHs without any PDU assigned (any more) are removed from the configuration. If an HOH requires a filter mask to receive several PDUs, a filter mask container is added to the corresponding CAN controller configuration (if it does not already exist), then the `Can` HOHs filter mask link is set to this filter mask container.

When you click the **Finish** button, the **Results** view displays a tab called **Edit CAN Buffer Assignment**. Click on this tab to see all nodes and values that are changed or added by the CAN Buffer Assignment Editor. In addition, you see information or warning messages that occurred during the write process.

4.3.2.4. Using the CAN Bit Timing Editor

This chapter provides instructions on how to set up the bit timing configuration parameters using the CAN Bit Timing Editor

The CAN Bit Timing Editor is launched via the **Sidebar** view of EB tresos Studio. If the **Sidebar** view is not displayed, see Section 4.3.2.3, “Using the CAN Buffer Assignment Editor” for instructions on how to activate it.

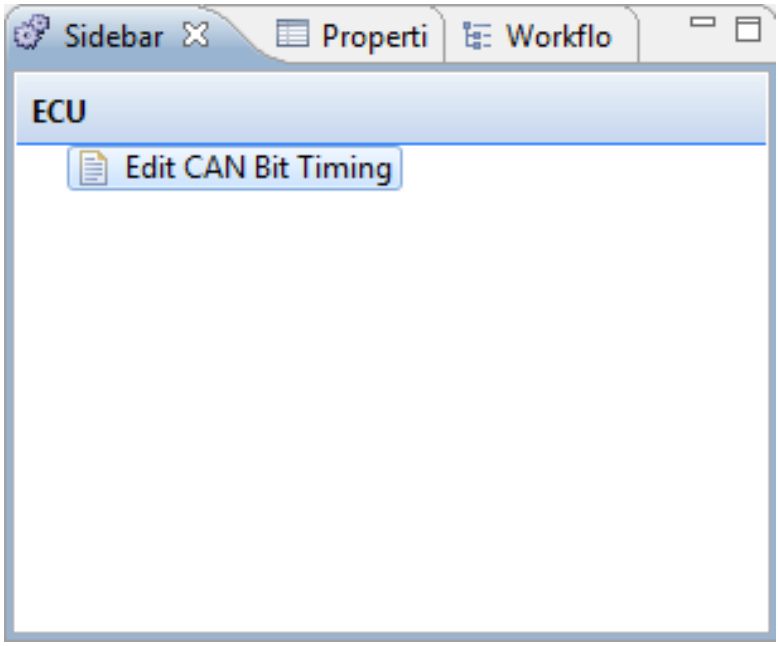


Figure 4.8. The CAN Bit Timing Editor in the **Sidebar** view

TIP

If you do not see the CAN Bit Timing Editor entry in the Sidebar view

If your **Sidebar** view does not contain any entries:

1. Check if the `Can` and `Mcu` configurations are loaded.
2. Check if the project or any project subelement is selected in the **Project Explorer** view.
3. Right-click the configuration in the **Project Explorer** view to load a configuration.
4. Select **Load Configuration**.

4.3.2.4.1. Configuring CanCpuClockRef

If the CAN Bit Timing Editor issues an error message related to the parameter `CanCpuClockRef`, configure this parameter in the EB tresos Studio configuration project:

1. Select the `CanController` container for which `CanCpuClockRef` must be configured.
2. Update the parameter `CanCpuClockRef`. For an example, see Figure 4.9, “Configuring `CanCpuClock-Ref`”.

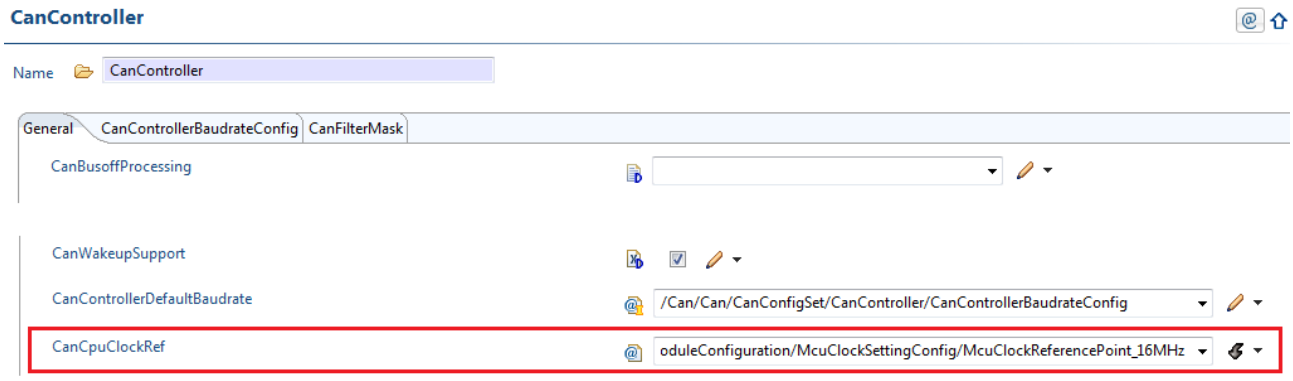


Figure 4.9. Configuring `CanCpuClockRef`

After the CAN Bit Timing Editor has started, its main window is displayed as shown in Figure 4.5, “CAN Bit Timing Editor Main Window”.

For information on how to use the CAN Bit Timing Editor, see the following instructions:

- Section 4.3.2.4.2, “Configuring the CAN and CAN FD bit timing parameters”
- Section 4.3.2.4.3, “Writing the bit timing configuration back to Can”

4.3.2.4.2. Configuring the CAN and CAN FD bit timing parameters

To configure the CAN bit timing parameters of a CAN communication controller, take the following steps:

1. Select the controller in the **CanController** drop-down list box.
2. Select the desired bit timing configuration in the **CAN Bit Timing** grid.
3. Set the **SJW [Tq]** field of the selected bit timing configuration to the desired value.

To configure the CAN FD bit timing parameters of a CAN communication controller supporting CAN FD, take the following steps:

1. Select the controller in the **CanController** drop-down list box.
2. Select the desired bit timing configuration in the **CAN FD Bit Timing** grid.
3. Set the **SJW [Tq]**, **TrcvDelayComp [ns]**, **TxBitRateSwitch** fields of the selected bit timing configuration to the desired values.

4.3.2.4.3. Writing the bit timing configuration back to Can

To write the bit timing configurations back to the `Can` configuration, click the **Finish** button.

When you click the **Finish** button, the **Results** view displays a tab called **Edit CAN Bit Timing**. Click this tab to see all nodes and values that are changed or added by the CAN Bit Timing Editor. In addition, you see information or warning messages that occurred during the write process.

4.3.3. Using the automatic CAN buffer assignment wizard

The functionality for CAN buffer assignment as described in Section 4.3.2.3, “Using the CAN Buffer Assignment Editor” can also be executed via command line or the unattended wizard **AutoConfigure CAN Buffer Assignment (CanAs.AutoConfigure)** in EB tresos Studio.

4.3.3.1. Command line

In order to run the unattended wizard via the command line, provide the given parameters as shown in the following example:

```
tresos_cmd.bat -DCanAsController=Sender -DCanAsHTHNumber=10
      -DCanAsAllowMixedHTHs=true -DCanAsSortHTHsByCanId=true

autoconfigure Demo CanAs.AutoConfigure
```

With these parameters, the unattended wizard is executed for the project named `Demo` as follows:

- The controller name is set to "Sender".
- The number of reserved HTHs is set to 10.
- Mixed HTHs are allowed.
- HTHs that send higher priority PDUs obtain lower ObjectId values than HTHs that send lower priority PDUs.

You can set the parameters as follows:

`-DCanAsController=<parameter>`
: This parameter specifies the name of the controller that contains the HOHs that are assigned. If this parameter is not provided, the HOHs of all controllers are assigned.

`-DCanAsHTHNumber=<parameter>`
: This parameter sets the number of reserved HTHs. The minimum value of this parameter is 1.

`-DCanAsAllowMixedHTHs=<parameter>`
: This parameter specifies whether mixed HTHs are allowed. Valid values are `true` or `false`. If no value is given, `true` is assumed.

`-DCanAsSortHTHsByCanId=<parameter>`
: This parameter specifies how the ObjectId values of HTHs shall be distributed. Valid values are `true` or `false`. If no value is given, `false` is assumed. Section 4.3.2.2.4, “The CAN Buffer Assignment Editor GUI”

provides a detailed description of the parameter in the section *Distribute HTH ObjectIds with descending priority*.

`autoconfigure`
: This parameter indicates that only the unattended wizard for the automatic CAN Buffer assignment shall be executed.

`<project name>`
: The name of the project on which the unattended wizard shall perform its operation.

`CanAs.AutoConfigure`
: The ID of the unattended wizard.

MCU level HOHs without controller assignment are distributed to controllers depending on the number of PDUs that the controllers send or receive.

4.3.3.1.1. Unattended wizard in EB tresos Studio

To configure the automatic CAN buffer assignment:

1. Select the menu item **Unattended wizard configuration**.
2. Select the sub item **Autoconfigure CAN Buffer Assignment(CanAs.AutoConfigure)**.
3. Configure the parameters in the window that appears, see Figure 4.11, “Configuring the unattended wizard for the automatic CAN buffer assignment”. Section 4.3.2.2.4, “The CAN Buffer Assignment Editor GUI” provides a description of the parameters.
4. Select the menu item **Autoconfigure CAN Buffer Assignment(CanAs.AutoConfigure)** as depicted in Figure 4.10, “Starting the unattended wizard for the automatic CAN buffer assignment” if you want to launch the automatic CAN buffer assignment for the selected project.

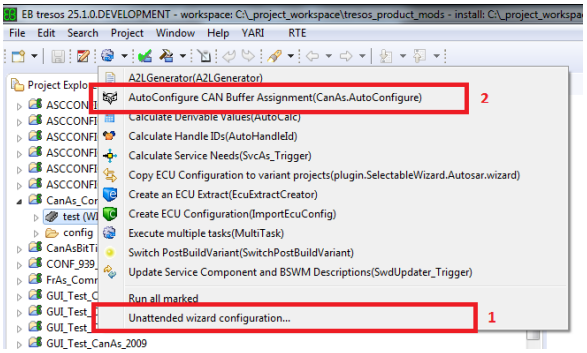


Figure 4.10. Starting the unattended wizard for the automatic CAN buffer assignment

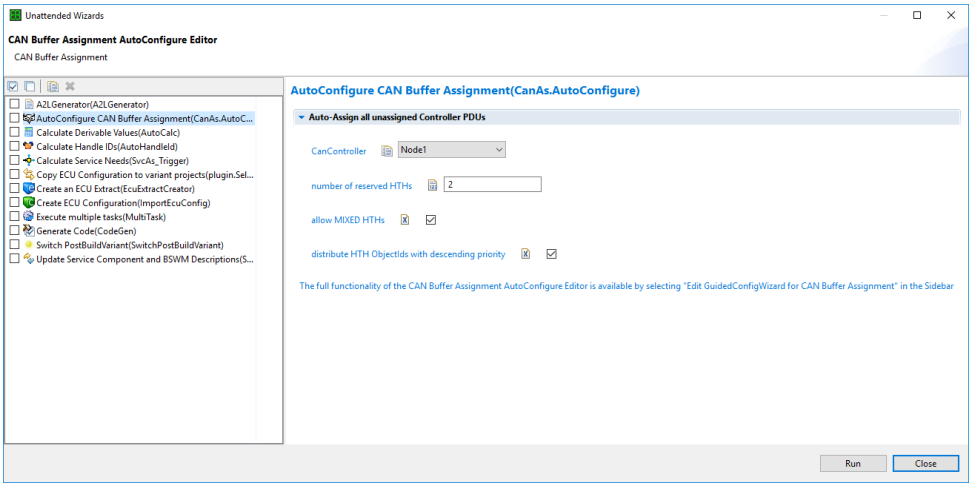


Figure 4.11. Configuring the unattended wizard for the automatic CAN buffer assignment

4.3.4. Configuring CAN MetaData

This section provides a starting point on how to configure the MetaData feature. For background information, see Section 4.2.3, “CANStack MetaData”.

Configuring CAN MetaData

Step 1
In `CanIf`, enable the configuration parameter `CanIfMetaDataSupport` to use the MetaData feature.

Step 2
In `CanTp`, enable the configuration parameter `CanTpDynIdSupport` to use the MetaData feature.

Step 3
Configure the `AddressingFormat` to CANTP_NORMALFIXED for related PDUs.

Each N-SDU that has `AddressingFormat` configured as CANTP_NORMALFIXED should reference an `EcuC` PDU with `MetaDataItemType` of type SOURCE_ADDRESS_16 and TARGET_ADDRESS_16.

Each N-PDU that has `AddressingFormat` configured as CANTP_NORMALFIXED should reference an `EcuC` PDU with `MetaDataItemType` of type CAN_ID_32.

Step 4
To support the handling of generic connections using N-SDUs with MetaData, enable the configuration parameter `CanTpGenericConnectionSupport`. When enabled, both N_Ta and N_Sa have dynamic values.

Step 5
In `CanIfTxPduCanIdMask`, define which bits of the CAN-ID reside in `CanIfTxPduCanId` and which are provided via Metadata.

CanTp throws an error concerning an invalid CAN-ID mask when CanIfTxPduCanIdMask overrides the N_Sa, N_Ta, or uni-/multicast info bit (bits 0..16).

4.3.5. (Prototype) Configuring the multicore distribution along network boundaries

This section provides a starting point on how to configure the CAN stack in order to distribute different CAN channels on different partitions. For background information, see Section 4.2.4, “(Prototype) Multicore distribution along network boundaries”.

Configuring the multicore distribution along network boundaries

Prerequisite:

- To use the feature, you must select a Can driver with multi-core support capability for this project. For details on how to configure the Can driver in a multi-core context, see its corresponding documentation.

Step 1
In the Can driver, CanControllerEcucPartitionRef should be configured for each controller by referencing the desired partition.

Step 2
In CanIf enable CanIfMultiCoreSupport.

Step 3
If CanIfDecoupledProcessingSupport is enabled, then for each CanIfTx/RxProcessing container, CanIfPartitionRef must be configured with the partition reference on which the corresponding main function will be processed.

Step 4
All Pdus belonging to a CAN channel should be mapped in EcuC to the same partition as the one referenced in CanControllerEcucPartitionRef and also the one referenced in CanIfPartitionRef, if decoupled processing is used.

This way it is ensured that all resources of a CAN channel remain on the same partition. Configuration messages are implemented to provide guidance.

Step 5
In CanSM enable CanSMMultiCoreSupport and CanSMDistributedChannelProcessingSupport.

TIP

Behavior in `CanSM`

For each partition referenced in `ComMChannelPartitionRef` via `CanSMComMNetworkHandleRef`, a new main function shall be created. CAN networks that do not have a `ComMChannelPartitionRef` configured will be processed in the generic main function as before.

On how to map main functions to different partitions in a multicore context please consult the `Rte` and `Os` corresponding documentation.

Step 6

Make sure that all controllers from a CAN Network are mapped to the same partition, via `CanControllerEcucPartitionRef`, as the ComM channel. If a transceiver with multicore capabilities is used, then the transceiver channnel should also be mapped to that respective partition.

Configuration messages are implemented to provide guidance.

Step 7

In `CanTp` enable `CanTpDedicatedChannelProcessingSupport` and `CanTpMultiCoreSupport`.

Create a `CanTpChannelProcessing` container for each partition on which you wish to distribute `CanTp`.

Reference the desired partition via `CanTpEcuCPartitionRef` and the `CanTp` channels via `CanTpChannelRef` which you wish to be processed on that partition.

TIP

Behavior in `CanTp`

For each `CanTpChannelProcessing` container a main function shall be created in which the processing of each `CanTp` channel referenced via `CanTpChannelRef` shall be performed.

Channels not mapped via `CanTpChannelRef` shall be processed in the generic main function as before.

On how to map main functions to different partitions in a multicore context please consult the `Rte` and `Os` corresponding documentation.

Step 8

Make sure that all Pdus from a `CanTp` channel are mapped in `EcuC` to the same partition as the channel.

Configuration messages are implemented to provide guidance.

4.4. CanNm module user guide

The network management module `CanNm` is part of the network and state management stack of EB tresos AutoCore.

For background information and configuration advice, see the EB tresos AutoCore Generic documentation, chapter *Network management and state management stack*.

5. ACG8 CAN Stack module references

5.1. Overview

This chapter provides module references for the ACG8 CAN 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 CAN 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. CanIf

5.2.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CanIfDefensiveProgramming	1..1	**Label:** Defensive Programming Options Parameters for defensive programming
CanIfCtrlDrvCfg	1..255	Configuration parameters for all the underlying CAN Driver modules are aggregated under this container. For each CAN Driver module a seperate instance of this container has to be provided.
CanIfDispatchCfg	1..1	Callback functions provided by upper layer modules of the CanIf. The callback functions defined in this container are common to all configured CAN Driver / CAN Transceiver Driver modules.
CanIfInitCfg	1..1	CanIfInitCfg contains the init parameter of the CAN Interface. Note: The number of instances of CanIfInitCfg is limited to 1, i.e. there is no multiple container support.
CanIfPrivateCfg	1..1	CanIfPrivateCfg contains the private configuration parameters of the CAN Interface.
CanIfPublicCfg	1..1	CanIfPublicCfg contains the public configuration parameters of the CAN Interface.
CanIfTrcvDrvCfg	0..n	CanIfTrcvDrvCfg contains the configuration parameters of all addressed CAN transceivers by each underlying CAN Transceiver Driver module. For each CAN Transceiver Driver a seperate instance of this container shall be provided.

Containers included		
CanIfDecoupledMeasurementSupport	1..1	CanIfDecoupledMeasurementSupport contains the configuration parameters of Rx and Tx decoupled processing measurement of the CAN Interface.
CanIfUpperLayerConfig	0..16	User upper layer configuration for CanIf. Any upper layer above CanIf needs a configuration in this list. This also applies for the well-known upper layers of CanIf which are CanTp, CanTSyn, CanNm, J1939Nm, J1939Tp and PduR. Upper layers are identified by their names. To add an entry for a well-known upper layer of CanIf the name of the container must be PDUR (for upper layer PduR), CAN_TP (for upper layer CanTp), CAN_TSYN (for upper layer CanTSyn), CAN_NM (for upper layer CanNm), J1939NM (for upper layer J1939Nm) or J1939TP (for upper layer J1939Tp). Any other name indicates an user defined upper layer (CDD). User defined upper layer also includes the AUTOSAR module Xcp. To assign a particular Pdu to a user specific upper CanIf module (this means any upper layer which is not PduR, CanNm, CanTSyn, CanTp, J1939Nm or J1939Tp), the parameter CanIfRxPduUpperLayerRef within the CanIfRxPduCfg configuration container and the parameter CanIfTxPduUpperLayerRef within the CanIfTxPduCfg configuration container must refer to an entry of this list.
CanIfMirroringSupport	1..1	The container contains Bus Mirroring related configuration parameters. The parameters from this container are editable if `CanIfBusMirroringSupport` is **enabled**.
CanIfHookOnRxSupport	1..1	The container contains Custom Hook on Reception related configuration parameters. The parameters from this container are editable if `CanIfHookOnReceptionSupport` is **enabled**.
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

Containers included		
		Additional published parameters not covered by Common-PublishedInformation 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. CanIfDefensiveProgramming

Parameters included	
Parameter name	**Multiplicity**
CanIfDefProgEnabled	1..1
CanIfPrecondAssertEnabled	1..1
CanIfPostcondAssertEnabled	1..1
CanIfStaticAssertEnabled	1..1
CanIfUnreachAssertEnabled	1..1
CanIfInvariantAssertEnabled	1..1

Parameter Name	CanIfDefProgEnabled
Label	Enable Defensive Programming
Description	Enables or disables the defensive programming feature for the module CanIf. 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	**CanIfPrecondAssertEnabled**
Label	Enable Precondition Assertions
Description	Enables handling of precondition assertion checks reported from the module CanIf. Dependency on parameter(s): ► Enable Development Error Detection (`CanIfPublicDevErrorDetect`): must be enabled ► Enable Defensive Programming (`CanIfDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfPostcondAssertEnabled**
Label	Enable Postcondition Assertions
Description	Enables handling of postcondition assertion checks reported from the module CanIf. Dependency on parameter(s): ► Enable Development Error Detection (`CanIfPublicDevErrorDetect`): must be enabled ► Enable Defensive Programming (`CanIfDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false

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

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

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

Parameter Name	**CanIfInvariantAssertEnabled**

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

5.2.1.2. CanIfCtrlDrvCfg

Containers included		
Container name	**Multiplicity**	**Description**
CanIfCtrlCfg	1..255	CanIfCtrlCfg contains the configuration parameter of an adressed CAN controller by an underlying CAN Driver module. This container is configurable per CAN controller.

Parameters included	
Parameter name	**Multiplicity**
CanIfCtrlDrvTxCancellation	1..1
CanIfCtrlDrvInitHohConfigRef	1..1
CanIfCtrlDrvNameRef	1..1

Parameter Name	**CanIfCtrlDrvTxCancellation**
Description	Selects whether transmit cancellation is supported and if the appropriate callback will be provided to the CAN Driver module. ► True: Enabled ► False: Disabled

	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	false	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfCtrlDrvInitHohConfigRef**	
Description	Reference to the Init Hoh Configuration.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfCtrlDrvNameRef**	
Description	CAN Interface driver reference. This reference can be used to get any information (Ex. Driver Name, Vendor ID) from the CAN Driver. The CAN Driver name can be derived from the SHORT-NAME of the CAN Driver module.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.3. CanIfCtrlCfg

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
CanIfCtrlId	1..1
CanIfCtrlWakeupSupport	1..1
CanIfCtrlCanCtrlRef	1..1
CanIfCtrlMaxRxNotifyPdus	1..1
CanIfCtrlMaxTxNotifyPdus	1..1
CanIfCtrlWakeupSourceInRef	0..1
CanIfCtrlWakeupSourceOutRef	0..1

Parameter Name	CanIfCtrlId	
Description	CanIfCtrlId abstracts from the CAN Driver specific parameter controller. Each controller of all connected CAN Driver modules shall be assigned to one specific ControllerId of the CanIf. Range: 0..number of configured controllers of all CAN Driver modules.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanIfCtrlWakeupSupport	
Description	CanIfCtrlWakeupSupport defines if a respective controller of the referenced CAN Driver modules is queriable for wake up events. ► True: Wakeup Support Enabled ► False: Wakeup Support Disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanIfCtrlCanCtrlRef
Description	CanIfCtrlCanCtrlRef references to the logical handle of the underlying CAN controller from the CAN Driver module to be served by the CAN Interface module. The following parameters of CanController config container shall be referenced by this link: CanControllerId, CanWakeupSourceRef.

	Range: 0..max. number of underlying supported CAN controllers	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfCtrlMaxRxNotifyPdus**	
Description	Maximum number of Rx-Pdus which support notification functions when Multi-core support is enabled. This configuration parameter allocates runtime memory per notified Rx-Pdu.	
Multiplicity	1..1	
Type	INTEGER	
Default value	64	
Range	<=65535 >=1	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfCtrlMaxTxNotifyPdus**	
Description	Maximum number of Tx-Pdus which support notification functions when Multi-core support is enabled. This configuration parameter allocates runtime memory per notfied Tx-Pdu.	
Multiplicity	1..1	
Type	INTEGER	
Default value	64	
Range	<=65535 >=1	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfCtrlWakeupSourceInRef**
Description	CanIfCtrlWakeupSourceInRef contains a reference to the wake up source for the controller as defined in the ECU State Manager.

	CanIfCtrlWakeupSourceInRef allows mapping of incoming wake up source (i.e. a wake up event is detected by the CanDrv Driver) to a specific contoller. Therefore the parameter is used as input to API CanIf_CheckWakeup(). The parameter is optional since providing the API CanIf_CheckWakeup() is also optional. If the parameter is used, CanIfCtrlakeupSourceOutRef must be set as well. Implementation Type: reference to EcuM_WakeupSourceType.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfCtrlWakeupSourceOutRef**	
Description	CanIfCtrlWakeupSourceOutRef contains a reference to the wake up source for the controller as defined in the ECU State Manager. CanIfCtrlWakeupSourceOutRef allows mapping of outgoing wake up source (i.e. wake up is signalled to the user notification API) to a specific controller. Therefore the parameter is used as output from CanIf_CheckWakeup() to <User_-SetWakeupEvent>. The parameter is also used for wake up validation, i.e. as input and output for API CanIf_CheckValidation(). The parameter is optional since providing the API CanIf_CheckWakeup() is also optional. Implementation Type: reference to EcuM_WakeupSourceType.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.4. CanIfDispatchCfg

Parameters included	
Parameter name	**Multiplicity**
CanIfDispatchUserCtrlBusOffName	1..1
CanIfDispatchUserCtrlBusOffUL	1..1

Parameters included	
CanIfDispatchUserCtrlModeIndicationName	1..1
CanIfDispatchUserCtrlModeIndicationUL	1..1
CanIfDispatchUserCheckTrcvWakeFlagIndicationName	0..1
CanIfDispatchUserCheckTrcvWakeFlagIndicationUL	0..1
CanIfDispatchUserClearTrcvWufFlagIndicationName	0..1
CanIfDispatchUserClearTrcvWufFlagIndicationUL	0..1
CanIfDispatchUserConfirmPnAvailabilityName	1..1
CanIfDispatchUserConfirmPnAvailabilityUL	1..1
CanIfDispatchUserTrcvModeIndicationName	1..1
CanIfDispatchUserTrcvModeIndicationUL	1..1
CanIfDispatchUserValidateWakeupEventName	1..1
CanIfDispatchUserValidateWakeupEventUL	1..1
CanIfTranslateTxCanIdFunc	0..1
CanIfTranslateRxCanIdFunc	0..1
CanIfDispatchUserSetWakeupEventName	1..1
CanIfDispatchUserSetWakeupEventUL	1..1

Parameter Name	CanIfDispatchUserCtrlBusOffName	
Description	CanIfDispatchUserCtrlBusOffName defines the name of <User_-ControllerBusOff>. CanIfDispatchUserCtrlBusOffName depends on the parameter CanIfDis-patchUserCtrlBusOffUL. If CanIfDispatchUserCtrlBusOffUL equals CAN_SM the name of <User_ControllerBusOff> is fixed to CanSM_ControllerBu-sOff. If CanIfDispatchUserCtrlBusOffUL equals CDD, the name of <User_-ControllerBusOff> is selectable.	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanIfDispatchUserCtrlBusOffUL
Description	CanIfDispatchUserCtrlBusOffUL defines the upper layer (UL) module to which the notifications of all ControllerBusOff events from the CAN Driver modules have to be routed via <User_ControllerBusOff>.

	The upper layer (UL) module as the provider of <User_ControllerBusOff> must always be configured.
Multiplicity	1..1
Type	ENUMERATION
Default value	CAN_SM
Range	CAN_SM CDD
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfDispatchUserCtrlModeIndicationName**
Description	CanIfDispatchUserCtrlModeIndicationName defines the name of <User_-ControllerModeIndication>. CanIfDispatchUserCtrlModeIndicationName depends on the parameter CanIfDispatchUserCtrlModeIndicationUL. If CanIfDispatchUserCtrlModeIndicationUL equals CAN_SM the name of <User_ControllerModeIndication> is fixed to CanSM_ControllerModeIndication. If CanIfDispatchUserCtrlModeIndicationUL equals CDD, the name of <User_ControllerModeIndication> is selectable.
Multiplicity	1..1
Type	FUNCTION-NAME
Configuration class	**PreCompile:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfDispatchUserCtrlModeIndicationUL**
Description	CanIfDispatchUserCtrlModeIndicationUL defines the upper layer (UL) module to which the notifications of all ControllerTransition events from the CAN Driver modules have to be routed via <User_ControllerModeIndication>.
Multiplicity	1..1
Type	ENUMERATION
Default value	CAN_SM
Range	CAN_SM CDD
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	CanIfDispatchUserCheckTrcvWakeFlagIndicationName	
Description	This parameter defines the name of <User_CheckTrcvWakeFlagIndication>.. If CANIF_DISPATCH_USERCHECKTRCVWAKEFLAGINDICATION_UL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CANIF_PUBLIC_PN_SUPPORT equals False, this parameter shall not be configurable.	
Multiplicity	0..1	
Type	FUNCTION-NAME	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanIfDispatchUserCheckTrcvWakeFlagIndicationUL	
Description	This parameter defines the upper layer module to which the CheckTrcvWakeFlagIndication from the Driver modules have to be routed.. If CANIF_PUBLIC_PN_SUPPORT equals False, this parameter shall not be configurable.	
Multiplicity	0..1	
Type	ENUMERATION	
Default value	CAN_SM	
Range	CAN_SM CDD	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanIfDispatchUserClearTrcvWufFlagIndicationName	
Description	This parameter defines the name of <User_ClearTrcvWufFlagIndication>.. If CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL equals CAN_SM the name of <User_ClearTrcvWufFlagIndication> is fixed. If it equals CDD, the name is selectable. If CANIF_PUBLIC_PN_SUPPORT equals False, this parameter shall not be configurable.	
Multiplicity	0..1	
Type	FUNCTION-NAME	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanIfDispatchUserClearTrcvWufFlagIndicationUL

Description	This parameter defines the upper layer module to which the ClearTrcvWufFlagIndication from the Driver modules have to be routed.. If CANIF_PUBLIC_PN_SUPPORT equals False, this parameter shall not be configurable.
Multiplicity	0..1
Type	ENUMERATION
Default value	CAN_SM
Range	CAN_SM CDD
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfDispatchUserConfirmPnAvailabilityName**
Description	This parameter defines the name of User_ConfirmPnAvailability. If CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL equals CAN_SM the name of User_ConfirmPnAvailability is fixed. If it equals CDD, the name is selectable. If CANIF_PUBLIC_PN_SUPPORT equals False, this parameter shall not be configurable.
Multiplicity	1..1
Type	FUNCTION-NAME
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfDispatchUserConfirmPnAvailabilityUL**
Description	This parameter defines the upper layer module to which the ConfirmPnAvailability notification from the Driver modules have to be routed. If CANIF_PUBLIC_PN_SUPPORT equals False, this parameter shall not be configurable.
Multiplicity	1..1
Type	ENUMERATION
Default value	CAN_SM
Range	CAN_SM CDD
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfDispatchUserTrcvModeIndicationName**

Description	CanIfDispatchUserTrcvModeIndicationName defines the name of <User_-TrcvModeIndication>. CanIfDispatchUserTrcvModeIndicationName depends on the parameter CanIfDispatchUserTrcvModeIndicationUL. If CanIfDispatchUserTrcvModeIndicationUL equals CAN_SM the name of <User_TrcvModeIndication> is fixed to CanSM_TrcvModeIndication. If CanIfDispatchUserTrcvModeIndicationUL equals CDD, the name of <User_TrcvModeIndication> is selectable.	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfDispatchUserTrcvModeIndicationUL**	
Description	CanIfDispatchUserTrcvModeIndicationUL defines the upper layer (UL) module to which the notifications of all TransceiverTransition events from the CAN Transceiver Driver modules are routed via <User_TrcvModeIndication>.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	CAN_SM	
Range	CAN_SM CDD	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfDispatchUserValidateWakeupEventName**	
Description	CanIfDispatchUserValidateWakeupEventName defines the name of <User_-ValidateWakeupEvent>. CanIfDispatchUserValidateWakeupEventName depends on the parameter CanIfDispatchUserValidateWakeupEventUL. If CanIfDispatchUserValidateWakeupEventUL equals ECUM the name of <User_ValidateWakeupEvent> is fixed to EcuM_ValidateWakeupEvent. If CanIfDispatchUserValidateWakeupEventUL equals CDD, the name of <User_ValidateWakeupEvent> is selectable. If parameter CanIfPublicWakeupCheckValidSupport is disabled, no <User_-ValidateWakeupEvent> API can be configured.	
Multiplicity	1..1	

Type	FUNCTION-NAME	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfDispatchUserValidateWakeupEventUL**	
Description	CanIfDispatchUserValidateWakeupEventUL defines the upper layer (UL) module to which the notifications about positive former requested wake up sources have to be routed via <User_ValidateWakeupEvent>. If parameter CanIfPublicWakeupCheckValidSupport is disabled, CanIfDispatchUserValidateWakeupEventUL cannot be configured.	
Multiplicity	1..1	
Type	ENUMERATION	
Range	CDD ECUM	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfTranslateTxCanIdFunc**
Label	CanIfTranslateTxCanIdFunc
Description	CanIfTranslateTxCanIdFunc can be used to configure the user specific CAN ID translation function/macro for Tx PDUs. If CanIfTranslateTxCanIdFunc is disabled, the configured CAN ID (parameter CanIfTxPduCanId) will be used when transmitting a message. If it is enabled, the configured function will be called with a CAN ID and the returned CAN ID will be used when the message is transmitted. The prototype of this function must be as follows (if a macro is used it must behave accordingly): Can_IdType function-name(Can_IdType CanId) **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	0..1

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

Parameter Name	**CanIfTranslateRxCanIdFunc**
Label	CanIfTranslateRxCanIdFunc
Description	CanIfTranslateRxCanIdFunc can be used to configure the user specific CAN ID translation function/macro for Rx PDUs. If CanIfTranslateRxCanIdFunc is disabled, the configured CAN ID (parameter CanIfRxPduCanId) will be expected by CanIf_RxIndication() when receiving a message. If it is enabled, CanIf_RxIndication() will call this function to translate the received CAN ID into the configured CAN ID (parameter CanIfRxPduCanId) of the corresponding Rx PDU before and use this translated CAN ID for software filtering. The prototype of this function must be as follows (if a macro is used it must behave accordingly): Can_IdType function-name(Can_IdType CanId) **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	0..1
Type	FUNCTION-NAME
Default value	CanIf_TranslateRxCanId
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfDispatchUserSetWakeupEventName**
Description	CanIfDispatchUserSetWakeupEventName defines the name of <User_-SetWakeupEvent>. CanIfDispatchUserSetWakeupEventName depends on the parameter CanIfDispatchUserSetWakeupEventUL. If CanIfDispatchUserSetWakeupEventUL equals

	ECUM the name of <User_SetWakeupEvent> is fixed to EcuM_SetWakeupEvent. If CanIfDispatchUserSetWakeupEventUL equals CDD, the name of <User_SetWakeupEvent> is selectable.	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfDispatchUserSetWakeupEventUL**	
Description	CanIfDispatchUserSetWakeupEventUL defines the upper layer (UL) module to which the notifications about positive former requested wake up sources have to be routed via <User_SetWakeupEvent>.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	ECUM	
Range	CDD ECUM	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.2.1.5. CanIfInitCfg

Containers included		
Container name	**Multiplicity**	**Description**
CanIfBufferCfg	0..n	This container contains the Txbuffer configuration. Multiple buffers with different sizes could be configured. If CanIfBufferSize (CANIF834_Conf) equals 0, the CanIf Tx L-PDU only refers via this CanIfBufferCfg the corresponding CanIfHthCfg..
CanIfInitHohCfg	1..255	This container contains the references to the configuration setup of each underlying CAN Driver.
CanIfRxPduCfg	0..n	CanIfRxPduCfg contains the configuration parameters of each Rx L-PDU. The SHORT-NAME of "CanIfRxPduConfig" container itself represents the symolic name of Rx L-PDU.

Containers included		
CanIfTxPduCfg	0..n	CanIfTxPduCfg contains the configuration parameters of a Tx L-PDU. It has to be configured as often as a Tx L-PDU is needed. The SHORT-NAME of "CanIfTxPduConfig" container represents the symbolic name of Tx L-PDU.

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

Parameter Name	CanIfInitCfgSet	
Description	CanIfInitCfgSet is not used by the CanIf and therefore can not be edited.	
Multiplicity	1..1	
Type	STRING	
Default value		
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.6. CanIfBufferCfg

Parameters included	
Parameter name	**Multiplicity**
CanIfBufferSize	1..1
CanIfBufferHthRef	1..1

Parameter Name	CanIfBufferSize
Description	This parameter defines the number of CanIf Tx L-PDUs which can be buffered in one Txbuffer. If this value equals 0, the CanIf does not perform Txbuffering for the CanIf Tx L-PDUs which are assigned to this Txbuffer. If CanIfPublicTxBuffering equals False, this parameter equals 0 for all TxBuffer. If the CanHandleType of the referred HTH equals FULL, this parameter equals 0 for this TxBuffer..
Multiplicity	1..1
Type	INTEGER

Default value	0	
Range	<=255 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfBufferHthRef**	
Description	Reference to HTH, that defines the hardware object or the pool of hardware objects configured for transmission. All the CanIf Tx L-PDUs refer via the CanIfBufferCfg and this parameter to the HTHs if TxBuffering is enabled, or not.. Each HTH shall not be assigned to more than one buffer.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.7. CanIfInitHohCfg

Containers included		
Container name	**Multiplicity**	**Description**
CanIfHrhCfg	0..n	CanIfHrhCfg contains configuration parameter for each hardware receive object (HRH).
CanIfHthCfg	0..n	CanIfHthCfg contains parameters related to each HTH.

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

Parameter Name	**CanIfInitRefCfgSet**
Description	Selects the CAN Interface specific configuration setup. This type of external data structure shall contain the post build initialization data for the CAN Interface for all underlying CAN Drivers.
Multiplicity	1..1

Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.8. CanIfHrhCfg

Containers included		
Container name	**Multiplicity**	**Description**
CanIfHrhRangeCfg	0..0	CanIfHrhRangeCfg is not used and therefore can not be edited. The range configuration is done by container CanIfRxPduCanIdRange instead.

Parameters included	
Parameter name	**Multiplicity**
CanIfHrhSoftwareFilter	1..1
CanIfHrhCanCtrlIdRef	1..1
CanIfHrhCanHandleTypeRef	0..1
CanIfHrhIdSymRef	1..1

Parameter Name	**CanIfHrhSoftwareFilter**	
Description	Selects the hardware receive objects by using the HRH range/list from CAN Driver configuration to define, for which HRH a software filtering is be performed during receive processing. ► True: Software filtering is enabled. ► False: Software filtering is enabled. **Optimization Effect:** ► **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	**CanIfHrhCanCtrlIdRef**	
Description	Reference to controller ID to which the HRH belongs to. A controller can contain one or more HRHs.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfHrhCanHandleTypeRef**	
Description	CanIfHrhCanHandleTypeRef shall refer to the same HRH as CanIfHrhIdSymRef. Therefore CanIfHrhCanHandleTypeRef can not be edited.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfHrhIdSymRef**	
Description	CanIfHrhIdSymRef refers to a particular HRH object in the CanDrv configuration (see CanHardwareObject CAN324_Conf) The CanIf receives the following information of the CanDrv module by this reference: ► CanHandleType (see CAN323_Conf) ► CanObjectId (see CAN326_Conf)	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.9. CanIfHrhRangeCfg

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
CanIfHrhRangeRxPduLowerCanId	1..1
CanIfHrhRangeRxPduRangeCanIdType	1..1
CanIfHrhRangeRxPduUpperCanId	1..1

Parameter Name	CanIfHrhRangeRxPduLowerCanId	
Description	CanIfHrhRangeRxPduLowerCanId is not used.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=536870911 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanIfHrhRangeRxPduRangeCanIdType	
Description	CanIfHrhRangeRxPduRangeCanIdType is not used.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	STANDARD	
Range	EXTENDED STANDARD	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanIfHrhRangeRxPduUpperCanId
Description	CanIfHrhRangeRxPduUpperCanId is not used.
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	<=536870911 >=0

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.10. CanIfHthCfg

Parameters included	
Parameter name	**Multiplicity**
CanIfHthCanCtrlIdRef	1..1
CanIfHthCanHandleTypeRef	0..1
CanIfHthIdSymRef	1..1

Parameter Name	**CanIfHthCanCtrlIdRef**	
Description	Reference to controller ID to which the HTH belongs to. A controller can contain one or more HTHs.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfHthCanHandleTypeRef**	
Description	CanIfHthCanHandleTypeRef shall refer to the same HTH as CanIfHthIdSymRef. Therefore CanIfHthCanHandleTypeRef can not be edited.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**Link:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfHthIdSymRef**
Description	CanIfHthIdSymRef refers to a particular HTH object in the CAN Driver module configuration. The HTH ID is unique in a given CAN Driver. The HTH IDs are defined in the CAN Driver module and hence it is derived from CAN Driver configuration.
Multiplicity	1..1

Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.11. CanIfRxPduCfg

Containers included		
Container name	**Multiplicity**	**Description**
CanIfRxPduCanIdRange	0..1	Range of CAN identifier of Rx L-PDUs used by the CAN Interface. CanIfRxPduCanIdRange is used if a range of CAN identifier is assigned to the PDU. If a single ID is assigned then the parameter CanIfRxPduCanId shall be used instead. The boundaries of the range are configured with the parameter CanIfRxPduCanIdRangeLowerCanId and CanIfRxPduCanIdRangeUpperCanId. **Optimization Effect:** ► **ROM increase (config):** Using this container increases the ROM consumption of the module configuration. ► **ROM increase (code):** Using this container increases the ROM consumption of the module code. ► **Execution time increase (code):** Using this container increases the execution time of the module code.
CanIfTTRxFrameTriggering	1..1	CanIfTTRxFrameTriggering is not used by the CanIf.

Parameters included	
Parameter name	**Multiplicity**
CanIfRxPduCanId	1..1
CanIfRxPduCanIdType	1..1
CanIfRxPduCanIdMask	1..1
CanIfRxPduDlc	1..1
CanIfRxPduId	1..1
CanIfRxPduReadData	1..1

Parameters included	
CanIfRxPduReadNotifyStatus	1..1
CanIfRxPduUserRxIndicationName	1..1
CanIfRxPduUserRxIndicationUL	0..1
CanIfRxPduBswSchExclAreaIdRef	1..1
CanIfRxPduHrhIdRef	1..n
CanIfRxPduRef	1..1
CanIfRxPduTargetPduID	1..1
CanIfRxPduUpperLayerRef	1..1

Parameter Name	CanIfRxPduCanId	
Description	CAN identifier of Rx L-PDUs used by the CAN Interface. CanIfRxPduCanId is used if exactly one CAN identifier is assigned to the PDU. If a range is assigned then the CanIfRxPduCanIdRange container shall be used instead. Range: ► 11 bit for standard CAN identifier. ► 29 bit for extended CAN identifier. Example: The parameter is used by the CanIf within the Software Filtering functions.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanIfRxPduCanIdType
Description	CanIfRxPduCanIdType describes the type of the CAN identifier of Rx L-PDUs used by the CAN Driver for CAN L-PDU reception. Range: ► EXTENDED_CAN for CAN 2.0 or CAN FD frame with extended identifier (29 bits). ► EXTENDED_FD_CAN for CAN FD frame with extended identifier (29 bits).

	► EXTENDED_NO_FD_CAN for CAN 2.0 frame with extended identifier (29 bits). ► STANDARD_CAN for CAN 2.0 or CAN FD frame with standard identifier (11 bits). ► STANDARD_FD_CAN for CAN FD frame with standard identifier (11 bits). ► STANDARD_NO_FD_CAN for CAN 2.0 frame with standard identifier (11 bits).
Multiplicity	1..1
Type	ENUMERATION
Default value	STANDARD_CAN
Range	EXTENDED_CAN EXTENDED_FD_CAN EXTENDED_NO_FD_CAN STANDARD_CAN STANDARD_FD_CAN STANDARD_NO_FD_CAN
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfRxPduCanIdMask**
Description	Identifier mask which denotes relevant bits in the CAN Identifier. This parameter defines a CAN Identifier range in an alternative way to CanIfRxPduCanIdRange. It identifies the bits of the configured CAN Identifier that must match the received CAN Identifier. Range: ► 11 bit for standard CAN identifier. ► 29 bit for extended CAN identifier.
Multiplicity	1..1
Type	INTEGER
Default value	536870911
Range	<=536870911 >=0
Configuration class	**PostBuild:** VariantPostBuild

Origin	AUTOSAR_ECUC

Parameter Name	**CanIfRxPduDlc**	
Description	Data Length code of Rx L-PDUs used by the CAN Interface. This information is used for DLC checking. A Rx L-PDU passes the DLC check if the received DLC is equal or greater than this value. If a Rx L-PDU passes the DLC check, CanIf passes the PDU to the upper layer module. Otherwise the Rx L-PDU is silently discarded. A value of 0 disables the DLC check for this PDU. If DLC check is disabled via the parameter CanIfPrivateDlcCheck this value has no effect on reception.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfRxPduId**	
Description	Unique ID for the Rx L-PDU in the CAN Interface. Range: 0..max. number of defined CanRxPduIds.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=65534 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfRxPduReadData**
Description	Enables and disables the Rx buffering for reading of Rx L-PDU data. ► True: Enabled ► False: Disabled **Optimization Effect:**

	► **RAM increase (config):** Enabling this parameter increases the RAM consumption of the module configuration. ► **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	**CanIfRxPduReadNotifyStatus**
Description	Enables and disables receive indication for each Rx L-PDU for reading its notification status. ► True: Enabled ► False: Disabled **Optimization Effect:** ► **RAM increase (config):** Enabling this parameter increases the RAM consumption of the module configuration. ► **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	**CanIfRxPduUserRxIndicationName**
Description	In order to make this configuration container capable for post-build configuraton it is not possible to use this paramter. Please use the configuration parameter of the same name in the container CanIfUpperLayerConfig relating to CanIfRxPduUserRxIndicationUL.
Multiplicity	1..1
Type	FUNCTION-NAME
Default value	

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfRxPduUserRxIndicationUL**	
Description	CanIfRxPduUserRxIndicationUL defines the upper layer (UL) module to which the indication of the successfully received CanRxPduId is routed via <User_-RxIndication>. <User_RxIndication> is invoked when the indication of the configured Can-RxPduId is received by a Rx indication event from the CAN Driver module. If no upper layer (UL) module is configured, no <User_RxIndication> is called in case of a Rx indication event of the CanRxPduId from the CAN Driver module.	
Multiplicity	0..1	
Type	ENUMERATION	
Default value	PDUR	
Range	CAN_NM CAN_TP CAN_TSYN J1939NM J1939TP CDD PDUR	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfRxPduBswSchExclAreaIdRef**	
Description	CanIfRxPduBswSchExclAreaIdRef is not used by the CanIf and therefore can not be edited.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfRxPduHrhIdRef**
Description	CanIfRxPduHrhIdRef refers to the HRH to which Rx L-PDU belongs to.

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

Parameter Name	**CanIfRxPduRef**	
Description	Reference to the "global" PDU structure to allow harmonization of handle IDs in the com stack.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfRxPduTargetPduID**
Label	CanIfRxPduTargetPduID
Description	CanIfRxPduTargetPduID defines the PDU ID to be delivered for this Rx L-PDU if this message is received on the bus. Please note that CanIfRxPduTargetPduID is used only if this Rx L-PDU is assigned to a user specific upper layer (parameter *CanIfRxPduUserRxIndicationUL* is set to *CDD*) and if CanIfUseCddHandleIds of its referenced UL Cdd is set to false.
Multiplicity	1..1
Type	INTEGER
Range	<=65535 >=0
Configuration class	**PostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfRxPduUpperLayerRef**
Label	CanIfRxPduUpperLayerRef
Description	CanIfRxPduUpperLayerRef references the user specific upper layer for this Rx L-PDU in the case that *CanIfRxPduUserRxIndicationUL* is set to *CDD*.
Multiplicity	1..1
Type	REFERENCE

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

5.2.1.12. CanIfRxPduCanIdRange

Parameters included	
Parameter name	**Multiplicity**
CanIfRxPduCanIdRangeLowerCanId	1..1
CanIfRxPduCanIdRangeUpperCanId	1..1

Parameter Name	**CanIfRxPduCanIdRangeLowerCanId**	
Description	Lower CAN identifier of a Rx L-PDU for identifier range definition, in which all CAN IDs are mapped to one PDU ID. Range: ► 11 bit for standard CAN identifier. ► 29 bit for extended CAN identifier.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfRxPduCanIdRangeUpperCanId**	
Description	Upper CAN identifier of a Rx L-PDU for identifier range definition, in which all CAN IDs are mapped to one PDU ID. Range: ► 11 bit for standard CAN identifier. ► 29 bit for extended CAN identifier.	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Configuration class	**VariantPostBuild:**	VariantPostBuild

Origin	AUTOSAR_ECUC

5.2.1.13. CanIfTTRxFrameTriggering

Parameters included	
Parameter name	**Multiplicity**
CanTTRxJoblistTimeMark	0..1
CanIfTTRxHwObjectTriggerIdRef	1..1

Parameter Name	**CanTTRxJoblistTimeMark**	
Description	CanTTRxJoblistTimeMark is not used by the CanIf and therefore can not be edited.	
Multiplicity	0..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfTTRxHwObjectTriggerIdRef**	
Description	CanIfTTRxHwObjectTriggerIdRef is not used by the CanIf and therefore can not be edited.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.14. CanIfTxPduCfg

Containers included		
Container name	**Multiplicity**	**Description**
CanIfTTTxFrameTriggering	1..1	CanIfTTTxFrameTriggering is not used by the CanIf.

Parameters included	
Parameter name	**Multiplicity**
CanIfTxPduCanId	0..1

Parameters included	
CanIfTxPduCanIdType	1..1
CanIfTxPduCanIdMask	1..1
CanIfTxPduDlc	1..1
CanIfTxPduId	1..1
CanIfTxPduPnFilterPdu	1..1
CanIfTxPduReadNotifyStatus	1..1
CanIfTxPduTruncation	1..1
CanIfTxPduTruncateToFrame	1..1
CanIfTxPduType	1..1
CanIfTxPduUserTxConfirmationName	1..1
CanIfTxPduUserTxConfirmationUL	0..1
CanIfTxPduBswSchExclAreaIdRef	1..1
CanIfTxPduSourcePduID	1..1
CanIfTxPduUpperLayerRef	1..1
CanIfTxPduBufferRef	1..1
CanIfTxPduRef	1..1

Parameter Name	CanIfTxPduCanId	
Description	CAN identifier of Tx L-PDUs used by the CAN Driver for CAN L-PDU transmission. Range: ► 11 bit for standard CAN identifier. ► 29 bit for extended CAN identifier.	
Multiplicity	0..1	
Type	INTEGER	
Default value	0	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanIfTxPduCanIdType
Description	CanIfTxPduCanIdType describes the type of the CAN identifier of Tx L-PDUs used by the CAN Driver for CAN L-PDU transmission.

	Range: ► STANDARD_CAN Can frame with standard identifier (11 bits). ► STANDARD_FD_CAN Can FD frame with standard identifier (11 bits). ► EXTENDED_CAN Can frame with extended identifier (29 bits). ► EXTENDED_FD_CAN Can FD frame with extended identifier (29 bits).
Multiplicity	1..1
Type	ENUMERATION
Default value	STANDARD_CAN
Range	EXTENDED_CAN EXTENDED_FD_CAN STANDARD_CAN STANDARD_FD_CAN
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfTxPduCanIdMask**
Description	Identifier mask which denotes relevant bits in the CAN Identifier. This parameter may be used to keep parts of the CAN Identifier of dynamic transmit L-PDUs static. Range: ► 11 bit for standard CAN identifier. ► 29 bit for extended CAN identifier.
Multiplicity	1..1
Type	INTEGER
Default value	3758096383
Range	<=3758096383 >=0
Configuration class	**PostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfTxPduDlc**
Description	CanIfTxPduDlc is not used by the CanIf and therefore can not be edited.

	Please note: The data length code is derived from the PduInfoPtr of the CanIf_-Transmit() function instead.
Multiplicity	1..1
Type	INTEGER
Default value	8
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfTxPduId**
Description	ECU wide unique, symbolic handle for Tx L-PDU. Range: 0..max. number of CanTxPduIds.
Multiplicity	1..1
Type	INTEGER
Range	<=65534 >=0
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfTxPduPnFilterPdu**
Description	If CanIfPublicPnFilterSupport is enabled, by this parameter PDUs could be configured which will pass the CanIfPnFilter. If there is no CanIfTxPduPnFilterPdu configured per controller, the corresponding controller applies no CanIfPnFilter.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfTxPduReadNotifyStatus**
Description	Enables and disables transmit confirmation for each Tx L-PDU for reading its notification status. ► True: Enabled ► False: Disabled **Optimization Effect:**

	► **RAM increase (config):** Enabling this parameter increases the RAM consumption of the module configuration. ► **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	**CanIfTxPduTruncation**
Description	Enables/disables truncation of PDUs that exceed the configured size.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfTxPduTruncateToFrame**
Description	Provides an option to truncate the TxPdu length before transmission either at the CAN frame length (8B for CAN 2.0, 64B for CAN FD) or at the configured PDU length in EcuC. ► True: Truncate the TxPdu to the CAN frame length: 8B for CAN 2.0, 64B for CAN FD (legacy behavior) ► False: Truncate the TxPdu to the configured PDU length in EcuC Please note that CanIfTxPduTruncateToFrame is used only if parameter CanIfTxPduTruncation is TRUE.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**PostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfTxPduType**

Description	CanIfTxPduType defines the type of each Tx L-PDU. CanIfTxPduType selects between static CAN ID or a possible CAN ID change via the API CanIf_SetDynamicTxId() Range: ► STATIC: CAN ID can not be changed during run-time ► DYNAMIC: CAN ID can be changed during run-time **Optimization Effect:** ► **RAM increase (config):** Setting this parameter to DYNAMIC increases the RAM consumption of the module configuration.
Multiplicity	1..1
Type	ENUMERATION
Default value	STATIC
Range	DYNAMIC STATIC
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfTxPduUserTxConfirmationName**
Description	In order to make this configuration container capable for post-build configuraton it is not possible to use this paramter. Please use the configuration parameter of the same name in the container CanIfUpperLayerConfig relating to CanIfTxPduUserTxConfirmationUL.
Multiplicity	1..1
Type	FUNCTION-NAME
Default value	
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfTxPduUserTxConfirmationUL**
Description	CanIfTxPduUserTxConfirmationUL defines the upper layer (UL) module to which the confirmation of the successfully transmitted CanTxPduId is routed via the <User_TxConfirmation>. <User_TxConfirmation> is invoked when the confirmation of the configured CanTxPduId was received by a Tx confirmation event from the CAN Driver mod-

	ule. If no upper layer (UL) module is configured, no <User_TxConfirmation> is called in case of a Tx confirmation event of the CanTxPduId from the CAN Driver module.
Multiplicity	0..1
Type	ENUMERATION
Default value	PDUR
Range	CAN_NM CAN_TP CAN_TSYN J1939NM J1939TP CDD PDUR
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfTxPduBswSchExclAreaIdRef**
Description	CanIfTxPduBswSchExclAreaIdRef is not used by the CanIf and therefore can not be edited.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfTxPduSourcePduID**
Label	CanIfTxPduSourcePduID
Description	CanIfTxPduSourcePduID defines the PDU ID to be used for Tx confirmations of this Tx L-PDU if this message successfully transmitted. Please note that CanIfTxPduSourcePduID is used only if this Tx L-PDU is assigned to a user specific upper layer (parameter *CanIfTxPduUserTxConfirmationUL* is set to *CDD*) and if CanIfUseCddHandleIds of its referenced UL Cdd is set to false.
Multiplicity	1..1
Type	INTEGER

Range	<=65535 >=0	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfTxPduUpperLayerRef**	
Label	CanIfTxPduUpperLayerRef	
Description	CanIfTxPduUpperLayerRef references the user specific upper layer for this Tx L-PDU in the case that *CanIfTxPduUserTxConfirmationUL* is set to *CDD*.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfTxPduBufferRef**	
Description	Configurable reference to a CanIf buffer configuration..	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfTxPduRef**	
Description	Reference to the "global" PDU structure to allow harmonization of handle IDs in the com stack.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.15. CanIfTTTxFrameTriggering

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
CanIfTTTxJoblistTimeMark	0..1
CanIfTTTxHwObjectTriggerIdRef	1..1

Parameter Name	CanIfTTTxJoblistTimeMark	
Description	CanIfTTTxJoblistTimeMark is not used by the CanIf and therefore can not be edited.	
Multiplicity	0..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanIfTTTxHwObjectTriggerIdRef	
Description	CanIfTTTxHwObjectTriggerIdRef is not used by the CanIf and therefore can not be edited.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.16. CanIfPrivateCfg

Containers included		
Container name	**Multiplicity**	**Description**
CanIfTTGeneral	1..1	CanIfTTGeneral is not used by the CanIf.

Parameters included	
Parameter name	**Multiplicity**
CanIfPrivateDlcCheck	1..1
CanIfPrivateSoftwareFilterType	1..1
CanIfSupportTTCAN	1..1

Parameter Name	CanIfPrivateDlcCheck

Description	Selects whether the DLC check is supported. ► True: Enabled ► False: Disabled **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	**CanIfPrivateSoftwareFilterType**	
Description	Selects the desired software filter mechanism for reception only. Each implemented software filtering method is identified by this enumeration number. Range: The types of implemented software filtering methods. Only BINARY is supported.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	BINARY	
Range	BINARY	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfSupportTTCAN**
Description	CanIfSupportTTCAN is not used by the CanIf and therefore can not be edited.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.17. CanIfTTGeneral

Parameters included	
Parameter name	**Multiplicity**
CanIfTTJoblist	1..1
CanIfTTMaxIsrDelay	1..1

Parameter Name	**CanIfTTJoblist**	
Description	CanIfTTJoblist is not used by the CanIf and therefore can not be edited.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfTTMaxIsrDelay**	
Description	CanIfTTMaxIsrDelay is not used by the CanIf and therefore can not be edited.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.18. CanIfPublicCfg

Containers included		
Container name	**Multiplicity**	**Description**
CanIfRxProcessing	0..n	Configuration of a dedicated Rx MainFunction. The name of the generated function uses the pattern `CanIf_-MainFunctionRx_-"Short-Name"`.

Containers included		
CanIfTxProcessing	0..n	Configuration of a dedicated Tx MainFunction. The name of the generated function uses the pattern `CanIf_-MainFunctionTx_-"Short-Name"`.

Parameters included	
Parameter name	**Multiplicity**
CanIfSetBaudrateApi	1..1
CanIfPublicCancelTransmitSupport	1..1
CanIfBusMirroringSupport	1..1
CanIfPublicCddHeaderFile	0..n
CanIfPublicChangeBaudrateSupport	1..1
CanIfPublicDevErrorDetect	1..1
CanIfPublicHandleTypeEnum	1..1
CanIfPublicMultipleDrvSupport	1..1
CanIfPublicNumberOfCanHwUnits	1..1
CanIfPublicPnSupport	1..1
CanIfPublicReadRxPduDataApi	1..1
CanIfPublicReadRxPduNotifyStatusApi	1..1
CanIfPublicReadTxPduNotifyStatusApi	1..1
CanIfPublicSetDynamicTxIdApi	1..1
CanIfMetaDataSupport	1..1
CanIfPublicTxBuffering	1..1
CanIfPublicTxConfirmPollingSupport	1..1
CanIfDetReportRuntimeError	1..1
CanIfDecoupledProcessingSupport	1..1
CanIfMultiCoreSupport	1..1
CanIfRelocatablePbcfgEnable	1..1
CanIfSoftwareFilteringSupport	1..1
CanIfCanDriverCompatibility	1..1
CanIfPublicCanIdTypeEnum	1..1
CanIfPublicSingleCtrlOpt	1..1
CanIfPublicHohTranslationOpt	1..1
CanIfPublicCtrlWakeupSupport	1..1

Parameters included	
CanIfPublicTrcvWakeupSupport	1..1
CanIfPublicRangeReceptionSupport	1..1
CanIfPublicTrcvSupport	1..1
CanIfPublicMaxCtrl	1..1
CanIfPublicMaxTxBuffers	1..1
CanIfPublicMaxTxBufferSize	1..1
CanIfPublicMaxTxPdus	1..1
CanIfPublicMaxHths	1..1
CanIfPublicMaxRxNotifyPdus	1..1
CanIfPublicMaxTxNotifyPdus	1..1
CanIfPublicMaxRxBuffer	1..1
CanIfPublicMaxDynTxPdus	1..1
CanIfSingleCanTrcvAPIInfixEnable	1..1
CanIfTxOfflineActiveSupport	1..1
CanIfHookOnReceptionSupport	1..1
CanIfValidateWakeupOnStartedCtrlOnly	1..1
CanIfPublicVersionInfoApi	1..1
CanIfPublicWakeupCheckValidSupport	1..1
CanIfPublicWakeupCheckValidByNM	1..1

Parameter Name	CanIfSetBaudrateApi	
Description	Configuration parameter to enable/disable the CanIf_SetBaudrate API to change the baud rate of a CAN Controller. ► True: CanIf_SetBaudrate API is supported. ► False: CanIf_SetBaudrate API is not supported.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CanIfPublicCancelTransmitSupport

Description	CanIfPublicCancelTransmitSupport enables/disables a dummy API for upper layer modules which allows to request the cancellation of an I-PDU. ► True: Enabled ► False: Disabled **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	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfBusMirroringSupport**
Description	CanIfBusMirroringSupport enables/disables frame mirroring support through the Mirror module. ► **True:** Enabled ► **False:** Disabled The parameter affects the availablility of the following APIs ► `CanIf_EnableBusMirroring` ► `CanIf_GetControllerErrorState` ► `CanIf_GetControllerTxErrorCounter` The parameter affects the signature of the function `CanIf_GetControllerMode`, the second parameter having the type: ► **True:** `Can_ControllerStateType*` ► **False:** `CanIf_ControllerModeType*` The parameter affects the signature of the function `CanIf_GetTrcvMode` as follows: ► **True:** `uint8 TransceiverId, CanTrcv_TrcvModeType * TransceiverModePtr`

	► **False:** `CanTrcv_TrcvModeType* TransceiverModePtr, uint8 TransceiverId` For additional parameters see CanIfMirroring. **Optimization Effect:** ► **ROM increase (code):** Enabling this parameter increases the ROM consumption of the module code. ► **RAM increase (config):** Enabling this parameter increases the RAM consumption of the module configuration. ► **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	**CanIfPublicCddHeaderFile**
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	**CanIfPublicChangeBaudrateSupport**
Description	Configuration parameter to enable/disable the API to change the baudrate of a CAN controller.. ► True: Enabled ► False: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfPublicDevErrorDetect**
Description	Enables and disables the development error detection and notification mechanism. ► True: Enabled ► False: Disabled **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	**CanIfPublicHandleTypeEnum**
Description	CanIfPublicHandleTypeEnum is used to configure the type Can_HwHandleType. The type Can_HwHandelType represents the hardware object handles of a CAN hardware unit. For CAN hardware units or in general systems with more than 254 hardware objects, the extended range (UINT16) shall be used. than **Optimization Effect:** ► **ROM reduction (config):** Setting this parameter to UINT8 reduces the ROM consumption of the module configuration.
Multiplicity	1..1
Type	ENUMERATION
Default value	UINT8
Range	UINT16 UINT8
Configuration class	**VariantPostBuild:** VariantPostBuild

Origin	AUTOSAR_ECUC

Parameter Name	**CanIfPublicMultipleDrvSupport**	
Description	Selects support for multiple CAN Drivers. ► True: Enabled ► False: Disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfPublicNumberOfCanHwUnits**	
Description	CanIfPublicNumberOfCanHwUnits is not used by the CanIf and therefore can not be edited. Please note: The CanIf uses the default value CanIfPublicNumberOfCanHwUnits = 1.	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfPublicPnSupport**	
Description	Selects support of Partial Network features in CanIf. ► True: Enabled ► False: Disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanIfPublicReadRxPduDataApi	
Description	Enables / Disables the API CanIf_ReadRxPduData() for reading Rx L-PDU data. ► True: Enabled ► False: Disabled **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	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanIfPublicReadRxPduNotifyStatusApi	
Description	Enables and disables the API CanIf_ReadRxNotifStatus() for reading the Rx L-PDU data. ► True: Enabled ► False: Disabled **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	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanIfPublicReadTxPduNotifyStatusApi

Description	Enables and disables the API CanIf_ReadTxNotifStatus() for reading the notification status of Tx L-PDUs. ► True: Enabled ► False: Disabled **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	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfPublicSetDynamicTxIdApi**
Description	Enables and disables the API CanIf_SetDynamicTxId() for reconfiguration of the CAN identifier for dynamic Tx L-PDUs. ► True: Enabled ► False: Disabled **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	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfMetaDataSupport**
Description	Enable support for dynamic ID handling using L-SDU MetaData.

	► True: Metadata support enabled. ► False: Metadata support disabled.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfPublicTxBuffering**
Description	Enables and disables the buffering of Tx L-PDUs within the CAN Interface module. ► True: Enabled ► False: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfPublicTxConfirmPollingSupport**
Description	Configuration parameter to enable/disable the API CanIf_GetTxConfirmationState() to poll for the Tx confirmation state.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanIfDetReportRuntimeError**
Description	Switches the **Runtime Error Reporting to Det** ON or OFF. ► **TRUE**: `CANIF_E_TXPDU_LENGTH_EXCEEDED` is reported to Det ► **FALSE**: `CANIF_E_TXPDU_LENGTH_EXCEEDED` 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	**CanIfDecoupledProcessingSupport**
Description	Enable/disable the assignment of PDUs to selected MainFunctions for processing. ► True: CanIf processes the reception indication and the transmission confirmation events in context of MainFunction. ► False: CanIf processes the reception indication and the transmission confirmation events in ISR context (as defined by the SWS).
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfMultiCoreSupport**
Description	Enable/disable the Multicore support. ► True: Different CAN channels, together with its corresponding resources (controllers, HOHs, PDUs, MainFunctions, etc.), can be allocated on different cores/partitions. ► False: All CAN channels, together with its corresponding resources (controllers, HOHs, PDUs, MainFunctions, etc.), share the same core/partition.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild

Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfRelocatablePbcfgEnable**	
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	**CanIfSoftwareFilteringSupport**
Description	Enables/disable support for Software filtering. True: Software filtering is enabled. False: Software filtering is disabled (only if all HRHs have parameter "CanIfHrhSoftwareFilter" set to OFF and no Rx Pdu has parameter "CanIfRxPduCanIdMask" enabled). **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	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfCanDriverCompatibility**
Description	Specifies with which AUTOSAR Revision of the CAN driver, CanIf shall be compatible with. ► **ASR402**: CanIf behaves as specified in AUTOSAR 4.0 Rev.2 in regards to the Can module.

	► **ASR403**: CanIf behaves as specified in AUTOSAR 4.0 Rev.3 in regards to the Can module. ► **ASR422**: CanIf behaves as specified in AUTOSAR 4.2 Rev.2 in regards to the Can module. ► **ASR431**: CanIf behaves as specified in AUTOSAR 4.3 Rev.1 in regards to the Can module. ► **ASR440**: CanIf behaves as specified in AUTOSAR 4.4 Rev.0 in regards to the Can module.
Multiplicity	1..1
Type	ENUMERATION
Default value	ASR403
Range	ASR402 ASR403 ASR422 ASR431 ASR440
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfPublicCanIdTypeEnum**
Label	CanIfPublicCanIdTypeEnum
Description	CanIfPublicCanIdTypeEnum is used to configure the data type Can_IdType. The type Can_IdType represents the CAN identifiers (IDs) used on the bus. If any extended CAN IDs are used, this parameter must be set to *UINT32*.
Multiplicity	1..1
Type	ENUMERATION
Default value	UINT32
Range	UINT16 UINT32
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfPublicSingleCtrlOpt**
Description	Optimization for using only a single Can controller.

	Optimization Effect: ► **ROM reduction (code):** Enabling this parameter reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Enabling this parameter reduces 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	**CanIfPublicHohTranslationOpt**
Description	Optimization for a certain ordering of HOH CanObjedctIds. This optimization can be enabled, if the following criterias for CanObjectIds are met: ► All HRH CanObjecdtIds must be smaller than any of the HTH CanObjectId. ► All CanObjecdtIds must be 0-based and dense. **Optimization Effect:** ► **ROM reduction (code):** Enabling this parameter reduces the ROM consumption of the module code. ► **ROM reduction (config):** Enabling this parameter reduces the ROM consumption of the module configuration. ► **Execution time reduction (code):** Enabling this parameter reduces 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	**CanIfPublicCtrlWakeupSupport**
Description	Enables wakeup detection via Can controllers. **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
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfPublicTrcvWakeupSupport**
Description	Enables wakeup detection via Can transceivers. **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	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfPublicRangeReceptionSupport**
Description	Enables reception of CanId ranges using a single HRH. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **ROM reduction (config):** Disabling this parameter reduces the ROM consumption of the module configuration. ► **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	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfPublicTrcvSupport**
Description	Enables support of Can transceivers. Disabling this parameter disables the following interfaces: ► `CanTrcv_GetOpMode` ► `CanTrcV_SetOpMode` **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code.
Multiplicity	1..1
Type	BOOLEAN
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfPublicMaxCtrl**
Description	Maximum number of Can controller supported. This configuration parameter allocates runtime memory per Can controller.
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfPublicMaxTxBuffers**
Description	Maximum number of TxBuffers supported. This configuration parameter allocates runtime memory per Tx Buffer. Please note, that the available buffer size is defined via parameter CanIfPublicMaxTxBufferSize.
Multiplicity	1..1
Type	INTEGER
Default value	64
Range	<=65535

	>=1	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfPublicMaxTxBufferSize**	
Description	Maximum shared memory in bytes reserved for all TxBuffers. This configuration parameter allocates runtime memory.	
Multiplicity	1..1	
Type	INTEGER	
Default value	512	
Range	<=65535 >=1	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfPublicMaxTxPdus**	
Description	Maximum number of Tx Pdus across all controllers and variants. This configuration parameter allocates runtime memory.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfPublicMaxHths**	
Description	Maximum number of Hths supported. This configuration parameter allocates runtime memory per Hth.	
Multiplicity	1..1	
Type	INTEGER	
Default value	16	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfPublicMaxRxNotifyPdus**

Description	Maximum number of Rx-Pdus which support notification functions. This configuration parameter allocates runtime memory per notfied Rx-Pdu.
Multiplicity	1..1
Type	INTEGER
Default value	64
Range	<=65535 >=1
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfPublicMaxTxNotifyPdus**
Description	Maximum number of Tx-Pdus which support notification functions. This configuration parameter allocates runtime memory per notfied Tx-Pdu.
Multiplicity	1..1
Type	INTEGER
Default value	64
Range	<=65535 >=1
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfPublicMaxRxBuffer**
Description	Maximum size of Rx-Buffer allocated in bytes. The Rx-Buffer is required only, if the CanIf_ReadRxPduData() Api function is used.
Multiplicity	1..1
Type	INTEGER
Default value	72
Range	<=65535 >=1
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfPublicMaxDynTxPdus**

Description	Maximum number of dynamic Tx Pdus supported.	
Multiplicity	1..1	
Type	INTEGER	
Default value	8	
Range	<=65535 >=1	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfSingleCanTrcvAPIInfixEnable**	
Description	This parameter defines if CanIf shall use the Vendor Id and the API Infix for accessing the CanTrcv module in case a single CanTrcv driver is configured. true: CanIf uses the Vendor Id and the API Infix of the CanTrcv for accessing the CanTrcv API (e.g. CanTrcv_1_T01_SetOpMode) in case only a single CanTrcv driver is used. In addtion this name mangling is also used for including the CanTrcv header file (e.g. CanTrcv_1_T01.h) false: CanIf does not use the Vendor Id and the API Infix of the CanTrcv in case only a single CanTrcv driver is used. Note: If more than one CanTrcv 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	**CanIfTxOfflineActiveSupport**
Description	Determines wether TxOfflineActive feature is supported by CanIf. True: Enabled False: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfHookOnReceptionSupport**
Description	Enable/disable reporting a successful reception to a CDD module. The contained information reported will be CAN_ID + data + received CAN index. ► True: CanIf reports a successful reception to a CDD. ► False: CanIf does not report a successful reception to a CDD.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfValidateWakeupOnStartedCtrlOnly**
Description	Enable/disable the wakeup validation based on the corresponding controller mode. True: The wakeup validation is performed only when the corresponding controller is in CAN_CS_STARTED mode (handling according to ASR 4.3.1 or later). False: The wakeup validation is performed only when the controller is in CANIF_CS_SLEEP mode or when the wakeup flag was previously set.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfPublicVersionInfoApi**
Description	Enables and disables the API CanIf_GetVersionInfo() for reading the version information about the CAN Interface. **Optimization Effect:** ► **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	AUTOSAR_ECUC	

Parameter Name	**CanIfPublicWakeupCheckValidSupport**	
Description	Selects support for wake up validation. ► True: Enabled ► False: Disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfPublicWakeupCheckValidByNM**	
Description	If enabled, only NM messages shall validate a detected wake-up event (see CANIF722) at the corresponding wake-up source in the CanIf. If disabled, all messages shall validate such a wake-up event. ► True: Enabled ► False: Disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.19. CanIfRxProcessing

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
CanIfPartitionRef	1..1
CanIfRxPduProcessingRef	1..n
CanIfRxPduQueueSize	1..1
CanIfPublicMaxPayloadQueueSize	1..1

Parameter Name	CanIfPartitionRef	
Description	Reference to EcucPartition to allow grouping of MainFunction according to EcucPartition elements.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CanIfRxPduProcessingRef	
Description	Reference to CanIfRxPduCfg which is assigned to this MainFunction.	
Multiplicity	1..n	
Type	REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CanIfRxPduQueueSize	
Description	CanIfRxPduQueueSize defines the local queue for handling of RxPdus.	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Range	<=65535 >=1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CanIfPublicMaxPayloadQueueSize
Description	Maximum shared memory in bytes reserved for all RxQueues. This configuration parameter allocates runtime memory.

	The grand total of CanIfPublicMaxPayloadQueueSize of all the CanIfRxProcessing containers can not exceed 65535.	
Multiplicity	1..1	
Type	INTEGER	
Default value	512	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.2.1.20. CanIfTxProcessing

Parameters included	
Parameter name	**Multiplicity**
CanIfPartitionRef	1..1
CanIfTxPduProcessingRef	1..n
CanIfTxPduQueueSize	1..1

Parameter Name	**CanIfPartitionRef**	
Description	Reference to EcucPartition to allow grouping of MainFunction according to EcucPartition elements.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfTxPduProcessingRef**	
Description	Reference to CanIfTxPduCfg which is assigned to this MainFunction.	
Multiplicity	1..n	
Type	CHOICE-REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfTxPduQueueSize**
Description	CanIfTxPduQueueSize defines the local queue size for handling of TxPdus.
Multiplicity	1..1
Type	INTEGER

Default value	1	
Range	<=65535 >=1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.2.1.21. CanIfTrcvDrvCfg

Containers included		
Container name	**Multiplicity**	**Description**
CanIfTrcvCfg	1..255	CanIfTrcvCfg contains the configuration parameters of one addressed CAN transceiver by the underlying CAN Transceiver Driver module. For each CAN transceiver a seperate instance of this container has to be provided.

5.2.1.22. CanIfTrcvCfg

Parameters included	
Parameter name	**Multiplicity**
CanIfTrcvId	1..1
CanIfTrcvWakeupSupport	1..1
CanIfTrcvCanTrcvRef	1..1
CanIfTrcvWakeupSourceInRef	0..1
CanIfTrcvWakeupSourceOutRef	0..1

Parameter Name	**CanIfTrcvId**
Description	CanIfTrcvId abstracts from the CAN Transceiver Driver specific parameter transceiver. Each transceiver of all connected CAN Transceiver Driver modules shall be assigned to one specific TransceiverId of the CanIf. Range: 0..number of configured transceivers of all CAN Transceiver Driver modules.
Multiplicity	1..1

Type	INTEGER	
Range	<=254 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfTrcvWakeupSupport**	
Description	CanIfTrcvWakeupSupport defines if a respective transceiver of the referenced CAN Transceiver Driver modules is queriable for wake up events. CanIfTrcvWakeupSupport can only be set to true, if the CAN transceiver, which is referenced by parameter CanIfTrcvCanTrcvRef, also supports wakeup, i.e. parameter CanTrcvWakeupByBusUsed is set to true. ► True: Wakeup Support Enabled ► False: Wakeup Support Disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfTrcvCanTrcvRef**	
Description	CanIfTrcvCanTrcvRef references exactly one CAN Transceiver (transceiver channel) of an underlying CAN Transceiver Driver module. If using multiple CAN Transceivers of the same CAN Transceiver Driver module, each reference located in the same CanIfTrcvDrvCfg container must point to the same underlying CAN Transceiver Driver module (but to different CAN Transceivers).	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfTrcvWakeupSourceInRef**
Description	CanIfTrcvWakeupSourceInRef contains a reference to the wake up source for the transceiver as defined in the ECU State Manager.

	CanIfTrcvWakeupSourceInRef allows mapping of incoming wake up source (i.-e. a wake up event is detected by the CanTrcv Driver) to a specific transceiver. Therefore the parameter is used as input to API CanIf_CheckWakeup(). The parameter is optional since providing the API CanIf_CheckWakeup() is also optional. If the parameter is used, CanIfTrcvWakeupSourceOutRef must be set as well. Implementation Type: reference to EcuM_WakeupSourceType.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanIfTrcvWakeupSourceOutRef**	
Description	CanIfTrcvWakeupSourceOutRef contains a reference to the wake up source for the transceiver as defined in the ECU State Manager. CanIfTrcvWakeupSourceOutRef allows mapping of outgoing wake up source (i.e. wake up is signalled to the user notification API) to a specific transceiv-er. Therefore the parameter is used as output from CanIf_CheckWakeup() to <User_SetWakeupEvent>. The parameter is also used for wake up valida-tion, i.e. as input and output for API CanIf_CheckValidation(). The parameter is optional since providing the API CanIf_CheckWakeup() is also optional. Implementation Type: reference to EcuM_WakeupSourceType.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.23. CanIfDecoupledMeasurementSupport

Parameters included	
Parameter name	**Multiplicity**
CanIfRxDecoupledMeasurementSupport	1..1
CanIfTxDecoupledMeasurementSupport	1..1

Parameters included	
CanIfNumberOfRxPdusExceedingQueueApiName	1..1
CanIfNumberOfEnqueuedRxPdusApiName	1..1
CanIfNumberOfTxPdusExceedingQueueApiName	1..1
CanIfNumberOfEnqueuedTxPdusApiName	1..1

Parameter Name	CanIfRxDecoupledMeasurementSupport	
Description	Enable/disable the measurement support for decoupled processing of receive events. CanIf will forward to the CDD module through CanIfNumberOfEnqueue-dRxPdusApiName the number of queued receive indications events in context of MainFunction.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CanIfTxDecoupledMeasurementSupport	
Description	Enable/disable the measurement support for decoupled processing of transmit confirmation events. CanIf will forward to the CDD module through CanIfNum-berOfEnqueuedTxPdusApiName the number of queued transmit confirmations events in context of MainFunction.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CanIfNumberOfRxPdusExceedingQueueApiName
Description	CanIfNumberOfRxPdusExceedingQueueApiName defines the name of the CDD API called by the MainFunction to report the number of Rx Pdus that could not be added in the queue due to full queue. CanIfNum-berOfRxPdusExceedingQueueApiName is enabled when the parameter CanIfRxDecoupledMeasurementSupport is set to true. API signature: void <CanIfNumberOfRxPdusExceedingQueueApiName>(uint8 MainFuncId, uint16 NoOfPdus);

Multiplicity	1..1	
Type	FUNCTION-NAME	
Default value		
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfNumberOfEnqueuedRxPdusApiName**	
Description	CanIfNumberOfEnqueuedRxPdusApiName defines the name of the CDD API called by the MainFunction to report the number of Rx Pdus added in the queue. CanIfNumberOfEnqueuedRxPdusApiName is enabled when the parameter CanIfRxDecoupledMeasurementSupport is set to true. API signature: void <CanIfNumberOfEnqueuedRxPdusApiName>(uint8 MainFuncId, uint16 NoOfPdus);	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Default value		
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfNumberOfTxPdusExceedingQueueApiName**	
Description	CanIfNumberOfTxPdusExceedingQueueApiName defines the name of the CDD API called by the MainFunction to report the number of Tx Pdus that could not be added in the queue due to full queue. CanIfNumberOfTxPdusExceedingQueueApiName is enabled when the parameter CanIfTxDecoupledMeasurementSupport is set to true. API signature: void <CanIfNumberOfTxPdusExceedingQueueApiName>(uint8 MainFuncId, uint16 NoOfPdus);	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Default value		
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfNumberOfEnqueuedTxPdusApiName**
Description	CanIfNumberOfEnqueuedTxPdusApiName defines the name of the CDD API called by the MainFunction to report the number of Tx Pdus added in the queue.

	CanIfNumberOfEnqueuedTxPdusApiName is enabled when the parameter CanIfTxDecoupledMeasurementSupport is set to true. API signature: void <CanIfNumberOfEnqueuedTxPdusApiName>(uint8 MainFuncId, uint16 NoOfPdus);	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Default value		
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.2.1.24. CanIfUpperLayerConfig

Parameters included	
Parameter name	**Multiplicity**
CanIfUpperLayerUseCanId	1..1
CanIfPublicTxConfResultSupport	1..1
CanIfTxPduUserTxConfirmationName	0..1
CanIfRxPduUserRxIndicationName	0..1
CanIfUserDlcErrorNotification	0..1
CanIfUserDlcPassedNotification	0..1
CanIfUseCddHandleIds	1..1

Parameter Name	**CanIfUpperLayerUseCanId**
Label	CanIfUpperLayerUseCanId
Description	CanIfUpperLayerUseCanId defines if the signature of the API function <UL>_RxIndication contains the CAN ID as additional argument. ► True: Signature of <UL>_RxIndication changes to <UL>_RxIndication(PduIdType, PduInfoType*, Can_IdType). ► False: Signature of <UL>_RxIndication is conform to AUTOSAR. For the well-known upper layers of CanIf (PduR, CanTp, CanTSyn, CanNm, J1939Nm or J1939Tp) this parameter must be set to False.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

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

Parameter Name	**CanIfPublicTxConfResultSupport**	
Description	Determines whether the Tx confirmation shall be sent in case of successful transmission only or the Tx confirmation shall be sent in failure too reporting the result (E_OK/E_NOT_OK) Currently, this configuration is for J1939, CanTSyn and CDD modules only. True : Send the result as a parameter (E_OK/E_NOT_OK) False: Tx Confirmation in case of successful transmission only	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfTxPduUserTxConfirmationName**
Description	CanIfTxPduUserTxConfirmationName defines the name of the <User_-TxConfirmation>. CanIfTxPduUserTxConfirmationName depends on the parameter CanIfTxP-duUserTxConfirmationUL. If CanIfTxPduUserTxConfirmationUL is configured to a standard AUTOSAR upper layer (CanTp, CanTSyn, CanNm, J1939Nm, J1939Tp and PduR only), the name of the Tx confirmation function <User_-TxConfirmation> is fixed: ► CAN_TP: CanTp_TxConfirmation ► CAN_TSYN: CanTSyn_TxConfirmation ► CAN_NM: CanNm_TxConfirmation ► J1939NM: J1939Nm_TxConfirmation ► J1939TP: J1939Tp_TxConfirmation ► PDUR: PduR_CanIfTxConfirmation If CanIfTxPduUserTxConfirmationUL equals CDD, the name of the <User_-TxConfirmation> is selectable. If CanIfTxPduUserTxConfirmationName is disabled there are no Tx-Confirma-tions for this upper layer.

Multiplicity	0..1	
Type	FUNCTION-NAME	
Default value		
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanIfRxPduUserRxIndicationName**
Description	CanIfRxPduUserRxIndicationName defines the name of the <User_-RxIndication>. CanIfRxPduUserRxIndicationName depends on the parameter CanIfRxPdu-UserRxIndicationUL. If CanIfRxPduUserRxIndicationUL is configured to a standard AUTOSAR upper layer, the name of the <User_RxIndication> is fixed: ► CAN_TP: CanTp_RxIndication ► CAN_TSYN: CanTSyn_RxIndication ► CAN_NM: CanNm_RxIndication ► J1939NM: J1939Nm_RxIndication ► J1939TP: J1939Tp_RxIndication ► PDUR: PduR_CanIfRxIndication If CanIfRxPduUserRxIndicationUL equals CDD, the name of the <User_-RxIndication> is selectable. If CanIfRxPduUserRxIndicationName is disabled there are no Rx-Indications for this upper layer.
Multiplicity	0..1
Type	FUNCTION-NAME
Default value	
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfUserDlcErrorNotification**
Label	CanIfUserDlcErrorNotification
Description	Name of target user specific DLC check failed notification service. If CanIfUserDlcErrorNotification is disabled no call-out function will be called for this upper layer.

	To use the CanIfUserDlcErrorNotification a function prototype must exists in one of the header files pointed by CanIfPublicCddHeaderFile. **Optimization Effect:** ► **ROM increase (code):** Using this parameter increases the ROM consumption of the module code. ► **Execution time increase (code):** Using this parameter increases the execution time of the module code.
Multiplicity	0..1
Type	FUNCTION-NAME
Default value	
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfUserDlcPassedNotification**
Label	CanIfUserDlcPassedNotification
Description	Name of target user specific DLC check passed notification service. If CanIfUserDlcPassedNotification is disabled no call-out function will be called for this upper layer. To use the CanIfUserDlcErrorNotification a function prototype must exists in one of the header files pointed by CanIfPublicCddHeaderFile. **Optimization Effect:** ► **ROM increase (code):** Using this parameter increases the ROM consumption of the module code. ► **Execution time increase (code):** Using this parameter increases the execution time of the module code.
Multiplicity	0..1
Type	FUNCTION-NAME
Default value	
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanIfUseCddHandleIds**
Description	Determines whether for a Pdu that has a Cdd configured as an upper layer, it shall use the handle id taken from the Cdd entry that references that

	Pdu (CDD/CddComStackContribution/CddComIfUpperLayerContribution/CddComIfUpperLayerRx(or Tx)Pdu/CddComIfHandleId) or if it shall use CanIfTxPduSourcePduID/CanIfRxPduTargetPduID as a handle id. True: Use Cdd handle id False: Use CanIfTxPduSourcePduID/CanIfRxPduTargetPduID
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.2.1.25. CanIfMirroringSupport

Parameters included	
Parameter name	**Multiplicity**
CanIfMirrorToCDDReportingEnable	1..1
CanIfMirrorToCDDReportingFunctionName	1..1
CanIfMirrorToCDDReportingHeader	1..1
CanIfCanTxErrorCounterSupported	1..1
CanIfTxErrorCounterValue	1..1
CanIfCanControllerErrorStateSupported	1..1
CanIfErrorStateValue	1..1
TxMirrorBufferSize	1..1
TxMirrorNumTxPdus	1..1

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

Parameter Name	**CanIfMirrorToCDDReportingHeader**	
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	**CanIfCanTxErrorCounterSupported**
Label	Can Driver Tx Error Counter API Supported
Description	The parameter specifies whether the Can Driver exposes the API `Can_GetControllerTxErrorCounter` (see SWS_Can_00516). ► True: API is **available**. `CanIf_GetControllerTxErrorCounter` forwards the call to `Can_GetControllerTxErrorCounter`. ► False: API is **unavailable**. `CanIf_GetControllerTxErrorCounter` sets the parameter `TxErrorCounterPtr` to the value specified for the configuration parameter CanIfTxErrorCounterValue. The return value is always **E_OK**.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	CanIfTxErrorCounterValue	
Label	Tx Error Counter Value	
Description	The parameter specifies the value to which `TxErrorCounterPtr` is set when the API `CanIf_GetControllerTxErrorCounter` is called. Note: **E_OK** is returned.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=255 >=0	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CanIfCanControllerErrorStateSupported	
Label	Can Driver Controller Error State API Supported	
Description	The parameter specifies whether the Can Driver exposes the API `Can_GetControllerErrorState` (see SWS_Can_91004). ► True: API is **available**. `CanIf_GetControllerErrorState` forwards the call to `Can_GetControllerErrorState`. ► False: API is **unavailable**. `CanIf_GetControllerErrorState` sets the parameter `ErrorStatePtr` to the value specified for the configuration parameter CanIfErrorStateValue. The return value is always **E_OK**.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CanIfErrorStateValue
Label	Error State Value
Description	The parameter specifies the value to which `ErrorStatePtr` is set when the API `CanIf_GetControllerErrorState` is called. Note: **E_OK** is returned.
Multiplicity	1..1

Type	ENUMERATION	
Default value	CAN_ERRORSTATE_ACTIVE	
Range	CAN_ERRORSTATE_ACTIVE	
	CAN_ERRORSTATE_PASSIVE	
	CAN_ERRORSTATE_BUSOFF	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TxMirrorBufferSize**	
Label	Tx Mirror Buffer Size	
Description	The parameter specifies in bytes the size of the Tx buffer used when mirroring.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=0xFFFFFFFF	
	>=0x0	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**TxMirrorNumTxPdus**	
Label	Tx Mirror Max TxPdus	
Description	Maximum number CanIfTxPdus for Mirroring.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=65535	
	>=0	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.2.1.26. CanIfHookOnRxSupport

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
CanIfHookOnReceptionHeader	1..1
CanIfHookOnReceptionFunctionName	1..1

Parameter Name	CanIfHookOnReceptionHeader	
Description	Defines the header file name which contains the extern declaration of the custom hook function to be called on a successful reception. Range of characters is 1.. 32. Example: Cdd.h	
Multiplicity	1..1	
Type	STRING	
Configuration class	**PreCompile:**	VariantPostBuild

Parameter Name	CanIfHookOnReceptionFunctionName	
Description	Defines the CDD function name called in case of reporting a successful reception to a CDD. Example: Cdd_HookOnReception The CDD function prototype should be void <funcName>(Can_IdType CanId, PduInfoType * PduInfo, uint8 ControllerId)	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Configuration class	**PreCompile:**	VariantPostBuild

5.2.1.27. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1

Parameters included	
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	5
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	6
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	10
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	19
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	60
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

5.2.1.28. PublishedInformation

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

Parameter Name	**PbcfgMSupport**
Label	PbcfgM support
Description	Specifies whether or not the CanIf 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. Recommended configurations

5.2.2.1. CanIfRecConfigurationDflt

Containers included	
Container name	**Container definition**
CAN_NM	CanIfUpperLayerConfig
CAN_TP	CanIfUpperLayerConfig
CAN_TSYN	CanIfUpperLayerConfig
PDUR	CanIfUpperLayerConfig
J1939NM	CanIfUpperLayerConfig
J1939TP	CanIfUpperLayerConfig

Parameters included	
Parameter name	**Value**

5.2.2.1.1. CAN_NM

Parameters included	
Parameter name	**Value**
CanIfUpperLayerUseCanId	false
CanIfTxPduUserTxConfirmationName	CanNm_TxConfirmation
CanIfRxPduUserRxIndicationName	CanNm_RxIndication
CanIfUserDlcErrorNotification	**(DISABLED)**

Parameters included	
CanIfUserDlcPassedNotification	**(DISABLED)**

5.2.2.1.2. CAN_TP

Parameters included	
Parameter name	**Value**
CanIfUpperLayerUseCanId	false
CanIfTxPduUserTxConfirmationName	CanTp_TxConfirmation
CanIfRxPduUserRxIndicationName	CanTp_RxIndication
CanIfUserDlcErrorNotification	**(DISABLED)**
CanIfUserDlcPassedNotification	**(DISABLED)**

5.2.2.1.3. CAN_TSYN

Parameters included	
Parameter name	**Value**
CanIfUpperLayerUseCanId	false
CanIfTxPduUserTxConfirmationName	CanTSyn_TxConfirmation
CanIfRxPduUserRxIndicationName	CanTSyn_RxIndication
CanIfUserDlcErrorNotification	**(DISABLED)**
CanIfUserDlcPassedNotification	**(DISABLED)**

5.2.2.1.4. PDUR

Parameters included	
Parameter name	**Value**
CanIfUpperLayerUseCanId	false
CanIfTxPduUserTxConfirmationName	PduR_CanIfTxConfirmation
CanIfRxPduUserRxIndicationName	PduR_CanIfRxIndication
CanIfUserDlcErrorNotification	**(DISABLED)**
CanIfUserDlcPassedNotification	**(DISABLED)**

5.2.2.1.5. J1939NM

Parameters included	
Parameter name	**Value**
CanIfUpperLayerUseCanId	false
CanIfTxPduUserTxConfirmationName	J1939Nm_TxConfirmation
CanIfRxPduUserRxIndicationName	J1939Nm_RxIndication
CanIfUserDlcErrorNotification	**(DISABLED)**
CanIfUserDlcPassedNotification	**(DISABLED)**

5.2.2.1.6. J1939TP

Parameters included	
Parameter name	**Value**
CanIfUpperLayerUseCanId	false
CanIfTxPduUserTxConfirmationName	J1939Tp_TxConfirmation
CanIfRxPduUserRxIndicationName	J1939Tp_RxIndication
CanIfUserDlcErrorNotification	**(DISABLED)**
CanIfUserDlcPassedNotification	**(DISABLED)**

5.2.3. Application programming interface (API)

5.2.3.1. Type definitions

5.2.3.1.1. CanIf_ConfigType

Purpose	Type for the CAN interface configuration.
Type	struct
Description	This type defines the global configuration of the CAN interface. Please note that internal types are necessary as elements which are also published by the header files.

5.2.3.2. Macro constants

5.2.3.2.1. CANIF_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
Value	4U

5.2.3.2.2. CANIF_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
Value	0U

5.2.3.2.3. CANIF_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.
Value	3U

5.2.3.2.4. CANIF_E_INVALID_RXPDUID

Purpose	CANIF_E_INVALID_RXPDUID.
Value	60U

5.2.3.2.5. CANIF_E_INVALID_TXPDUID

Purpose	CANIF_E_INVALID_TXPDUID.
Value	50U

5.2.3.2.6. CANIF_E_NOK_NOSUPPORT

Purpose	CANIF_E_NOK_NOSUPPORT.
Value	40U

5.2.3.2.7. CANIF_E_PARAM_CANID

Purpose	CANIF_E_PARAM_CANID.
Value	10U

5.2.3.2.8. CANIF_E_PARAM_CONTROLLER

Purpose	CANIF_E_PARAM_CONTROLLER.
Value	14U

5.2.3.2.9. CANIF_E_PARAM_CONTROLLERID

Purpose	CANIF_E_PARAM_CONTROLLERID.
Value	15U

5.2.3.2.10. CANIF_E_PARAM_CTRLMODE

Purpose	CANIF_E_PARAM_CTRLMODE.
Value	21U

5.2.3.2.11. CANIF_E_PARAM_DLC

Purpose	CANIF_E_PARAM_DLC.
Value	11U

5.2.3.2.12. CANIF_E_PARAM_HRH

Purpose	CANIF_E_PARAM_HRH.
Value	12U

5.2.3.2.13. CANIF_E_PARAM_LPDU

Purpose	CANIF_E_PARAM_LPDU.

Value	13U

5.2.3.2.14. CANIF_E_PARAM_PDU_MODE

Purpose	CANIF_E_PARAM_PDU_MODE.
Value	22U

5.2.3.2.15. CANIF_E_PARAM_POINTER

Purpose	CANIF_E_PARAM_POINTER.
Value	20U

5.2.3.2.16. CANIF_E_PARAM_TRCV

Purpose	CANIF_E_PARAM_TRCV.
Value	17U

5.2.3.2.17. CANIF_E_PARAM_TRCVMODE

Purpose	CANIF_E_PARAM_TRCVMODE.
Value	18U

5.2.3.2.18. CANIF_E_PARAM_TRCVWAKEUPMODE

Purpose	CANIF_E_PARAM_TRCVWAKEUPMODE.
Value	19U

5.2.3.2.19. CANIF_E_PARAM_WAKEUPSOURCE

Purpose	CANIF_E_PARAM_WAKEUPSOURCE.
Value	16U

5.2.3.2.20. CANIF_E_TXPDU_LENGTH_EXCEEDED

Purpose	CANIF_E_TXPDU_LENGTH_EXCEEDED.
Value	90U

5.2.3.2.21. CANIF_E_UNINIT

Purpose	CANIF_E_UNINIT.
Value	30U

5.2.3.2.22. CANIF_MODULE_ID

Purpose	AUTOSAR module identification.
Value	60U

5.2.3.2.23. CANIF_SID_CANCELTRANSMIT

Purpose	CanIf_CancelTransmit() service ID.
Value	0x18U

5.2.3.2.24. CANIF_SID_CANCELTXCONFIRMATION

Purpose	CanIf_CancelTxConfirmation() service ID.
Value	0x15U

5.2.3.2.25. CANIF_SID_CHECKTRCVWAKEFLAG

Purpose	CanIf_CheckTrcvWakeFlag() service ID.
Value	0x1fU

5.2.3.2.26. CANIF_SID_CHECKTRCVWAKEFLAGIND

Purpose	CanIf_CheckTrcvWakeFlagIndication() service ID.

Value	0x21U

5.2.3.2.27. CANIF_SID_CHECKVALIDATION

Purpose	CanIf_CheckValidation() service ID.
Value	0x12U

5.2.3.2.28. CANIF_SID_CHECKWAKEUP

Purpose	CanIf_CheckWakeup() service ID.
Value	0x11U

5.2.3.2.29. CANIF_SID_CLEARTRCVWUFFLAG

Purpose	CanIf_ClearTrcvWufFlag() service ID.
Value	0x1eU

5.2.3.2.30. CANIF_SID_CLEARTRCVWUFFLAGIND

Purpose	CanIf_ClearTrcvWufFlagIndication() service ID.
Value	0x20U

5.2.3.2.31. CANIF_SID_CONFIRMPNAVAILABILITY

Purpose	CanIf_ConfirmPnAvailability() service ID.
Value	0x1aU

5.2.3.2.32. CANIF_SID_CONTROLLERBUSOFF

Purpose	CanIf_ControllerBusOff() service ID.
Value	0x16U

5.2.3.2.33. CANIF_SID_CONTROLLERMODEIND

Purpose	CanIf_ControllerModeIndication() service ID.
Value	0x17U

5.2.3.2.34. CANIF_SID_ENABLEBUSMIRRORING

Purpose	
Value	0x4cU

5.2.3.2.35. CANIF_SID_GETCONTROLLERERRORSTATE

Purpose	
Value	0x4bU

5.2.3.2.36. CANIF_SID_GETCONTROLLERMODE

Purpose	CanIf_GetControllerMode() service ID.
Value	0x04U

5.2.3.2.37. CANIF_SID_GETCONTROLLERTXERRORCOUNTER

Purpose	
Value	0x4eU

5.2.3.2.38. CANIF_SID_GETPDUMODE

Purpose	CanIf_GetPduMode() service ID.
Value	0x0aU

5.2.3.2.39. CANIF_SID_GETTRANSCEIVERMODE

Purpose	CanIf_GetTrcvMode() service ID.

Value	0x0eU

5.2.3.2.40. CANIF_SID_GETTRCVWAKEUPREASON

Purpose	CanIf_GetTrcvWakeupReason() service ID.
Value	0x0fU

5.2.3.2.41. CANIF_SID_GETTXCONFIRMSTATE

Purpose	CanIf_GetTxConfirmationState() service ID.
Value	0x19U

5.2.3.2.42. CANIF_SID_GETVERSIONINFO

Purpose	CanIf_GetVersionInfo() service ID.
Value	0x0bU

5.2.3.2.43. CANIF_SID_INIT

Purpose	CanIf_Init() service ID.
Value	0x01U

5.2.3.2.44. CANIF_SID_READRXNOTIFSTATUS

Purpose	CanIf_ReadRxNotifStatus() service ID.
Value	0x08U

5.2.3.2.45. CANIF_SID_READRXPDUDATA

Purpose	CanIf_ReadRxPduData() service ID.
Value	0x06U

5.2.3.2.46. CANIF_SID_READTXNOTIFSTATUS

Purpose	CanIf_ReadTxNotifStatus() service ID.
Value	0x07U

5.2.3.2.47. CANIF_SID_RXINDICATION

Purpose	CanIf_RxIndication() service ID.
Value	0x14U

5.2.3.2.48. CANIF_SID_SETBAUDRATE

Purpose	CanIf_SetBaudrate() service ID.
Value	0x27U

5.2.3.2.49. CANIF_SID_SETCONTROLLERMODE

Purpose	CanIf_SetControllerMode() service ID.
Value	0x03U

5.2.3.2.50. CANIF_SID_SETDYNAMICTXID

Purpose	CanIf_SetDynamicTxId() service ID.
Value	0x0cU

5.2.3.2.51. CANIF_SID_SETPDUMODE

Purpose	CanIf_SetPduMode() service ID.
Value	0x09U

5.2.3.2.52. CANIF_SID_SETTRANSCEIVERMODE

Purpose	CanIf_SetTrcvMode() service ID.
Value	0x0dU

5.2.3.2.53. CANIF_SID_SETTRCVWAKEUPMODE

Purpose	CanIf_SetTrcvWakeupMode() service ID.
Value	0x10U

5.2.3.2.54. CANIF_SID_TRANSCEIVERMODEIND

Purpose	CanIf_TrcvModeIndication() service ID.
Value	0x18U

5.2.3.2.55. CANIF_SID_TRANSMIT

Purpose	CanIf_Transmit() service ID.
Value	0x05U

5.2.3.2.56. CANIF_SID_TXCONFIRMATION

Purpose	CanIf_TxConfirmation() service ID.
Value	0x13U

5.2.3.2.57. CANIF_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.
Value	6U

5.2.3.2.58. CANIF_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	10U

5.2.3.2.59. CANIF_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.

Value	19U

5.2.3.2.60. CANIF_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

5.2.3.3. Objects

5.2.3.3.1. CanIf_InitCfg_Xxx

Purpose	CanIf configuration.
Type	`const CanIf_ConfigType`
Description	This is the CanIf configuration that can be given to CanIf_Init as configuration parameter. Please note that the name of this element is configuration dependent and defined by the name of the CanIfInitCfg container.

5.2.3.4. Functions

5.2.3.4.1. CanIf_CancelTransmit

Purpose	Cancel transmit dummy function.	
Synopsis	`Std_ReturnType` **`CanIf_CancelTransmit`** `( PduIdType CanTxPduId );`	
Service ID	0x18	
Sync/Async	Asynchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`CanTxPduId`	Tx L-PDU handle
Return Value	Always return E_OK	
Description	This function has no functionality and is called by the AUTOSAR PduR to achieve bus agnostic behavior. Preconditions:	

	▶ The parameter CanTxPduId must be a valid Tx L-PDU

5.2.3.4.2. CanIf_CheckTrcvWakeFlag

Purpose	Check the wake flag of the designated CAN transceiver.	
Synopsis	`Std_ReturnType` **`CanIf_CheckTrcvWakeFlag`** `( uint8 TransceiverId );`	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different CAN transceiver	
Parameters (in)	`TransceiverId`	designated CAN transceiver
Return Value	Result of operation	
	`E_OK`	Request has been accepted
	`E_NOT_OK`	Request has not been accepted
Description	{0x1f}	

5.2.3.4.3. CanIf_CheckValidation

Purpose	Check for validated wakeup events.	
Synopsis	`Std_ReturnType` **`CanIf_CheckValidation`** `( EcuM_WakeupSourceType WakeupSource );`	
Service ID	0x12	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`WakeupSource`	Bitfield of wakeup sources to be validated
Return Value	Result of operation	
	`E_OK`	The validation request was accepted and executed
	`E_NOT_OK`	The validation request was not accepted
Description	This function is used to validate previous wakeup events.	

5.2.3.4.4. CanIf_CheckWakeup

Purpose	Check for Can and CanTrcv wakeups.

Synopsis	`Std_ReturnType` **`CanIf_CheckWakeup`** `( EcuM_WakeupSourceType WakeupSource );`	
Service ID	0x11	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`WakeupSource`	Bitfield of wakeup sources to be checked
Return Value	Result of operation	
	`E_OK`	The validation request was accepted and executed
	`E_NOT_OK`	The validation request was not accepted
Description	This function is used to check whether an underlying CAN driver or CAN transceiver driver signals a wakeup event.	

5.2.3.4.5. CanIf_ClearTrcvWufFlag

Purpose	Clear WUF flag of the designated CAN transceiver.	
Synopsis	`Std_ReturnType` **`CanIf_ClearTrcvWufFlag`** `( uint8 TransceiverId );`	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different CAN transceiver	
Parameters (in)	`TransceiverId`	designated CAN transceiver
Return Value	Result of operation	
	`E_OK`	Request has been accepted
	`E_NOT_OK`	Request has not been accepted
Description	{0x1e}	

5.2.3.4.6. CanIf_EnableBusMirroring

Purpose	Controls the mirroring on a controller.	
Synopsis	`Std_ReturnType` **`CanIf_EnableBusMirroring`** `( uint8 ControllerId , boolean MirroringActive );`	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`ControllerId`	Abstracted CanIf ControllerId which is assigned to a CAN controller.

	MirroringActive	TRUE: Mirror_ReportCanFrame will be called for each frame received or transmitted on the given controller. FALSE: Mirror_ReportCanFrame will not be called for the given controller.
Return Value	Result of the operation	
	E_OK	Mirroring mode was changed.
	E_NOT_OK	Wrong ControllerId.
Description	{0x4c}	

5.2.3.4.7. CanIf_GetControllerErrorState

Purpose	Returns the error state of the controller.	
Synopsis	`Std_ReturnType` **`CanIf_GetControllerErrorState`** `( uint8 ControllerId , Can_ErrorStateType * ErrorStatePtr );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller.
Parameters (out)	ErrorStatePtr	Pointer to a memory location, where the error state of the CAN controller will be stored.
Return Value	Result of the operation	
	E_OK	Error state request has been accepted or always if CanIfCanControllerErrorStateSupported is FALSE.
	E_NOT_OK	Error state request has not been accepted. This is never returned if CanIfCanControllerErrorStateSupported is FALSE.
Description	{0x4b}	

5.2.3.4.8. CanIf_GetControllerMode

Purpose	Get controller mode.
Synopsis	`Std_ReturnType` **`CanIf_GetControllerMode`** `( uint8 ControllerId , Can_ControllerStateType * ControllerModePtr );`

Service ID	0x04	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`ControllerId`	CAN controller
Parameters (out)	`ControllerModePtr`	Pointer for returning the current mode
Return Value	Result of operation	
	`E_OK`	The returned mode is valid
	`E_NOT_OK`	An error occurred during function execution
Description	This function queries the mode of the controller given in ControllerId. Preconditions: ► The CAN interface must already be initialized by CanIf_Init() ► The parameter ControllerId must address a valid controller ► The parameter ControllerModePtr must be a valid pointer	

5.2.3.4.9. CanIf_GetControllerTxErrorCounter

Purpose	Returns the TX error counter.	
Synopsis	`Std_ReturnType` **`CanIf_GetControllerTxErrorCounter`** `( uint8 ControllerId , uint8 * TxErrorCounterPtr );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`ControllerId`	Abstracted CanIf ControllerId which is assigned to a CAN controller.
Parameters (out)	`TxErrorCounterPtr`	Pointer to a memory location, where the current Tx error counter of the CAN controller will be stored.
Return Value	Result of the operation	
	`E_OK`	Error state request has been accepted or always if CanIfCanTxErrorCounterSupported is FALSE.
	`E_NOT_OK`	Error state request has not been accepted. This is never returned if CanIfCanTxErrorCounterSupported is FALSE.

Description	{0x4b}

5.2.3.4.10. CanIf_GetPduMode

Purpose	Read a L-PDU channel mode.	
Synopsis	`Std_ReturnType` **`CanIf_GetPduMode`** `( uint8 ControllerId , CanIf_PduModeType * PduModePtr );`	
Service ID	0x0a	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`ControllerId`	CAN controller
Parameters (out)	`PduModePtr`	Pointer to return the current mode
Return Value	Result of operation	
	`E_OK`	Channel mode request has been accepted
	`E_NOT_OK`	Channel mode request has not been accepted
Description	This function returns the current PDU mode of the requested controller (ControllerId) at the memory location referenced by PduModePtr. Preconditions: ► The CAN interface must already be initialized by CanIf_Init() ► The parameter ControllerId must be a valid CAN controller ► The parameter PduModePtr must be a valid pointer	

5.2.3.4.11. CanIf_GetTrcvMode

Purpose	Get transceiver mode.	
Synopsis	`Std_ReturnType` **`CanIf_GetTrcvMode`** `( uint8 TransceiverId , CanTrcv_TrcvModeType * TransceiverModePtr );`	
Service ID	0x0e	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`TransceiverModePtr`	Pointer to mode variable

	`TransceiverId`	CAN transceiver ID
Return Value	Result of operation	
	`E_OK`	Transceiver mode was changed as requested
	`E_NOT_OK`	Transceiver mode change failed; previous mode is still valid
Description	This function is used to read the mode of the transceiver assigned to transceiver channel TransceiverId to the memory location TransceiverModePtr.	

5.2.3.4.12. CanIf_GetTrcvWakeupReason

Purpose	Get transceiver wakeup reason.	
Synopsis	`Std_ReturnType` **`CanIf_GetTrcvWakeupReason`** `( uint8 TransceiverId , CanTrcv_TrcvWakeupReasonType * TrcvWuReasonPtr );`	
Service ID	0x0f	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`TransceiverId`	CAN transceiver channel ID
	`TrcvWuReasonPtr`	Address to write wakeup reason to
Return Value	Result of operation	
	`E_OK`	Wakeup reason was correctly read
	`E_NOT_OK`	Wakeup reason could not be read
Description	This function is used to read the last wakeup reason of the transceiver assigned to transceiver channel TransceiverId into TrcvWuReasonPtr.	

5.2.3.4.13. CanIf_GetTxConfirmationState

Purpose	Report controller Tx confirmation state.
Synopsis	`CanIf_NotifStatusType` **`CanIf_GetTxConfirmationState`** `( uint8 ControllerId );`
Service ID	0x19
Sync/Async	Synchronous
Reentrancy	Reentrant

Parameters (in)	`ControllerId`	Abstracted CanIf controller ID which is assigned to the CAN controller
Return Value	Notification status of the controller	
	`CANIF_NO_NOTIFICATION`	No notification was received
	`CANIF_TX_RX_NOTIFICATION`	A Tx confirmation was received
Description	This function reports, if any Tx confirmation has been received for the whole CAN controller since the last controller start. Preconditions: ► The parameter ControllerId must be a valid CanIf controller ID	

5.2.3.4.14. CanIf_GetVersionInfo

Purpose	Return module version information.	
Synopsis	`void` **`CanIf_GetVersionInfo`** `( Std_VersionInfoType * Versioninfo );`	
Service ID	0x0b	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (out)	`Versioninfo`	Version information
Description	This function returns the CAN interface version information in the memory area Versioninfo references. Preconditions: ► The parameter Versioninfo may not be a NULL pointer	

5.2.3.4.15. CanIf_Init

Purpose	CAN interface initialization function.
Synopsis	`void` **`CanIf_Init`** `( const CanIf_ConfigType * ConfigPtr );`
Service ID	0x01
Sync/Async	Synchronous
Reentrancy	Non-Reentrant

Parameters (in)	`ConfigPtr`	Pointer to the interface configuration
Description	This function initializes the CAN interface. Caution: ► CanIf_Init() shall not preempt any other CanIf function. ► No other CanIf function shall interrupt CanIf_Init().	

5.2.3.4.16. CanIf_IsValidConfig

Purpose	Validate configuration.
Synopsis	`Std_ReturnType` **`CanIf_IsValidConfig`** `( const void * voidConfigPtr );`
Service ID	0x60
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.

5.2.3.4.17. CanIf_ReadRxNotifStatus

Purpose	Read Rx notification status.	
Synopsis	`CanIf_NotifStatusType` **`CanIf_ReadRxNotifStatus`** `( PduIdType CanRxPduId );`	
Service ID	0x08	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`CanRxPduId`	Rx L-PDU handle
Return Value	Current notification status of the corresponding Rx L-PDU	
Description	This function provides the Rx L-PDU receive notification status of L-PDU CanRxPduId. Preconditions: ► The CAN interface must already be initialized by CanIf_Init()	

	▶ Transmit notification API must be enabled ▶ The parameter CanRxPduId must be a valid Rx L-PDU.

5.2.3.4.18. CanIf_ReadRxPduData

Purpose	Read received data.	
Synopsis	`Std_ReturnType` **`CanIf_ReadRxPduData`** `( PduIdType CanRxPduId , PduInfoType * PduInfoPtr );`	
Service ID	0x06	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`CanRxPduId`	Rx L-PDU handle
Parameters (out)	`PduInfoPtr`	Memory pointer to store received data
Return Value	Result of operation	
	`E_OK`	Request has been accepted
	`E_NOT_OK`	Request has not been accepted
Description	This function reads data previously received and stored in an internal buffer. Preconditions: ▶ The CAN interface must already be initialized by CanIf_Init() ▶ This API must be enabled by configuration ▶ The parameter CanRxPduId must be a valid Rx L-PDU ▶ The parameter PduInfoPtr must be a valid pointer	

5.2.3.4.19. CanIf_ReadTxNotifStatus

Purpose	Read Tx notification status.
Synopsis	`CanIf_NotifStatusType` **`CanIf_ReadTxNotifStatus`** `( PduIdType CanTxPduId );`
Service ID	0x07
Sync/Async	Synchronous
Reentrancy	Non-Reentrant

Parameters (in)	`CanTxPduId`	Tx L-PDU handle
Return Value	Current notification status of the corresponding Tx L-PDU	
Description	This function provides the Tx L-PDU transmit notification status of L-PDU CanTxPduId. Preconditions: ► The CAN interface must already be initialized by CanIf_Init() ► Transmit notification API must be enabled ► The parameter CanTxPduId must be a valid Tx L-PDU.	

5.2.3.4.20. CanIf_SetBaudrate

Purpose	Set Baudrate API function.	
Synopsis	`Std_ReturnType` **`CanIf_SetBaudrate`** `( uint8 ControllerId , uint16 BaudRateConfigID );`	
Service ID	0x27	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different Controllers	
Parameters (in)	`ControllerId`	CAN controller, whose baud rate shall be set
	`BaudRateConfigID`	references a baud rate configuration by ID
Return Value	Result of operation	
	`E_OK`	Service request accepted, setting of (new) baud rate started
	`E_NOT_OK`	Service request not accepted
Description	This service shall set the baud rate configuration of the CAN controller. Depending on necessary baud rate modifications the controller might have to reset.	

5.2.3.4.21. CanIf_SetControllerMode

Purpose	Controller mode setting function.
Synopsis	`Std_ReturnType` **`CanIf_SetControllerMode`** `( uint8 ControllerId , CanIf_ControllerModeType ControllerMode );`

Service ID	0x03	
Sync/Async	Asynchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`ControllerId`	Target controller
	`ControllerMode`	Requested mode transition
Return Value	Result of operation	
	`E_OK`	Controller mode request has been accepted
	`E_NOT_OK`	Controller mode request has not been accepted
Description	This function performs a mode transition of the controller given in ControllerId to the mode ControllerMode Preconditions: ► The CAN interface must already be initialized by CanIf_Init() ► The parameter ControllerId must address a valid controller ► The parameter ControllerMode must be a valid mode	

5.2.3.4.22. CanIf_SetDynamicTxId

Purpose	Set CAN ID of dynamic Tx L-PDU.	
Synopsis	`void CanIf_SetDynamicTxId ( PduIdType CanTxPduId , Can_IdType CanId );`	
Service ID	0x0c	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`CanTxPduId`	Tx L-PDU handle
	`CanId`	New CAN ID
Description	This function sets the CAN ID of Tx L-PDU CanTxPduId to the new value CanId. Preconditions: ► The CAN interface must already be initialized by CanIf_Init() ► Dynamic Tx L-PDUs are used in the current configuration ► The parameter CanTxPduId is a valid dynamic Tx L-PDU	

	▶ The parameter CanId is a valid CAN ID

5.2.3.4.23. CanIf_SetPduMode

Purpose	Set requested L-PDU channel mode.	
Synopsis	`Std_ReturnType` **`CanIf_SetPduMode`** `( uint8 ControllerId , CanIf_PduModeType PduModeRequest );`	
Service ID	0x09	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`ControllerId`	CAN controller
	`PduModeRequest`	Requested PDU mode
Return Value	Result of operation	
	`E_OK`	Request for mode transition has been accepted
	`E_NOT_OK`	Request for mode transition has not been accepted
Description	This function sets the L-PDU channel mode for the requested controller (ControllerId) to the requested mode (PduModeRequest). Preconditions: ▶ The CAN interface must already be initialized by CanIf_Init() ▶ The parameter ControllerId must be a valid CAN controller. ▶ The parameter PduModeRequest must be a valid PDU mode.	

5.2.3.4.24. CanIf_SetTrcvMode

Purpose	Set transceiver mode.
Synopsis	`Std_ReturnType` **`CanIf_SetTrcvMode`** `( uint8 TransceiverId , CanTrcv_TrcvModeType TransceiverMode );`
Service ID	0x0d
Sync/Async	Synchronous
Reentrancy	Non-Reentrant

Parameters (in)	`TransceiverId`	CAN transceiver ID
	`TransceiverMode`	Requested transceiver mode
Return Value	Result of operation	
	`E_OK`	Transceiver mode was changed as requested
	`E_NOT_OK`	Transceiver mode change failed; previous mode is still valid
Description	This function is used to set the mode of the transceiver assigned to transceiver channel TransceiverId to the mode given in TransceiverMode.	

5.2.3.4.25. CanIf_SetTrcvWakeupMode

Purpose	Set transceiver wakeup mode.	
Synopsis	`Std_ReturnType` **`CanIf_SetTrcvWakeupMode`** `( uint8 TransceiverId , CanTrcv_TrcvWakeupModeType TrcvWakeupMode );`	
Service ID	0x10	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`TransceiverId`	CAN transceiver channel ID
	`TrcvWakeupMode`	Wakeup mode/event to set
Return Value	Result of operation	
	`E_OK`	Wakeup state has been changed
	`E_NOT_OK`	Wakeup state change has failed
Description	This function is used to set the wakeup mode/event TrcvWakeupMode of the transceiver assigned to transceiver channel TransceiverId.	

5.2.3.4.26. CanIf_Transmit

Purpose	Transmit an L-PDU.
Synopsis	`Std_ReturnType` **`CanIf_Transmit`** `( PduIdType CanTxPduId , const PduInfoType * PduInfoPtr );`
Service ID	0x05
Sync/Async	Synchronous

Reentrancy	Reentrant	
Parameters (in)	`CanTxPduId`	Tx L-PDU handle
	`PduInfoPtr`	Message content to be transmitted
Return Value	Result of operation	
	`E_OK`	Transmit request has been accepted
	`E_NOT_OK`	Transmit request has not been accepted
Description	This function transmits the data given through PduInfoPtr through the L-PDU given by CanTxPduId. Preconditions: ► The CAN interface must already be initialized by CanIf_Init() ► The parameter CanTxPduId must be a valid Tx L-PDU ► The parameter PduInfoPtr must be a valid pointer	

5.2.4. Integration notes

5.2.4.1. Exclusive areas

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

5.2.4.1.1. SCHM_CANIF_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.

5.2.4.2. Production errors

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

5.2.4.3. 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
CONST_32
VAR_INIT_8
CONST_8
VAR_INIT_UNSPECIFIED
VAR_FAST_INIT_UNSPECIFIED
VAR_CLEARED_8
VAR_CLEARED_16
VAR_CLEARED_32
VAR_CLEARED_UNSPECIFIED
CONST_UNSPECIFIED
CODE
CONFIG_DATA_UNSPECIFIED
CODE_CC_BLOCK

5.2.4.4. 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.

5.2.4.4.1. lim.CanIf.EB_INTREQ_CanIf_0001

Description	Data copying mechanism in Rx indication code CanIf_RxIndication() does not copy the data provided by the CAN driver. Instead the data pointer is directly propagated to the upper layer.

	NOTE: For dynamic length PDU reception, upper layers might expect that always a buffer with the maximum possible PDU size is provided. Since CanIf does not perform buffering it is the Can drivers responsibility to fulfil this requirement.
Rationale	The data is copied in the different upper layers. The CanIf does not copy it as well to reduce runtime overhead.

5.2.4.4.2. lim.CanIf.EB_INTREQ_CanIf_0002

Description	CanIf_CheckWakeup() must not preempt or be preempted by CanIf_SetControllerMode() The integrator must assure that CanIf_SetControllerMode() cannot preempt CanIf_CheckWakeup(). The integrator also must assure that CanIf_CheckWakeup() does not preempt CanIf_SetControllerMode().
Rationale	This limitation reduces code size and execution time by eliminating the need for extensive use of exclusive areas.

5.2.4.4.3. lim.CanIf.EB_INTREQ_CanIf_0003

Description	CanIf controller mode might differ from Can controller hardware mode The CanIf software controller mode is always changed if an according event on the hardware is detected. These events are the following: - Call of CanIf_ControllerModeIndication() (CANIF_CS_STOPPED, CANIF_CS_STARTED, CANIF_CS_SLEEP) - Call of CanIf_ControllerBusOff() (CANIF_CS_STOPPED) - Wakeup detection by calling Can_CheckWakeup() (CANIF_CS_STOPPED) The state is always set to the latest detected/reported event. The result therefore is strongly dependent from the actual Can driver module behavior. Possible inconsistencies: - Can_SetControllerMode(CAN_T_START) (current state is CANIF_CS_STARTED) - CanIf_ControllerBusOff() -> controller mode is changed to CANIF_CS_STOPPED both in CanIF SW and HW. - CanIf_ControllerModeIndication(CANIF_CS_STARTED) -> controller mode is changed to CANIF_CS_STARTED in CanIf SW although hardware stays stopped.

	- Can_SetControllerMode(CAN_T_SLEEP) (current state is CANIF_CS_STOPPED) - CanIf_CheckWakeup() reports wakeup event -> controller mode is changed to CANIF_CS_STOPPED both in CanIf SW and HW. - CanIf_ControllerModeIndication(CAN_T_SLEEP) -> controller mode is changed to CAN_T_SLEEP in CanIf SW although hardware stays stopped.
Rationale	

5.2.4.4.4. lim.CanIf.EB_INTREQ_CanIf_0004

Description	When the support for Bus Mirroring is enabled the signature of the functions CanIf_GetTrcvMode() and CanIf_GetControllerMode() is according to the AR4.4 specs. As these are called from integration code / CDDs, the correct order and type of the parameters has to be ensured.
Rationale	Changing the signature of the functions guarantees interoperability with the Bus Mirroring module while maintaining backwards compatibility when the feature is off.

5.2.4.4.5. lim.CanIf.EB_INTREQ_CanIf_0005

Description	Starting with ACG-8.7.2, CanIf supports compatibility with CAN driver ASR 4.3 and 4.4. During this development, CanIfEnableCanRel422Compatibility and CanIfEnableCanRev2Compatibility were removed and replaced with CanIfCanDriverCompatibility. After updating to this ACG version, replacing the obsolete parameters with the new one shall be taken into account. This will make the tests that used both CanIfEnableCanRel422Compatibility and CanIfEnableCanRev2Compatibility unusable, considering that a driver cannot be ASR 4.2.2 and ASR 4.0.2 at the same time anyway.
Rationale	Using this new paramenter CanIfCanDriverCompatibility, it is easier to specify the version of CAN driver the CAN Interface will be referring to.

5.2.4.4.6. lim.CanIf.EB_INTREQ_CanIf_0006

Description	When the support for Multicore is enabled, the CDD callbacks (defined via CanIfNumberOfEnqueuedRxPdusApiName, CanIfNumberOfRxPdusExceedingQueueApiName, CanIfNumberOfEnqueuedTxPdusApiName or CanIfNumberOfTxPdusExceedingQueueApiName parameters) for Decoupled processing queue size measurement support must be concurrently callable from different partitions/cores for different MainFunction Ids.
Rationale	Assuring that the CDD callbacks can operate on Multicore processors.

5.2.4.4.7. lim.CanIf.EB_INTREQ_CanIf_0007

Description	When the support for Multicore is enabled, the CDD callbacks (defined via CanIfHookOnReceptionFunctionName parameter) for the Custom Hook support must be concurrently callable from different partitions/cores for different PDU Ids.
Rationale	Assuring that the CDD callbacks can operate on Multicore processors.

5.2.4.4.8. lim.CanIf.EB_INTREQ_CanIf_0008

Description	When the support for Multicore is enabled, all PDUs corresponding to the J1939 stack must be mapped on the same core (the same one on which the J1939 stack is located) if the CAN stack is Multi-core distributed along network boundaries. This includes all the resources corresponding to those PDUs: interrupts, main functions, etc.
Rationale	The J1939 stack has no Multicore distribution capabilities.

5.2.4.4.9. lim.CanIf.EB_INTREQ_CanIf_0009

Description	When the support for Multicore is enabled, the CDD RxIndication callback (defined via CanIfRxPduUserRxIndicationName parameter) must be concurrently callable from different partitions/cores for different PDU Ids.
Rationale	Assuring that the CDD callback can operate on Multicore processors.

5.2.4.4.10. lim.CanIf.EB_INTREQ_CanIf_0010

Description	When the support for Multicore is enabled, the CDD TxConfirmation callback (defined via CanIfTxPduUserTxConfirmationName parameter) must be concurrently callable from different partitions/cores for different PDU Ids.
Rationale	Assuring that the CDD callback can operate on Multicore processors.

5.2.4.4.11. lim.CanIf.EB_INTREQ_CanIf_0011

Description	When the support for Multicore is enabled, the Can_SetControllerMode(), Can_SetBaudrate() and Can_CheckWakeup() APIs must be concurrently callable from different partitions/cores for different controller Ids.
Rationale	Assuring that the MCAL driver can operate on Multicore processors.

5.2.4.4.12. lim.CanIf.EB_INTREQ_CanIf_0012

Description	When the support for Multicore is enabled, the Can_Write() API must be concurrently callable from different partitions/cores for different hardware objects.
Rationale	Assuring that the MCAL driver can operate on Multicore processors.

5.2.4.4.13. lim.CanIf.EB_INTREQ_CanIf_0013

Description	When the support for Multicore is enabled, the CanTrcv_SetOpMode(), CanTrcv_GetOpMode(), CanTrcv_GetBusWuReason(), CanTrcv_SetWakeupMode(), CanTrcv_CheckWakeup() and CanTrcv_CheckWakeFlag() APIs must be concurrently callable from different partitions/cores for different transceiver Ids.
Rationale	Assuring that the transceiver driver can operate on Multicore processors.

5.2.4.4.14. lim.CanIf.EB_INTREQ_CanIf_0014

Description	When the support for Multicore is enabled, the CDD ControllerBusOff callback (defined via CanIfDispatchUserCtrlBusOffName parameter) must be concurrently callable from different partitions/cores for different controller Ids.
Rationale	Assuring that the CDD callback can operate on Multicore processors.

5.2.4.4.15. lim.CanIf.EB_INTREQ_CanIf_0015

Description	When the support for Multicore is enabled, the CDD ControllerModeIndication callback (defined via CanIfDispatchUserCtrlModeIndicationName parameter) must be concurrently callable from different partitions/cores for different controller Ids.
Rationale	Assuring that the CDD callback can operate on Multicore processors.

5.2.4.4.16. lim.CanIf.EB_INTREQ_CanIf_0016

Description	When the support for Multicore is enabled, the CDD TransceiverModeIndication callback (defined via CanIfDispatchUserTrcvModeIndicationName parameter) must be concurrently callable from different partitions/cores for different transceiver Ids.
Rationale	Assuring that the CDD callback can operate on Multicore processors.

5.2.4.4.17. lim.CanIf.EB_INTREQ_CanIf_0017

Description	When the support for Multicore is enabled, the CDD ConfirmPnAvailability callback (defined via CanIfDispatchUserConfirmPnAvailabilityName parameter) must be concurrently callable from different partitions/cores for different transceiver Ids.
Rationale	Assuring that the CDD callback can operate on Multicore processors.

5.2.4.4.18. lim.CanIf.EB_INTREQ_CanIf_0018

Description	When the support for Multicore is enabled, the CDD ClearTrcvWufFlagIndication callback (defined via CanIfDispatchUserClearTrcvWufFlagIndicationName parameter) must be concurrently callable from different partitions/cores for different transceiver Ids.
Rationale	Assuring that the CDD callback can operate on Multicore processors.

5.2.4.4.19. lim.CanIf.EB_INTREQ_CanIf_0019

Description	When the support for Multicore is enabled, the CDD CheckTransceiverWakeFlagIndication callback (defined via CanIfDispatchUserCheckTrcvWakeFlagIndicationName parameter) must be concurrently callable from different partitions/cores for different transceiver Ids.
Rationale	Assuring that the CDD callback can operate on Multicore processors.

5.3. CanNm

5.3.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CanNmGeneral	1..1	
CanNmGlobalConfig	1..1	This container contains the global configuration parameter of the CanNm. The parameters and the parameters of the sub

Containers included		
		containers shall be mapped to the C data type CanNm_ConfigType (for parameters where it is possible) which is passed to the CanNm_Init function. The CanNm only supports compile time configuration. Only the parameters from the first module configuration container from this list are used for module configuration. .
CanNmDefensiveProgramming	1..1	**Label:** Defensive Programming Options Parameters for defensive programming
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	
Configuration class	**VariantPostBuild:**	VariantPostBuild

5.3.1.1. CanNmGeneral

Parameters included	
Parameter name	**Multiplicity**
CanNmMultiCoreSupport	1..1
CanNmPnSupported	1..1

Parameters included	
CanNmRelocatablePbcfgEnable	1..1
CanNmMaxPn	0..1

Parameter Name	CanNmMultiCoreSupport	
Label	CanNm multicore support	
Description	Enables MultiCoreSupport.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CanNmPnSupported	
Label	Support for Partial Network Cluster (PNC)	
Description	Enables or disables support of partial networking. ► `False`: Partial Networking is disabled ► `True`: Partial Networking is enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CanNmRelocatablePbcfgEnable	
Label	CanNmRelocatablePbcfgEnable	
Description	Enables/disables 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	**CanNmMaxPn**	
Label	CanNmMaxPn	
Description	The maximum number of Partial Networking Clusters that can be configured.	
Multiplicity	0..1	
Type	INTEGER	
Default value	0	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.3.1.2. CanNmGlobalConfig

Containers included		
Container name	**Multiplicity**	**Description**
CanNmChannelConfig	1..n	**Label:** Channel Configuration This container holds the channel specific configuration parameter of the CanNm.
CanNmPnInfo	0..1	PN information configuration.

Parameters included	
Parameter name	**Multiplicity**
CanNmActiveWakeupBitEnabled	1..1
CanNmBusLoadReductionEnabled	1..1
CanNmBusSynchronizationEnabled	1..1
CanNmComControlEnabled	1..1
CanNmNodeIdCallback	0..1
CanNmNodeIdCallbackHeader	1..1
CanNmComUserDataSupport	1..1
CanNmCoordinatorSyncSupport	1..1
CanNmDevErrorDetect	1..1
CanNmImmediateRestartEnabled	1..1
CanNmImmediateTxconfEnabled	1..1
CanNmMainFunctionPeriod	1..1

Parameters included	
CanNmNumberOfChannels	1..1
CanNmPostBuildRamSize	1..1
CanNmPassiveModeEnabled	1..1
CanNmPduRxIndicationEnabled	1..1
CanNmPnEiraCalcEnabled	0..1
CanNmPnResetTime	0..1
CanNmRemoteSleepIndEnabled	1..1
CanNmStateChangeIndEnabled	1..1
CanNmUserDataEnabled	1..1
CanNmVersionInfoApi	1..1
CanNmPnEiraRxNSduRef	0..1

Parameter Name	CanNmActiveWakeupBitEnabled	
Label	Active Wakeup Bit Enable	
Description	Enables/Disables the handling of the Active Wakeup Bit in the CanNm module.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	CanNmBusLoadReductionEnabled	
Label	Bus Load Reduction	
Description	Pre-processor switch for enabling busload reduction support. The bus load reduction mechanisms ensures that the bus load is limited to maximum two NM messages within a Message Cycle Time (`CanNmMsgCycleTime`). Dependencies: ► Passive Mode must be disabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild

Origin	AUTOSAR_ECUC

Parameter Name	**CanNmBusSynchronizationEnabled**	
Label	Bus Synchronization	
Description	Pre-processor switch for enabling bus synchronization support. This feature is required for gateway nodes only. . The bus synchronization functionality triggers the transmission of a single NM message independently of the normal periodic transmission. Therefore, the following API function is provided: ► `CanNm_RequestBusSynchronization()` Dependencies: ► The value of this parameter has to be synchronized with the value of respective parameter in the Nm module. ► Passive Mode must be disabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmComControlEnabled**	
Label	Communication Control	
Description	Enable the Communication Control support. Dependencies: ► The value of this parameter has to be synchronized with the value of respective parameter in the Nm module. ► Passive Mode must be disabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild

Origin	AUTOSAR_ECUC

Parameter Name	**CanNmNodeIdCallback**	
Label	CanNm Node Id callback	
Description	Name of the callback function to be called if CanNmNodeIdCallback is enabled.	
Multiplicity	0..1	
Type	FUNCTION-NAME	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanNmNodeIdCallbackHeader**	
Label	Node Id callback header	
Description	The name of a header file that will be included to obtain the external declaration of the callback function. Dependencies: ► This parameter is only available if CanNmNodeIdCallback is enabled.	
Multiplicity	1..1	
Type	STRING	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanNmComUserDataSupport**	
Description	Enable/disable the user data support.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmCoordinatorSyncSupport**
Description	*The functionality related to this parameter is not supported by the current implementation.* Enables/disables the coordinator synchronisation support.
Multiplicity	1..1

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

Parameter Name	**CanNmDevErrorDetect**	
Label	Enable Development Error Detection	
Description	Enable development error detection.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmImmediateRestartEnabled**	
Label	Immediate Restart	
Description	Enabling the asynchronous transmission of a NM PDU upon bus communication request in Prepare-Bus-Sleep mode. Dependencies: ► Passive Mode must be disabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmImmediateTxconfEnabled**
Label	Immediate Transmission Confirmation
Description	Enable the immediate transmission(Tx) confirmation functionality. If this parameter is enabled it is assumed that each Network Management PDU transmission request results in a successful Network Management PDU transmission. Dependencies:

	► Passive Mode must be disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanNmMainFunctionPeriod**
Label	Main Function Period [s]
Description	Call cycle in seconds of CanNm_MainFunction `CanNm_MainFunction_x`
Multiplicity	1..1
Type	FLOAT
Default value	0.001
Range	<=0.255 >=0.001
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanNmNumberOfChannels**
Label	Number Of Channels
Description	Maximum number of Can NM channels allowed within one ECU.
Multiplicity	1..1
Type	INTEGER
Default value	1
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanNmPostBuildRamSize**
Label	CanNmPostBuildRamSize
Description	Number of bytes for TX and RX buffers Value should be set as: the sum of the first RxPdu lengths on each channel multiplied with 2 (in case passive mode is disabled). Size should be big enugh to hold eventual changes of PDU lenghts at postbuild time
Multiplicity	1..1

Type	INTEGER	
Default value	96	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanNmPassiveModeEnabled**	
Label	Passive Mode	
Description	Enable the passive mode. In passive mode, the CanNm will not be able to wake up the bus and will not send NM messages. It will only listen to the NM messages and silently monitor the bus.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmPduRxIndicationEnabled**	
Label	PDU Receive Indication	
Description	Enable the notification of reception of a NM message. If a NM message is received the function `Nm_PduRxIndication()` is called. Dependencies: ► The value of this parameter has to be synchronized with the value of respective parameter in the Nm module.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmPnEiraCalcEnabled**
Description	Specifies if CanNm calculates the PN request information for internal and external requests.
Multiplicity	0..1

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

Parameter Name	**CanNmPnResetTime**	
Description	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests in the EIRA and in the ERA. The value shall be the same for every channel.	
Multiplicity	0..1	
Type	FLOAT	
Default value	0.01	
Range	<=65.535 >=0.0010	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmRemoteSleepIndEnabled**	
Label	Remote Sleep Indication	
Description	Enable Support for Remote Sleep Indication. The Remote Sleep Indication allows the CanNm module to detect a situation where all nodes in the cluster are ready to sleep apart from one node which still keeps the bus awake. Therefore the following API is provided: ► `Nm_CheckRemoteSleepIndication()` Dependencies: ► The value of this parameter has to be synchronized with the value of respective parameter in the Nm module. ► Passive Mode must be disabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild

Origin	AUTOSAR_ECUC

Parameter Name	**CanNmStateChangeIndEnabled**	
Label	State Change Indication	
Description	Pre-processor switch for enabling the CAN NM state change notification. This parameter shall be derived from NM_STATE_CHANGE_IND_ENABLED.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmUserDataEnabled**
Label	User Data
Description	Enable support for transmission of user data in NM messages. Therefore the following API functions are provided: ► `Nm_GetUserData()` ► `Nm_SetUserData()` *(Only if Passive Mode Support is disabled)* Dependencies: ► The value of this parameter has to be synchronized with the value of respective parameter in the Nm module.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanNmVersionInfoApi**
Label	Enable Version Info API
Description	Provide API function for retrieving version information: ► `CanNm_GetVersionInfo()`
Multiplicity	1..1
Type	BOOLEAN

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

Parameter Name	**CanNmPnEiraRxNSduRef**	
Description	Reference to a Pdu in the COM-Stack. EIRA is forwarded to PduR using this Pdu.	
Multiplicity	0..1	
Type	REFERENCE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.3.1.3. CanNmChannelConfig

Containers included		
Container name	**Multiplicity**	**Description**
CanNmRxPdu	1..n	**Label:** Receive PDU ID and Reference This container holds the `CanNmRxPduId` and the `CanNmRxPduRef`.
CanNmTxPdu	0..1	**Label:** Transmit PDU Reference This container contains the `CanNmTxConfirmationPduId` and the `CanNmTxPduRef`.
CanNmUserDataTxPdu	0..1	**Label:** User Data Transmission PDU This optional container is used to configure the UserNm PDU. This container is only available if `CanNmComUserDataSupport` is enabled.
CanNmUserDataRxPdu	0..1	**Label:** User Data Reception PDUs This optional container is used to configure the UserNm PDU. This container is only available if `CanNmComUserDataSupport` is enabled.

Parameters included	
Parameter name	**Multiplicity**
CanNmNodeIdEnabled	1..1
CanNmRepeatMsgIndEnabled	1..1

Parameters included	
CanNmNodeDetectionEnabled	1..1
CanNmAllNmMessagesKeepAwake	0..1
CanNmCarWakeUpBitPosition	1..1
CanNmCarWakeUpBytePosition	1..1
CanNmCarWakeUpFilterEnabled	1..1
CanNmCarWakeUpFilterNodeId	1..1
CanNmCarWakeUpRxEnabled	1..1
CanNmPnEnabled	0..1
CanNmPnEraCalcEnabled	0..1
CanNmPnHandleMultipleNetworkRequests	0..1
CanNmPnEraRxNSduRef	0..1
CanNmBusLoadReductionActive	1..1
CanNmImmediateNmCycleTime	1..1
CanNmImmediateNmTransmissions	1..1
CanNmMsgCycleOffset	1..1
CanNmMsgCycleTime	1..1
CanNmMsgReducedTime	1..1
CanNmMsgTimeoutTime	1..1
CanNmNodeId	1..1
CanNmPduCbvPosition	1..1
CanNmPduNidPosition	1..1
CanNmRemoteSleepIndTime	1..1
CanNmRepeatMessageTime	1..1
CanNmTimeoutTime	1..1
CanNmUserDataLength	1..1
CanNmWaitBusSleepTime	1..1
CanNmComMNetworkHandleRef	1..1

Parameter Name	CanNmNodeIdEnabled
Label	Node Identifier
Description	Enable support for sending of Node Ids in NM messages and provide functions for retrieving the node identifier from the most recently received NM PDU and the local node identifier.

	Therefore the following API functions are provided: ► `Nm_GetNodeIdentifier()` ► `Nm_GetLocalNodeIdentifier()` Dependencies: ► The value of this parameter has to be synchronized with the value of the respective parameter in the Nm module.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanNmRepeatMsgIndEnabled**
Label	Repeat Message Indication
Description	Enable the notification that a Repeat Message Request Bit has been received. If a Repeat Message Request Bit has been received the function `Nm_RepeatMessageIndication()` is called. Dependencies: ► Node Detection must be enabled.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanNmNodeDetectionEnabled**
Label	Node Detection
Description	Enable the handling *Repeat Message Request Bit* in the *Control Bit Vector*. If the *Request Message Bit* in the NM message set the nodes receiving the message start sending NM messages. For setting the *Repeat Message Request Bit* in NM messages following API function is provided: ► `CanNm_RepeatMessageRequest()`

	Dependencies: ► The value of this parameter has to be synchronized with the value of the respective parameter in the Nm module. ► Support for Node Identifiers must be enabled. ► Passive Mode must be disabled.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanNmAllNmMessagesKeepAwake**
Description	Specifies if CanNm drops irrelevant NM messages.
Multiplicity	0..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanNmCarWakeUpBitPosition**
Description	Specifies the Bit position of the CWU within the CanNmCarWakeUpBytePosition.
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=7 >=0
Configuration class	**PreCompile:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanNmCarWakeUpBytePosition**
Description	Specifies the Byte position of the CWU within the NM-Message.
Multiplicity	1..1
Type	INTEGER

Default value	2	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmCarWakeUpFilterEnabled**	
Description	If CWU filtering is supported, only the CWU bit within the NM message with source node identifier CanNmCarWakeUpFilterNodeId is considered as CWU request.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmCarWakeUpFilterNodeId**	
Description	Source node identifier for CWU filtering. If CWU filtering is supported, only the CWU bit within the NM message with source node identifier CanNmCarWakeUp-FilterNodeId is considered as CWU request.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=255 >=0	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmCarWakeUpRxEnabled**	
Description	Enables or disables support of CarWakeUp bit evaluation in received NM mes-sages.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanNmPnEnabled	
Description	Enables or disables support of partial networking. ► false : Partial networking not supported. ► true : Partial networking supported.	
Multiplicity	0..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanNmPnEraCalcEnabled	
Description	Specifies if CanNm calculates the PN request information for external requests.(ERA)	
Multiplicity	0..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanNmPnHandleMultipleNetworkRequests	
Description	In case this parameter is set to **true** a call of the API function `CanNm_NetworkRequest()` in the state `NormalOperationState`, `ReadySleepState` or `RepeatMessageState` causes the CanNm to (re-)enter the `RepeatMessageState` and to send immediate Nm messages. Dependencies: ► Support for Partial Networks must be enabled for this channel (`CanNmPnEnabled`). ► The number of immediate transmissions must greater than 0 for this channel (`CanNmImmediateNmTransmissions`).	
Multiplicity	0..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PostBuild:**	VariantPostBuild

Origin	AUTOSAR_ECUC

Parameter Name	**CanNmPnEraRxNSduRef**	
Description	Reference to a Pdu in the COM-Stack. The SduRef is required for every CanNm Channel, because ERA is reported per channel.	
Multiplicity	0..1	
Type	REFERENCE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmBusLoadReductionActive**	
Label	Bus Load Reduction Active	
Description	This parameter defines if bus load reduction for the respective NM channel is active or not. Dependencies: ► Support for Bus Load Reduction must be enabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmImmediateNmCycleTime**
Label	Immediate NM PDU cycle time
Description	Defines the immediate NM PDU cycle time in seconds which is used for CanNmImmediateNmTransmissions NM PDU transmissions. This parameter is only valid if CanNmImmediateNmTransmissions is greater one Dependencies: ► This parameter is active only if `CanNmImmediateNmTransmissions` greater than one ► The transmission of the first NM PDU shall be delayed by the time indicated by `CANNM_MSG_CYCLE_OFFSET` in order to avoid bursts of NM messages if `CanNmImmediateNmTransmissions` is zero
Multiplicity	1..1

Type	FLOAT	
Default value	0.001	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmImmediateNmTransmissions**	
Label	Number of immediate NM PDUs	
Description	Defines the number of immediate NM PDUs which shall be transmitted. If the value is zero no immediate NM PDUs are transmitted. The cycle time of immediate NM PDUs is defined by `CanNmImmediateNmCycleTime`. Dependencies: ► `CanNmImmediateNmCycleTime` is active only if this parameter greater than zero ► The transmission of the first NM PDU shall be delayed by the time indicated by `ANNM_MSG_CYCLE_OFFSET` in order to avoid bursts of NM messages if CanNmImmediateNmTransmissions is zero	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmMsgCycleOffset**
Label	Message Cycle Offset [s]
Description	Time offset in seconds of the periodic transmission. It determines the start delay of the transmission. Dependencies: ► If Passive Mode is enabled this parameter is ignored. ► The Message Cycle Offset must be smaller than the Message Cycle Time. ► The value must be multiple of the Main Function Period.
Multiplicity	1..1
Type	FLOAT
Default value	0.001

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmMsgCycleTime**	
Label	Message Cycle Time [s]	
Description	Period of a NM message in seconds. It determines the periodic rate in the 'periodic transmission mode with bus load reduction'; and is the basis for transmit scheduling in the 'periodic transmission mode without bus load reduction'. Dependencies: ► If Passive Mode is enabled this parameter is ignored. ► The value must be multiple of the Main Function Period.	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.002	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmMsgReducedTime**
Label	Message Reduced Time [s]
Description	Node specific bus cycle time in the periodic transmission mode with bus load reduction. Dependencies: ► This parameter is only valid if `CanNmBusLoadReductionEnabled == True` and `CanNmBusLoadReductionActive == True` and `CanNmPassiveModeEnabled == False` ► Value must be smaller than the Message Cycle Time of this NM channel. ► Value must be greater than or equal to half the Message Cycle Time of this NM channel. ► Value must be a multiple of the Main Function Period.
Multiplicity	1..1
Type	FLOAT
Default value	0.001

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmMsgTimeoutTime**	
Label	Message Timeout Time [s]	
Description	Transmission Timeout of NM message. If there is no transmission confirmation by the CAN Interface within this timeout, the CanNm module shall give an error notification. Dependencies: ► If Passive Mode is enabled this parameter is ignored. ► Value must be a multiple of the Main Function Period. ► Value must be less than the Message Cycle Time.	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.002	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmNodeId**	
Label	Node Identifier	
Description	Node identifier of local node. Dependencies: ► This parameter is only valid if `CanNmPassiveModeEnabled = False` ► If the Node Identifier Position is set to *CANNM_PDU_OFF* this parameter is ignored.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=255 >=0	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanNmPduCbvPosition	
Label	PDU Control Bit Vector Position	
Description	This parameter defines the position of the Control Bit Vector within a NM message: ► CANNM_PDU_BYTE_0: byte 0 ► CANNM_PDU_BYTE_1: byte 1, ► CANNM_PDU_OFF: Control bit vector is not part of the NM PDU Dependencies: ► If Node Detection support is disabled this parameter is ignored. ► The Control Bit Vector must not occupy the same byte as the Node Id.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	CANNM_PDU_BYTE_1	
Range	CANNM_PDU_BYTE_0 CANNM_PDU_BYTE_1 CANNM_PDU_OFF	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanNmPduNidPosition
Label	PDU Node Identifier Position
Description	This parameter defines the position of the Node Id within a NM message: ► CANNM_PDU_BYTE_0: byte 0 ► CANNM_PDU_BYTE_1: byte 1, ► CANNM_PDU_OFF: source node identifier is not part of the NM PDU Dependencies: ► If Node Id support is disabled this parameter is ignored. ► The Node Id must not occupy the same byte as the Control Bit Vector
Multiplicity	1..1
Type	ENUMERATION
Default value	CANNM_PDU_BYTE_0

Range	CANNM_PDU_BYTE_0	
	CANNM_PDU_BYTE_1	
	CANNM_PDU_OFF	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmRemoteSleepIndTime**	
Label	Remote Sleep Indication Time [s]	
Description	Timeout for Remote Sleep Indication. It defines the time in seconds how long it shall take to recognize that all other nodes are ready to sleep. Dependencies: ► CanNmRemoteSleepIndTime >= CanNmMsgCycleTime	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.000	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmRepeatMessageTime**
Label	Repeat Message Time [s]
Description	Timeout for Repeat Message State. It defines the time in seconds how long the NM shall stay in the Repeat Message State. Typically the value of this parameter should be a multiple of Message Cycle Time. The value 0 denotes that *no Repeat Message State* is configured. It means that Repeat Message State is transient what implicates that it is left immediately after entrance and in result no start-up stability is guaranteed and no node detection procedure is possible. Dependencies: ► Value must be a multiple of the Main Function Period.

Multiplicity	1..1	
Type	FLOAT	
Default value	0.001	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmTimeoutTime**
Label	Timeout Time [s]
Description	Network Timeout for NM messages. It specifies the time in seconds how long the NM stays in the Network Mode before transition to Prepare Bus-Sleep Mode takes place after the network has been released. Transition to Prepare Bus-Sleep Mode take place if ► the network has been released and ► no NM messages are received within this period It shall be equal for all nodes in the cluster. It shall be greater than CanNmMsgCycleTime. Dependencies: ► Value must be a multiple of the Main Function Period.
Multiplicity	1..1
Type	FLOAT
Default value	0.004
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanNmUserDataLength**
Label	User Data Length
Description	Defines the length of the user data contained in the NM PDU Dependencies: ► The size of the user data must be smaller than the length of the PDU with length of the Control Bit Vector and/or the Node Id subtracted.
Multiplicity	1..1

Type	INTEGER	
Default value	6	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmWaitBusSleepTime**	
Label	Wait Bus Sleep Time [s]	
Description	Timeout for bus calm down phase. This parameter specifies the time in seconds how long the NM shall stay in the Prepare Bus-Sleep Mode before transition into Bus-Sleep Mode takes place. Transition to Prepare Bus-Sleep Mode take place if ► the network has not been requested again and ► no NM messages are received within this period Typically the value of this parameter should be a multiple of Message Cycle Time. It shall be equal for all nodes in the cluster. It shall be long enough to make all Tx-buffer empty. Dependencies: ► Value must be a multiple of the Main Function Period.	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.004	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmComMNetworkHandleRef**	
Description	This reference points to the unique channel defined by the ComMChannel and provides access to the unique channel index value in ComMChannelId. Dependencies:	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.3.1.4. CanNmRxPdu

Parameters included	
Parameter name	**Multiplicity**
CanNmRxPduId	1..1
CanNmRxPduRef	1..1

Parameter Name	**CanNmRxPduId**	
Label	Receive PDU ID	
Description	This parameter defines the Rx PDU ID of the CanIf L-PDU range that is associated with this CanNmChannel instance.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmRxPduRef**	
Label	Receive PDU Reference	
Description	Reference to the global PDU that is used by this CanNmChannel instance.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.3.1.5. CanNmTxPdu

Parameters included	
Parameter name	**Multiplicity**
CanNmTxConfirmationPduId	1..1
CanNmTxPduRef	1..1

Parameter Name	**CanNmTxConfirmationPduId**
Description	Handle Id to be used by the Lower Layer (CanIf) to confirm the transmission of the `CanNmTxPdu`.

Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmTxPduRef**
Label	Transmit PDU Reference
Description	The reference to a common PDU structure used for transmission of NM messages.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.3.1.6. CanNmUserDataTxPdu

Parameters included	
Parameter name	**Multiplicity**
CanNmTxUserDataPduId	1..1
CanNmTxUserDataPduRef	1..1

Parameter Name	**CanNmTxUserDataPduId**
Description	This parameter defines the handle ID of the Tx NM User Data I-PDU. The handle ID is used by PduR to invoke `CanNm_Transmit()`.
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanNmTxUserDataPduRef**
Description	Reference to the Tx NM User Data I-PDU in the global PDU collection
Multiplicity	1..1
Type	REFERENCE

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.3.1.7. CanNmUserDataRxPdu

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

Parameter Name	**CanNmRxUserDataPduRef**	
Description	Reference to the Rx NM User Data I-PDU in the global PDU collection	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.3.1.8. CanNmPnInfo

Containers included		
Container name	**Multiplicity**	**Description**
CanNmPnFilterMaskByte	0..7	PN information configuration.

Parameters included	
Parameter name	**Multiplicity**
CanNmPnInfoLength	1..1
CanNmPnInfoOffset	1..1

Parameter Name	**CanNmPnInfoLength**
Description	Specifies the length of the PN request information in the NM message.
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	<=7

	>=1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmPnInfoOffset**	
Description	Specifies the offset of the PN request information in the NM message.	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Range	<=31 >=1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.3.1.9. CanNmPnFilterMaskByte

Parameters included	
Parameter name	**Multiplicity**
CanNmPnFilterMaskByteIndex	1..1
CanNmPnFilterMaskByteValue	1..1

Parameter Name	**CanNmPnFilterMaskByteIndex**	
Description	Specifies the offset of the PN request information in the NM message.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanNmPnFilterMaskByteValue**
Description	Parameter to configure the filter mask byte.
Multiplicity	1..1
Type	INTEGER

Default value	0	
Range	<=255 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.3.1.10. CanNmDefensiveProgramming

Parameters included	
Parameter name	**Multiplicity**
CanNmDefProgEnabled	1..1
CanNmPrecondAssertEnabled	1..1
CanNmPostcondAssertEnabled	1..1
CanNmStaticAssertEnabled	1..1
CanNmUnreachAssertEnabled	1..1
CanNmInvariantAssertEnabled	1..1

Parameter Name	**CanNmDefProgEnabled**	
Label	Enable Defensive Programming	
Description	Enables or disables the defensive programming feature for the module CanNm. 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	**CanNmPrecondAssertEnabled**

Label	Enable Precondition Assertions	
Description	Enables handling of precondition assertion checks reported from the module CanNm. Dependency on parameter(s): ► Enable Development Error Detection (`CanNmDevErrorDetect`): must be enabled ► Enable Defensive Programming (`CanNmDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

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

Parameter Name	**CanNmStaticAssertEnabled**
Label	Enable Static Assertions
Description	Enables handling of static assertion checks reported from the module CanNm. Dependency on parameter(s):

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

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

Parameter Name	**CanNmInvariantAssertEnabled**
Label	Enable Invariant Assertions
Description	Enables handling of invariant assertion checks reported from functions of the module CanNm. Dependency on parameter(s): ► Enable Development Error Detection (`CanNmDevErrorDetect`): must be enabled ► Enable Defensive Programming (`CanNmDefProgEnabled`): must be enabled

Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.3.1.11. 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	3	
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	6
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	19
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	8
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	31
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

5.3.1.12. PublishedInformation

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

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

5.3.2. Application programming interface (API)

5.3.2.1. Macro constants

5.3.2.1.1. CANNM_API_ID_RXINDICATION

Purpose	CanNM API ID.
Value	16U
Description	Definition of CANNM_API_ID_RXINDICATION.

5.3.2.1.2. CANNM_API_ID_TXCONFIRMATION

Purpose	CanNM API ID.
Value	15U
Description	Definition of CANNM_API_ID_TXCONFIRMATION.

5.3.2.1.3. CANNM_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
Value	4U

5.3.2.1.4. CANNM_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
Value	0U

5.3.2.1.5. CANNM_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.
Value	3U

5.3.2.1.6. CANNM_E_BUSSLEEPMODE

Purpose	Error code for case in which SchM_Call for Nm_BusSleepMode fails.
Value	248U

5.3.2.1.7. CANNM_E_CARWAKEUPINDICATION

Purpose	Error code for case in which SchM_Call for Nm_CarWakeUpIndication fails.
Value	240U

5.3.2.1.8. CANNM_E_INIT_FAILED

Purpose	Error code for the case in which CanNm initialization fails.

Value	0x05U

5.3.2.1.9. CANNM_E_INVALID_CHANNEL

Purpose	Error Code for Invalid channel.
Value	0x02U

5.3.2.1.10. CANNM_E_INVALID_FUNCTION_ARG

Purpose	Error code for other invalid API function argument in API.
Value	0x23U

5.3.2.1.11. CANNM_E_INVALID_PDUID

Purpose	Error code for Invalid PDU Id.
Value	0x21U

5.3.2.1.12. CANNM_E_NETWORKMODE

Purpose	Error code for case in which SchM_Call for Nm_NetworkMode fails.
Value	249U

5.3.2.1.13. CANNM_E_NETWORKSTARTINDICATION

Purpose	Error code for case in which SchM_Call for Nm_NetworkStartIndication fails.
Value	250U

5.3.2.1.14. CANNM_E_NETWORK_TIMEOUT

Purpose	Error code for unexpected timeout of NM timer.

Value	0x11U

5.3.2.1.15. CANNM_E_NET_START_IND

Purpose	Error code for Reception of NM messages in Bus-Sleep Mode.
Value	0x04U

5.3.2.1.16. CANNM_E_NO_INIT

Purpose	Initialization status before module initilaization.
Value	0x01U

5.3.2.1.17. CANNM_E_NULL_POINTER

Purpose	Error code for NULL pointers.
Value	0x12U

5.3.2.1.18. CANNM_E_PDURXINDICATION

Purpose	Error code for case in which SchM_Call for Nm_PduRxIndication fails.
Value	244U

5.3.2.1.19. CANNM_E_PREPAREBUSSLEEPMODE

Purpose	Error code for case in which SchM_Call for Nm_PrepareBusSleepMode fails.
Value	247U

5.3.2.1.20. CANNM_E_REMOTESLEEPCANCELLATION

Purpose	Error code for case in which SchM_Call for Nm_RemoteSleepCancellation fails.
Value	245U

5.3.2.1.21. CANNM_E_REMOTESLEEPINDICATION

Purpose	Error code for case in which SchM_Call for Nm_RemoteSleepIndication fails.
Value	246U

5.3.2.1.22. CANNM_E_REPEATMESSAGEINDICATION

Purpose	Error code for case in which SchM_Call for Nm_RepeatMessageIndication fails.
Value	242U

5.3.2.1.23. CANNM_E_STATECHANGENOTIFICATION

Purpose	Error code for case in which SchM_Call for Nm_StateChangeNotification fails.
Value	243U

5.3.2.1.24. CANNM_E_TXTIMEOUTEXCEPTION

Purpose	Error code for case in which SchM_Call for Nm_TxTimeoutException fails.
Value	241U

5.3.2.1.25. CANNM_INSTANCE_ID

Purpose	Instance Id of CanNm.
Value	0U

5.3.2.1.26. CANNM_INVALID_PDU_INSTANCE_ID

Purpose	Instance Id of CanNm when an invalid PDU is passed.
Value	255U

5.3.2.1.27. CANNM_MODULE_ID

Purpose	AUTOSAR module identification.

Value	31U

5.3.2.1.28. CANNM_PDU_BYTE_0

Purpose	AUTOSAR API service ID.
Value	0U
Description	Definition of CANNM_PDU_BYTE_0.

5.3.2.1.29. CANNM_PDU_BYTE_1

Purpose	AUTOSAR API service ID.
Value	1U
Description	Definition of CANNM_PDU_BYTE_1.

5.3.2.1.30. CANNM_PDU_OFF

Purpose	AUTOSAR API service ID.
Value	3U
Description	Definition of CANNM_PDU_OFF.

5.3.2.1.31. CANNM_SERVID_CANNMTRANSMIT

Purpose	AUTOSAR API service ID.
Value	0x14U
Description	Definition of CANNM_SERVID_CANNMTRANSMIT.

5.3.2.1.32. CANNM_SERVID_CHECKREMOTESLEEPINDICATION

Purpose	AUTOSAR API service ID.
Value	0xD0U
Description	Definition of CANNM_SERVID_CHECKREMOTESLEEPINDICATION.

5.3.2.1.33. CANNM_SERVID_CONFIRMPNAVAILABILITY

Purpose	AUTOSAR API service ID.
Value	0x16U
Description	Definition of CANNM_SERVID_CONFIRMPNAVAILABILITY.

5.3.2.1.34. CANNM_SERVID_DISABLECOMMUNICATION

Purpose	AUTOSAR API service ID.
Value	0x0CU
Description	Definition of CANNM_SERVID_DISABLECOMMUNICATION.

5.3.2.1.35. CANNM_SERVID_ENABLECOMMUNICATION

Purpose	AUTOSAR API service ID.
Value	0x0DU
Description	Definition of CANNM_SERVID_ENABLECOMMUNICATION.

5.3.2.1.36. CANNM_SERVID_GETLOCALNODEIDENTIFIER

Purpose	AUTOSAR API service ID.
Value	0x07U
Description	Definition of CANNM_SERVID_GETLOCALNODEIDENTIFIER.

5.3.2.1.37. CANNM_SERVID_GETNODEIDENTIFIER

Purpose	AUTOSAR API service ID.
Value	0x06U
Description	Definition of CANNM_SERVID_GETNODEIDENTIFIER.

5.3.2.1.38. CANNM_SERVID_GETPDUDATA

Purpose	AUTOSAR API service ID.

Value	0x0AU
Description	Definition of CANNM_SERVID_GETPDUDATA.

5.3.2.1.39. CANNM_SERVID_GETSTATE

Purpose	AUTOSAR API service ID.
Value	0x0BU
Description	Definition of CANNM_SERVID_GETSTATE.

5.3.2.1.40. CANNM_SERVID_GETUSERDATA

Purpose	AUTOSAR API service ID.
Value	0x05U
Description	Definition of CANNM_SERVID_GETUSERDATA.

5.3.2.1.41. CANNM_SERVID_GETVERSIONINFO

Purpose	AUTOSAR API service ID.
Value	0xF1U
Description	Definition of CANNM_SERVID_GETVERSIONINFO.

5.3.2.1.42. CANNM_SERVID_INIT

Purpose	AUTOSAR API service ID.
Value	0x00U
Description	Definition of CANNM_SERVID_INIT.

5.3.2.1.43. CANNM_SERVID_MAINFUNCTION_X

Purpose	AUTOSAR API service ID.
Value	0x13U

Description	Definition of CANNM_SERVID_MAINFUNCTION_X.

5.3.2.1.44. CANNM_SERVID_NETWORKGWERAREQUEST

Purpose	AUTOSAR API service ID.
Value	0xFEU
Description	Definition of CANNM_SERVID_NETWORKGWERAREQUEST.

5.3.2.1.45. CANNM_SERVID_NETWORKRELEASE

Purpose	AUTOSAR API service ID.
Value	0x03U
Description	Definition of CANNM_SERVID_NETWORKRELEASE.

5.3.2.1.46. CANNM_SERVID_NETWORKREQUEST

Purpose	AUTOSAR API service ID.
Value	0x02U
Description	Definition of CANNM_SERVID_NETWORKREQUEST.

5.3.2.1.47. CANNM_SERVID_PASSIVESTARTUP

Purpose	AUTOSAR API service ID.
Value	0x01U
Description	Definition of CANNM_SERVID_PASSIVESTARTUP

5.3.2.1.48. CANNM_SERVID_REPEATMESSAGEREQUEST

Purpose	AUTOSAR API service ID.
Value	0x08U
Description	Definition of CANNM_SERVID_REPEATMESSAGEREQUEST.

5.3.2.1.49. CANNM_SERVID_REQUESTBUSSYNCHRONIZATION

Purpose	AUTOSAR API service ID.
Value	0xC0U
Description	Definition of CANNM_SERVID_REQUESTBUSSYNCHRONIZATION.

5.3.2.1.50. CANNM_SERVID_RXINDICATION

Purpose	AUTOSAR API service ID.
Value	0x10U
Description	Definition of CANNM_SERVID_RXINDICATION.

5.3.2.1.51. CANNM_SERVID_SETUSERDATA

Purpose	AUTOSAR API service ID.
Value	0x04U
Description	Definition of CANNM_SERVID_SETUSERDATA.

5.3.2.1.52. CANNM_SERVID_TXCONFIRMATION

Purpose	AUTOSAR API service ID.
Value	0x0FU
Description	Definition of CANNM_SERVID_TXCONFIRMATION.

5.3.2.1.53. CANNM_SERVID_TXTIMEOUTEXCEPTION

Purpose	AUTOSAR API service ID.
Value	0x27U
Description	Definition of CANNM_SERVID_TXTIMEOUTEXCEPTION.

5.3.2.1.54. CANNM_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.

Value	6U

5.3.2.1.55. CANNM_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	19U

5.3.2.1.56. CANNM_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	8U

5.3.2.1.57. CANNM_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

5.3.2.2. Functions

5.3.2.2.1. CanNm_CheckRemoteSleepIndication

Purpose	Check if sleep indication has taken place.	
Synopsis	`Std_ReturnType` **`CanNm_CheckRemoteSleepIndication`** `( NetworkHandleType nmChannelHandle , boolean * nmRemoteSleepIndPtr );`	
Service ID	208	
Sync/Async	Asynchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`nmChannelHandle`	Identification of the NM-channel.
Parameters (out)	`nmRemoteSleepIndPtr`	Pointer where check result of remote sleep indication shall be copied to.
Return Value	Standard Return Code	
	`E_OK`	No Error.

	E_NOT_OK	Checking of remote sleep indication bits has failed/not executed.
Description	This function checks if remote sleep indication has taken place or not.	

5.3.2.2.2. CanNm_ConfirmPnAvailability

Purpose	Enable PN Filtering.	
Synopsis	`void` **`CanNm_ConfirmPnAvailability`** `( NetworkHandleType nmChannelHandle );`	
Service ID	0x16	
Sync/Async	Synchronous	
Reentrancy	Reentrant (but not for the same NM Channel)	
Parameters (in)	nmChannelHandle	Identification of the NM channel.
Description	Enables the PN filter functionality on the indicated NM channel. Availability: Preconditions: ► The API is only available if CanNmPnSupported is TRUE.	

5.3.2.2.3. CanNm_DisableCommunication

Purpose	Disable NM PDU transmission.	
Synopsis	`Std_ReturnType` **`CanNm_DisableCommunication`** `( NetworkHandleType nmChannelHandle );`	
Service ID	12	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (But not for the same NM Channel)	
Parameters (in)	nmChannelHandle	Identification of the NM channel.
Return Value	Standard Return Code	
	E_OK	No Error.
	E_NOT_OK	Disabling of NM PDU transmission ability has failed/not executed.
Description	This function disables the NM PDU transmission ability due to a ISO14229 Communication Control (28hex) service.	

5.3.2.2.4. CanNm_EnableCommunication

Purpose	Enable NM PDU transmission.	
Synopsis	`Std_ReturnType` **`CanNm_EnableCommunication`** `( NetworkHandleType nmChannelHandle );`	
Service ID	13	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (But not for the same NM Channel)	
Parameters (in)	`nmChannelHandle`	Identification of the NM channel.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Enabling of NM PDU transmission ability has failed/not executed.
Description	This function enables the NM PDU transmission ability due to a ISO14229 Communication Control (28hex) service.	

5.3.2.2.5. CanNm_GetLocalNodeIdentifier

Purpose	Get Local Node Identifier.	
Synopsis	`Std_ReturnType` **`CanNm_GetLocalNodeIdentifier`** `( NetworkHandleType nmChannelHandle , uint8 * nmNodeIdPtr );`	
Service ID	7	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`nmChannelHandle`	Identification of the NM channel.
Parameters (out)	`nmNodeIdPtr`	Pointer where node identifier of the local node shall be copied to.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Getting of the node identifier of the local node has failed.
Description	This function gets the node identifier configured as the local node.	

5.3.2.2.6. CanNm_GetNodeIdentifier

Purpose	Get Node Identifier.	
Synopsis	`Std_ReturnType` **`CanNm_GetNodeIdentifier`** `( NetworkHandleType nmChannelHandle , uint8 * nmNodeIdPtr );`	
Service ID	6	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`nmChannelHandle`	Identification of the NM channel.
Parameters (out)	`nmNodeIdPtr`	Pointer where node identifier out of the most recently received NM PDU shall be copied to.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Getting of the node identifier out of the most recently received NM PDU has failed.
Description	This function gets the node identifier out of the most recently received NM PDU.	

5.3.2.2.7. CanNm_GetPduData

Purpose	Retrieve the data of the last received NM message.	
Synopsis	`Std_ReturnType` **`CanNm_GetPduData`** `( NetworkHandleType nmChannelHandle , uint8 * nmPduDataPtr );`	
Service ID	10	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`nmChannelHandle`	Identification of the NM channel.
	`nmPduDataPtr`	Pointer where NM PDU data shall be copied to.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Getting of NM PDU data has failed.
Description	This function retrieves the whole PDU data out of the most recently received NM message.	

	Preconditions: ► The channel handle should be valid and the module should have been initialized for this channel (checked).

5.3.2.2.8. CanNm_GetState

Purpose	Get the State and mode of the Network Management.	
Synopsis	`Std_ReturnType` **`CanNm_GetState`** `( NetworkHandleType nmChannelHandle , Nm_StateType * nmStatePtr , Nm_ModeType * nmModePtr );`	
Service ID	11	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`nmChannelHandle`	Identification of the NM-channel.
Parameters (out)	`nmStatePtr`	Pointer to state of network management.
	`nmModePtr`	Pointer to mode of network management.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Getting of NM state has failed.
Description	This function returns the state and the mode of the network management.	

5.3.2.2.9. CanNm_GetUserData

Purpose	Get User Data from NM messages.	
Synopsis	`Std_ReturnType` **`CanNm_GetUserData`** `( NetworkHandleType nmChannelHandle , uint8 * nmUserDataPtr );`	
Service ID	5	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`nmChannelHandle`	Identification of the NM channel.
	`nmUserDataPtr`	Pointer to where user data out of the most recently received NM message shall be copied to.
Return Value	Standard Return Code	

	`E_OK`	No Error.
	`E_NOT_OK`	Getting of user data has failed.
Description	This function retrieves the user data from the last received NM message. Preconditions: ► The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.3.2.2.10. CanNm_GetVersionInfo

Purpose	Get version information for the CAN Network Management.	
Synopsis	`void` **`CanNm_GetVersionInfo`** `( Std_VersionInfoType * versioninfo );`	
Service ID	241	
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: ► Module Id ► Vendor Id ► Vendor specific version numbers (BSW00407). Note: This function can be called even if CanNm is not initialized.	

5.3.2.2.11. CanNm_Init

Purpose	Initialization of CanNm module.
Synopsis	`void` **`CanNm_Init`** `( const CanNm_ConfigType * cannmConfigPtr );`
Service ID	1
Sync/Async	Synchronous
Reentrancy	Non-Reentrant

Parameters (in)	`cannmConfigPtr`	Pointer to selected configuration structure.
Description	This function initializes the CanNm module and starts the cyclic transmission of NM-packages. This function has to be called after initialization of the CanIf.	

5.3.2.2.12. CanNm_IsValidConfig

Purpose	Validate configuration.
Synopsis	`Std_ReturnType` **`CanNm_IsValidConfig`** `( const void * voidConfigPtr );`
Service ID	0x60
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.

5.3.2.2.13. CanNm_MainFunction

Purpose	Main function of the CanNm.
Synopsis	`void` **`CanNm_MainFunction`** `( void );`
Service ID	19
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This function handles scheduled tasks such as timers.

5.3.2.2.14. CanNm_NetworkGwEraRequest

Purpose	Network Gateway Era Request.
Synopsis	`Std_ReturnType` **`CanNm_NetworkGwEraRequest`** `( NetworkHandleType nmChannelHandle );`
Service ID	254

Sync/Async	Asynchronous	
Reentrancy	Reentrant (But not for the same NM Channel)	
Parameters (in)	`nmChannelHandle`	Identification of the NM channel.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Requesting of network has failed.
Description	This function request the network when bus communication is needed. Network state shall be changed to requested. If function is called active wakeup bit is not set	

5.3.2.2.15. CanNm_NetworkRelease

Purpose	Release the Network.	
Synopsis	`Std_ReturnType` **`CanNm_NetworkRelease`** `( NetworkHandleType nmChannelHandle );`	
Service ID	3	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (But not for the same NM Channel)	
Parameters (in)	`nmChannelHandle`	Identification of the NM channel.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Releasing of network has failed/not executed.
Description	This function releases the network, when there is no need for bus communication. Network state shall be changed to released.	

5.3.2.2.16. CanNm_NetworkRequest

Purpose	Network Request.
Synopsis	`Std_ReturnType` **`CanNm_NetworkRequest`** `( NetworkHandleType nmChannelHandle );`
Service ID	2
Sync/Async	Asynchronous
Reentrancy	Reentrant (But not for the same NM Channel)

Parameters (in)	`nmChannelHandle`	Identification of the NM channel.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Requesting of network has failed.
Description	This function request the network when bus communication is needed. Network state shall be changed to requested. If function is called active wakeup bit is set	

5.3.2.2.17. CanNm_PassiveStartUp

Purpose	Passive startup of CanNm module.	
Synopsis	`Std_ReturnType` **`CanNm_PassiveStartUp`** `( NetworkHandleType nmChannelHandle );`	
Service ID	1	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (But not for the same NM Channel)	
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Passive startup of network management has failed/not executed.
Description	This function performs a passive startup of the AUTOSAR CAN NM. It triggers the transition from Bus-Sleep Mode to the Network Mode in Repeat Message State. This service has no effect if the current state is not equal to Bus-Sleep Mode. In that case E_NOT_OK is returned.	

5.3.2.2.18. CanNm_RepeatMessageRequest

Purpose	Set the Repeat Message Request Bit.	
Synopsis	`Std_ReturnType` **`CanNm_RepeatMessageRequest`** `( NetworkHandleType nmChannelHandle );`	
Service ID	8	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (but not for the same NM Channel)	
Parameters (in)	`nmChannelHandle`	Identification of the NM channel.

Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Setting of Repeat Message Request Bit has failed/not executed.
Description	This function sets the Repeat Message Request Bit for NM messages transmitted next on the bus.	

5.3.2.2.19. CanNm_RequestBusSynchronization

Purpose	Request Bus Synchorization.	
Synopsis	`Std_ReturnType` **`CanNm_RequestBusSynchronization`** `( NetworkHandleType nmChannelHandle );`	
Service ID	192	
Sync/Async	synchronous	
Reentrancy	Reentrant (but not for the same NM Channel)	
Parameters (in)	`nmChannelHandle`	Identification of the NM-channel.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Requesting of bus synchronization has failed/not executed.
Description	This function requests bus synchronization.	

5.3.2.2.20. CanNm_RxIndication

Purpose	Indicates a received transmission.	
Synopsis	`void` **`CanNm_RxIndication`** `( PduIdType RxPduId , PduInfoType * PduInfoPtr );`	
Parameters (in)	`RxPduId`	Identification of the network through PDU-ID.
	`PduInfoPtr`	Contains the length of the received I-PDU and a pointer to a buffer containing the I-PDU.
Description	This function indicates the reception of an NM-message PDU.	

5.3.2.2.21. CanNm_SetUserData

Purpose	Set User Data for NM messages.	
Synopsis	`Std_ReturnType` **`CanNm_SetUserData`** `( NetworkHandleType nmChannelHandle , const uint8 * nmUserDataPtr );`	
Service ID	4	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`nmChannelHandle`	Identification of the NM channel.
	`nmUserDataPtr`	Pointer where the user data for the next transmitted NM message shall be copied from.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Setting of user data has failed.
Description	This function sets the user data for the next NM message that is transmitted on the bus. Preconditions: ► The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.3.2.2.22. CanNm_Transmit

Purpose	Dummy function.	
Synopsis	`Std_ReturnType` **`CanNm_Transmit`** `( PduIdType CanTxPduId , const PduInfoType * PduInfoPtr );`	
Service ID	0	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`CanTxPduId`	Identification of the NM channel.
	`PduInfoPtr`	Pointer to a structure with CAN L-PDU related data: DLC and pointer to CAN L-SDU buffer
Return Value	Standard Return Code	

	`E_OK`	always
Description	CanNm_Transmit is implemented as an empty function and always returns E_OK. The function CanNm_Transmit is only available if the configuration switch CanNmComUserDataSupport is enabled. Preconditions: ► None	

5.3.2.2.23. CanNm_TxConfirmation

Purpose	Confirms a transmission.	
Synopsis	`void` **`CanNm_TxConfirmation`** `( PduIdType TxPduId );`	
Parameters (in)	`TxPduId`	Identification of the network through PDU-ID.
Description	This function confirms the transmission of a NM-package.	

5.3.3. Integration notes

5.3.3.1. Exclusive areas

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

5.3.3.1.1. SCHM_CANNM_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.

`CanNm` uses exclusive areas for protecting the global data against concurrent read/write access:

- The status of `CanNm` channels - the consistency of this global variable must be assured as it can be read/written by the `CanNm` state machine and/or following user interfaces:
  - `CanNm_NetworkRequest()`
  - `CanNm_NetworkRelease()`
  - `CanNm_EnableCommunication()`
  - `CanNm_DisableCommunication()`
- The partial networking bits - the consistency of this global data must be assured as it can be read/written by the `CanNm` state machine, `RxIndication` and/or following user interface:
  - `CanNm_GetPduUserData()`
- The NM PDU data - the consistency of the PDU data must be assured as it can be read/written by the `RxIndication` and/or following interfaces:
  - `CanNm_GetUserData()`
  - `CanNm_GetPduData()`

5.3.3.2. Production errors

Production errors are not reported by the `CanNm` 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
CONST_8
CONST_32
VAR_INIT_8
CONST_UNSPECIFIED
CODE
VAR_INIT_UNSPECIFIED
VAR_CLEARED_8
VAR_CLEARED_UNSPECIFIED

VAR_INIT_BOOLEAN
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 guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the CanNm module.

5.4. CanSM

5.4.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CanSMDefensiveProgramming	1..1	**Label:** Defensive Programming Options Parameters for defensive programming
CanSMConfiguration	1..n	This container contains the global parameters of the CanSM and sub containers, which are for the CAN network specific configuration.
ReportToDem	1..1	**Label:** Production error handling Production error handling
CanSMGeneral	1..1	**Label:** General CanSM Configuration Container for general pre-compile parameters of the CanSM module.
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

Containers included		
		Additional published parameters not covered by Common-PublishedInformation 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

5.4.1.1. CanSMDefensiveProgramming

Parameters included	
Parameter name	**Multiplicity**
CanSMDefProgEnabled	1..1
CanSMPrecondAssertEnabled	1..1
CanSMPostcondAssertEnabled	1..1
CanSMStaticAssertEnabled	1..1
CanSMUnreachAssertEnabled	1..1
CanSMInvariantAssertEnabled	1..1

Parameter Name	CanSMDefProgEnabled
Label	Enable Defensive Programming
Description	Enables or disables the defensive programming feature for the module CanSM. 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	**CanSMPrecondAssertEnabled**
Label	Enable Precondition Assertions
Description	Enables handling of precondition assertion checks reported from the module CanSM. Dependency on parameter(s): ► Enable Development Error Detection (`CanSMDevErrorDetect`): must be enabled ► Enable Defensive Programming (`CanSMDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanSMPostcondAssertEnabled**
Label	Enable Postcondition Assertions
Description	Enables handling of postcondition assertion checks reported from the module CanSM. Dependency on parameter(s): ► Enable Development Error Detection (`CanSMDevErrorDetect`): must be enabled ► Enable Defensive Programming (`CanSMDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false

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

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

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

Parameter Name	**CanSMInvariantAssertEnabled**
Label	Enable Invariant Assertions

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

5.4.1.2. CanSMConfiguration

Containers included		
Container name	**Multiplicity**	**Description**
CanSMManagerNetwork	1..255	This container contains the CAN network specific parameters of each CAN network. **Optimization Effect:** ► **ROM reduction (config):** Using only one container of this type reduces the ROM consumption of the module configuration. ► **RAM reduction (config):** Using only one container of this type reduces the RAM consumption of the module configuration. ► **ROM reduction (code):** Using only one container of this type reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Using only one container of this type reduces the execution time of the module code.

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
CanSMModeRequestRepetitionMax	1..1
CanSMModeRequestRepetitionTime	1..1

Parameter Name	CanSMModeRequestRepetitionMax	
Label	Max. number of mode requests	
Description	Specifies the maximal amount of mode request repetitions without a respective mode indication from the CanIf module until the CanSM module reports a development error to the DET and tries to go back to no communication.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=255 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanSMModeRequestRepetitionTime	
Label	Mode request repetition time	
Description	Specifies in which time duration the CanSM module shall repeat mode change requests by using the API of the CanIf module.	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.0	
Range	<=65.534 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.4.1.3. CanSMManagerNetwork

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

Containers included		
CanSMController	1..255	This container contains the controller IDs assigned to a CAN network. **Optimization Effect:** ▶ **ROM reduction (config):** Using a single controller for each network reduces the ROM consumption of the module configuration. ▶ **ROM reduction (code):** Using a single controller for each network reduces the ROM consumption of the module code. ▶ **Execution time reduction (code):** Using a single controller for each network reduces the execution time of the module code.
CanSMDemEventParameterRefs	0..1	**Label:** Dem Events Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.

Parameters included	
Parameter name	**Multiplicity**
CanSMBorCounterL1ToL2	1..1
CanSMBorTimeL1	1..1
CanSMBorTimeL2	1..1
CanSMBorTimeTxEnsured	1..1
CanSMBorTxConfirmationPolling	1..1
CanSMComMNetworkHandleRef	1..1
CanSMTransceiverId	0..1
CanSMActivatePN	1..1

Parameter Name	CanSMBorCounterL1ToL2
Label	BOR L1 to L2 Threshold
Description	If the count of bus-offs is *greater than or equal to* this threshold, the bus-off recovery switches from level 1 (short recovery time) to level 2 (long recovery time).

	Remark: By comparison, the "BOR L2 Error Reporting Threshold" is a "greater than" threshold. **Optimization Effect:** ► **ROM reduction (config):** Using the same value for all networks reduces the ROM consumption of the module configuration. ► **ROM reduction (code):** Using the same value for all networks reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Using the same value for all networks reduces the execution time of the module code.
Multiplicity	1..1
Type	INTEGER
Default value	2
Range	<=255 >=0
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanSMBorTimeL1**
Label	BOR L1 Recovery Time [s]
Description	This time parameter defines in seconds the duration of the bus-off recovery time in level 1 (short recovery time). **Optimization Effect:** ► **ROM reduction (config):** Using the same value for all networks reduces the ROM consumption of the module configuration. ► **ROM reduction (code):** Using the same value for all networks reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Using the same value for all networks reduces the execution time of the module code.
Multiplicity	1..1
Type	FLOAT
Default value	0.5
Range	<=65.535 >=0

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanSMBorTimeL2**	
Label	BOR L2 Recovery Time [s]	
Description	This time parameter defines in seconds the duration of the bus-off recovery time in level 2 (long recovery time). **Optimization Effect:** ► **ROM reduction (config):** Using the same value for all networks reduces the ROM consumption of the module configuration. ► **ROM reduction (code):** Using the same value for all networks reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Using the same value for all networks reduces the execution time of the module code.	
Multiplicity	1..1	
Type	FLOAT	
Default value	1.5	
Range	<=65.535 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanSMBorTimeTxEnsured**
Label	BOR Time Ensured [s]
Description	This parameter defines in seconds the duration of the bus-off event check. This parameter is ignored if parameter CanSMBorTxConfirmationPolling is enabled. This check assesses, if the recovery has been successful after the recovery reenables the transmit path. If a new bus-off occurs during this time period, the CanSM assesses this bus-off as sequential bus-off without successful recovery. Because a bus-off only can be detected, when PDUs are transmitted, the time has to be great enough to ensure that PDUs are transmitted again (e. g. time period of the fastest cyclic transmitted PDU of the COM module / ComTxModeTimePeriodFactor).

	Optimization Effect: ► **ROM reduction (config):** Using the same value for all networks reduces the ROM consumption of the module configuration. ► **ROM reduction (code):** Using the same value for all networks reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Using the same value for all networks reduces the execution time of the module code.
Multiplicity	1..1
Type	FLOAT
Default value	5.0
Range	<=65.534 >=0
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanSMBorTxConfirmationPolling**
Label	BOR Tx Confirmation Polling
Description	This parameter shall configure, if the CanSM polls the CanIf_GetTxConfirmationState API to decide the bus-off state to be recovered instead of using the CanSMBorTimeTxEnsured parameter for this decision.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanSMComMNetworkHandleRef**
Label	ComM Channel
Description	Unique handle to identify one certain CAN 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	CanSMTransceiverId	
Label	Can Transceiver Channel	
Description	ID of the CAN transceiver assigned to the configured network handle. Reference to one of the transceivers managed by the CanIf module. **Optimization Effect:** ► **ROM increase (config):** Using this parameter increases the ROM consumption of the module configuration. ► **ROM increase (code):** Using this parameter increases the ROM consumption of the module code. ► **Execution time increase (code):** Using this parameter increases the execution time of the module code.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanSMActivatePN	
Label	Activate PN for this network	
Description	Activate/deactivate the partial networking for this network. ► true: Partial Networking activated for this network ► false: Partial Networking deactivated for this network	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.4.1.4. CanSMController

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

Parameter Name	CanSMControllerId	
Label	CAN Controller	
Description	Unique handle to identify one certain CAN controller. Reference to one of the CAN controllers managed by the CanIf module.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.4.1.5. CanSMDemEventParameterRefs

Parameters included	
Parameter name	**Multiplicity**
CANSM_E_BUS_OFF	0..1

Parameter Name	CANSM_E_BUS_OFF
Label	CANSM_E_BUS_OFF
Description	Reference to the configured DEM event to report bus-off errors for this CAN network. Dependency on parameter(s): ► `CanSMBusOffReportToDem`: Select DEM to enable the reporting of `CANSM_E_BUS_OFF`. Further notes: ► Activation: Thrown if there is a notification of a bus-off event on a CAN controller. ► Healing: Will be healed if communication goes on within the configurable timespan of bus-off recovery timers `CanSMBorTimeL1` and `CanSMBorTimeL2`. ► Trigger debounce: None. The error is reported on first occurrence. ► Rate of diagnostic checks: Checked cyclically within `CanSM_MainFunction()`.
Multiplicity	0..1
Type	SYMBOLIC-NAME-REFERENCE

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.4.1.6. ReportToDem

Parameters included	
Parameter name	**Multiplicity**
CanSMBusOffReportToDem	1..1
CanSMBusOffReportToDemDetErrorId	1..1

Parameter Name	**CanSMBusOffReportToDem**	
Label	Bus off recovery state machine production error	
Description	Selects the handling of the production error: *bus off recovery state machine error* ► `DEM`: All errors are reported to the Diagnostics Event Manager (Dem). ► `DET`: All errors are reported to the Development Error Tracer (Det) if enabled. ► `DISABLE`: Production errors are 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	DEM	
Range	DEM DET DISABLE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanSMBusOffReportToDemDetErrorId**

Label	Bus off recovery state machine Det error ID
Description	If a production error is reported towards the Det, this parameter defines the error id of the production errors CANSM_E_BUS_OFF for all networks. The Det instance id is the ComM channel ID (parameter ComMChannelId of the ComM channel referenced by parameter CanSMComMNetworkHandleRef).
Multiplicity	1..1
Type	INTEGER
Default value	128
Range	>=0 --- <=255
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.4.1.7. CanSMGeneral

Parameters included	
Parameter name	**Multiplicity**
CanSMDevErrorDetect	1..1
CanSMMainFunctionTimePeriod	1..1
CanSMVersionInfoApi	1..1
CanSMChangeBaudrateApi	1..1
CanSMPNSupport	1..1
CanSMSetBaudrateApi	1..1
CanSMBusDeactivatedBusOff	1..1
CanSMEnhancedBusOffReporting	1..1
CanSMTxTimeoutExceptionApi	1..1
CanSMMultiCoreSupport	1..1
CanSMDistributedChannelProcessingSupport	1..1
CanSMMaxNumberOfTransceivers	1..1

Parameter Name	**CanSMDevErrorDetect**
Label	Enable Development Error Detection

Description	Enables and disables the development error detection and notification mechanism. **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	**CanSMMainFunctionTimePeriod**
Label	Main Function Period [s]
Description	This parameter defines the cycle time of the function `CanSM_MainFunction` in seconds.
Multiplicity	1..1
Type	FLOAT
Default value	0.02
Range	<=65.535 >=0.001
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanSMVersionInfoApi**
Label	Enable Version Info API
Description	Activate/deactivate the version information API (`CanSM_GetVersionInfo`). ► true: Version information API activated ► false: Version information API deactivated **Optimization Effect:** ► **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	AUTOSAR_ECUC	

Parameter Name	**CanSMChangeBaudrateApi**
Description	The support of the Can_ChangeBaudrate API is optional.. ► true: Can_ChangeBaudrate API shall be supported ► false:Can_ChangeBaudrate API is not supported **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter decreases the ROM consumption of the module code. **This feature is currently not supported.**
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanSMPNSupport**
Label	Enable Partial Networking
Description	Activate/deactivate the partial networking support. ► true: Partial Networking enabled ► false: Partial Networking disabled **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	**CanSMSetBaudrateApi**	
Label	Enable API function CanSM_SetBaudrate()	
Description	Activate/deactivate the API function CanSM_SetBaudrate(). ► true: CanSM_SetBaudrate() enabled ► false: CanSM_SetBaudrate() disabled **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	**CanSMBusDeactivatedBusOff**	
Label	Enables specific Bus Off handling	
Description	This parameter selects how Bus Off is handled. Effect: ► Enabled: Bus Off will trigger No Communication until controller is restarted. ► Disabled: Bus Off will trigger Silent Communication (AUTOSAR handling) until controller is restarted.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanSMEnhancedBusOffReporting**
Label	Enable enhanced bus-off reporting
Description	Activate/deactivate enhanced bus-off reporting to BswM. Concerning to enhanced bus-off reporting the range of enumeration CanSM_BswMCurrentState-

	Type as defined in chapter 8.2.3 of the CAN State Manager SWS has changed to: Range: CANSM_BSWM_NO_COMMUNICATION CANSM_BSWM_SILENT_COMMUNICATION CANSM_BSWM_FULL_COMMUNICATION CANSM_BSWM_BUS_OFF CANSM_BSWM_CHANGE_BAUDRATE CANSM_BSWM_BUS_OFF_L1 CANSM_BSWM_BUS_OFF_L2 If this parameter is enabled and a bus-off event occurs and the bus-off counter is lower than CanSMBorCounterL1ToL2, CanSM reports the value CANSM_BSWM_BUS_OFF_L1 to BswM. If a bus-off event occurs and the bus-off counter is greater than or equal to CanSMBorCounterL1ToL2, CanSM reports the value CANSM_BSWM_BUS_OFF_L2 to BswM. ► true: CanSM reports the values CANSM_BSWM_BUS_OFF_L1 or CANSM_BSWM_BUS_OFF_L2 in case of bus-off. ► false: CanSM always reports the value CANSM_BSWM_BUS_OFF in case of bus-off (= SWS defined behavior).
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanSMTxTimeoutExceptionApi**
Label	Enable API function CanSM_TxTimeoutException()
Description	Activate/deactivate the API function CanSM_TxTimeoutException(). ► true: CanSM_TxTimeoutException() enabled ► false: CanSM_TxTimeoutException() disabled **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	**CanSMMultiCoreSupport**	
Label	Enable CanSM MultiCore Support	
Description	This parameter enables CanSM support for multicore. Effect: ► Enabled: CanSM multicore support is enabled. ► Disabled: CanSM multicore support is disabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanSMDistributedChannelProcessingSupport**	
Label	Enable CanSM Distributed Channel Processing Support	
Description	This parameter enables CanSM channels Distribution. Effect: ► Enabled: CanSM Distributed Channel Processing is enabled. ► Disabled: CanSM Distributed Channel Processing is disabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanSMMaxNumberOfTransceivers**
Label	Max. number of transceivers accross all post-build variants.
Description	Since different variants can configure different number of transceivers, this parameter specifies the highest number of configured transceivers across all post-build variants.

Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.4.1.8. 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	2	
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	2
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	7
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	7
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	140
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	

5.4.1.9. PublishedInformation

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

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

5.4.2. Application programming interface (API)

5.4.2.1. Type definitions

5.4.2.1.1. CanSM_NetworkModeStateType

Purpose	Definition of the NetworkModeStateType.	
Type	enum	
Constants	CANSM_UNINITED	Network is uninitialized (initial state).
	CANSM_NO_COMMUNICATION	No communication. Wakeup can be detected.
	CANSM_SILENT_COMMUNICATION	No outgoing communication.

	CANSM_FULL_COMMUNICATION	All communication is possible.
Description	This type defines the states of the network mode state machine.	

5.4.2.2. Functions

5.4.2.2.1. CanSM_CheckTransceiverWakeFlagIndication

Purpose	This callback function indicates the CheckTransceiverWakeFlag API process end for the notified CAN Transceiver.	
Synopsis	`void` **`CanSM_CheckTransceiverWakeFlagIndication`** `( uint8 Transceiver );`	
Service ID	10	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CAN Transceivers	
Parameters (in)	`Transceiver`	Requested Transceiver

5.4.2.2.2. CanSM_ClearTrcvWufFlagIndication

Purpose	This callback function shall indicate the CanIf_ClearTrcvWufFlag API process end for the notified CAN Transceiver.	
Synopsis	`void` **`CanSM_ClearTrcvWufFlagIndication`** `( uint8 Transceiver );`	
Service ID	8	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CAN Transceivers	
Parameters (in)	`Transceiver`	Requested Transceiver

5.4.2.2.3. CanSM_ConfirmPnAvailability

Purpose	This callback function indicates that the transceiver is running in PN communication mode.
Synopsis	`void` **`CanSM_ConfirmPnAvailability`** `( uint8 TransceiverId );`
Service ID	6
Sync/Async	Synchronous

Reentrancy	Reentrant	
Parameters (in)	`TransceiverId`	CAN transceiver, which was checked for PN availability

5.4.2.2.4. CanSM_ControllerBusOff

Purpose	This function is called to notify a bus-off event.	
Synopsis	`void` **`CanSM_ControllerBusOff`** `( uint8 ControllerId );`	
Service ID	4	
Sync/Async	Synchronous	
Reentrancy	Reentrant (only for different CanControllers)	
Parameters (in)	`ControllerId`	CAN controller, which detected a bus-off event.
Description	This function notifies the CanSM module of a bus-off event on a CAN controller. The bus-off recovery state machine is executed for the corresponding network handle.	

5.4.2.2.5. CanSM_ControllerModeIndication

Purpose	This callback shall notify the CanSM module about a CAN controller mode change.	
Synopsis	`void` **`CanSM_ControllerModeIndication`** `( uint8 ControllerId , CanIf_ControllerModeType ControllerMode );`	
Service ID	7	
Sync/Async	Synchronous	
Reentrancy	Reentrant (only for different CAN controllers)	
Parameters (in)	`ControllerId`	CAN controller, whose mode has changed
	`ControllerMode`	Notified CAN controller mode

5.4.2.2.6. CanSM_GetCurrentComMode

Purpose	Provide the current communication mode of a CAN network.
Synopsis	`Std_ReturnType` **`CanSM_GetCurrentComMode`** `( NetworkHandleType network , ComM_ModeType * ComM_ModePtr );`
Service ID	3

Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`network`	Handle of the target network.
Parameters (out)	`ComM_ModePtr`	Pointer to where to store the current mode.
Return Value	Std_ReturnType	
	`E_OK`	No Error.
	`E_NOT_OK`	Getting of current Communication Mode failed.
Description	This service gets the current communication mode of a CAN network.	

5.4.2.2.7. CanSM_GetVersionInfo

Purpose	Get version information of the CanSM module.
Synopsis	`void` **`CanSM_GetVersionInfo`** `( Std_VersionInfoType * VersionInfo );`
Service ID	1
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (out)	`VersionInfo` — Pointer to where to store the version information of this module.
Description	This service puts out the version information of this module (module ID, vendor ID, vendor specific version numbers related to BSW00407).

5.4.2.2.8. CanSM_Init

Purpose	Initializes the CanSM module.
Synopsis	`void` **`CanSM_Init`** `( const CanSM_ConfigType * ConfigPtr );`
Service ID	0
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	`ConfigPtr` — Pointer to init structure for the post-build configuration parameters of the CanSM.

		This parameter is ignored because post-build configuration is not supported. Please use NULL_PTR as parameter for the initialization.
Description	This function initializes the CanSM module. It is the first function called in CanSM.	

5.4.2.2.9. CanSM_MainFunction

Purpose	This function handles asynchronous events, such as mode changes.
Synopsis	`void CanSM_MainFunction ( void );`
Service ID	5
Production Errors	▶ CANSM_E_BUS_OFF: thrown, if there is a notification of a bus-off event on a CAN controller
Description	This is the main function of the CanSM. It handles asynchronous events, such as mode changes. It is called cyclically with a fixed period from the BSW Scheduler.

5.4.2.2.10. CanSM_RequestComMode

Purpose	Request a change of the communication mode of a CAN network.	
Synopsis	`Std_ReturnType CanSM_RequestComMode ( NetworkHandleType network , ComM_ModeType ComM_Mode );`	
Service ID	2	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (only for different network handles)	
Parameters (in)	`network`	Handle of the target network.
	`ComM_Mode`	Requested communication mode.
Return Value	Std_ReturnType	
	`E_OK`	No Errors.
	`E_NOT_OK`	Requesting of Communication Mode failed.
Description	This service changes the communication mode of a CAN network to the requested one.	

5.4.2.2.11. CanSM_SetBaudrate

Purpose	Requests change of baudrate for indicated network.	
Synopsis	`Std_ReturnType` **`CanSM_SetBaudrate`** `( NetworkHandleType network , uint16 BaudRateConfigID );`	
Service ID	13	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant for same network	
Parameters (in)	`network`	Handle of destined communication network for request.
	`BaudRateConfigID`	References a baud rate configuration by ID (see CanControllerBaudRateConfigID).
Return Value	Std_ReturnType	
	`E_OK`	Service request accepted, setting of (new) baud rate.
	`E_NOT_OK`	Service request not accepted.
Description	This service shall start an asynchronous process to change the baud rate for the configured CAN controllers of a certain CAN network. Depending on necessary baud rate modifications the controllers might have to reset.	

5.4.2.2.12. CanSM_TransceiverModeIndication

Purpose	This callback shall notify the CanSM module about a CAN transceiver mode change.	
Synopsis	`void` **`CanSM_TransceiverModeIndication`** `( uint8 TransceiverId , CanTrcv_TrcvModeType TransceiverMode );`	
Service ID	9	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CAN Transceivers	
Parameters (in)	`TransceiverId`	CAN transceiver, whose mode has changed
	`TransceiverMode`	Notified CAN transceiver mode

5.4.2.2.13. CanSM_TxTimeoutException

Purpose	Request recovery from Tx timeout exception.
Synopsis	`void` **`CanSM_TxTimeoutException`** `( NetworkHandleType Channel );`

Service ID	11	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (only for different channels)	
Parameters (in)	`Channel`	Affected CAN network
Description	This function notifies the CanSM module, that the CanNm has detected a tx time-out exception for the affected partial CAN network, which shall be recovered by the CanSM module with a transition to no communication and back to the requested communication mode.	

5.4.3. Integration notes

5.4.3.1. Exclusive areas

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

5.4.3.1.1. SCHM_CANSM_EXCLUSIVE_AREA_0

Protected data structures	All shared data that shall be protected from mutual access.
Recommended locking mechanism	The locking mechanism for this exclusive area can be disabled if all of of the following conditions are true: ► ComM_MainFunction() does not interrupt CanSM_MainFunction() (and vice versa) ► Dcm_MainFunction() does not interrupt CanSM_MainFunction() (and vice versa) ► Can driver Mainfunction does not interrupt CanSM_MainFunction() (and vice versa) ► CanTrcv driver Mainfunction does not interrupt CanSM_MainFunction() (and vice versa) If the condition listed above does not apply, the exclusive area shall 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.

5.4.3.2. Production errors

CANSM_E_BUS_OFF	CanSM_MainFunction

5.4.3.3. 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
VAR_INIT_UNSPECIFIED
VAR_INIT_8
VAR_CLEARED_8
VAR_CLEARED_UNSPECIFIED
CONST_8
CODE
CONFIG_DATA_8
CONFIG_DATA_UNSPECIFIED

5.4.3.4. 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.

5.4.3.4.1. lim.CanSM.EB_INTREQ_CanSM_0001

Description	The CanSM state machine is incompatible to a CanTrcv strictly implemented according to the requirements of the CanTrcv SWS for the purpose of making a transition to state COMM_NO_COMMUNICATION if partial network is enabled.
Rationale	During a transition towards state COMM_NO_COMMUNICATION (refer to Figure 7-3) the CanSM calls CanIf_CheckTrcvWakeFlag [CANSM458] and waits for the callback function CanSM_CheckTransceiverWakeFlagIndication [CANSM460]. Contrary to the

	requirements in the CanSM SWS the CanTrcv SWS [CanTrcv224] demands the call-back only for the case that a wakeup was detected. To ensure correct functionality the CanTrcv must invoke CanIf_CheckTrcvWakeFlagIndication in any case.

5.4.3.4.2. lim.CanSM.EB_INTREQ_CanSM_0002

Description	The threshold that switches from Bus-Off Recovery L1 to L2 was implemented in such a way that the actual time was a 1 less than the configured parameter in CanSM configuration. After the implementation, the time spent waiting to swith to L2 is now equal to the configured parameter.
Rationale	The behaviour was that L2 was reached a MainFunction call earlier than expected. The fix was meant to bring the expectation of the user, configured in the CanSM configuration into actual sequence of events.

5.5. CanTp

5.5.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CanTpDefensiveProgramming	1..1	**Label:** Defensive Programming Options Parameters for defensive programming
CanTpGeneral	1..1	This container contains the general configuration parameters of the CanTp module.
CanTpJumpTable	1..1	This container contains the jumptable related configuration parameters of the CanTp module.
CanTpConfig	1..1	This container contains the configuration parameters and sub containers of the AUTOSAR CanTp module. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist 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.

Containers included		
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by Common-PublishedInformation 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

5.5.1.1. CanTpDefensiveProgramming

Parameters included	
Parameter name	**Multiplicity**
CanTpDefProgEnabled	1..1
CanTpPrecondAssertEnabled	1..1
CanTpPostcondAssertEnabled	1..1
CanTpStaticAssertEnabled	1..1
CanTpUnreachAssertEnabled	1..1
CanTpInvariantAssertEnabled	1..1

Parameter Name	CanTpDefProgEnabled
Label	Enable Defensive Programming
Description	Enables or disables the defensive programming feature for the module CanTp. 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	**CanTpPrecondAssertEnabled**
Label	Enable Precondition Assertions
Description	Enables handling of precondition assertion checks reported from the module CanTp. Dependency on parameter(s): ► Enable Development Error Detection (`CanTpDevErrorDetect`): must be enabled ► Enable Defensive Programming (`CanTpDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanTpPostcondAssertEnabled**
Label	Enable Postcondition Assertions
Description	Enables handling of postcondition assertion checks reported from the module CanTp. Dependency on parameter(s): ► Enable Development Error Detection (`CanTpDevErrorDetect`): must be enabled ► Enable Defensive Programming (`CanTpDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false

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

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

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

Parameter Name	**CanTpInvariantAssertEnabled**
Label	Enable Invariant Assertions

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

5.5.1.2. CanTpGeneral

Containers included		
Container name	**Multiplicity**	**Description**
CanTpChannelProcessing	0..n	Configuration of a dedicated MainFunction. The name of the generated function uses the pattern `CanTp_-MainFunction_-"Short-Name"`.

Parameters included	
Parameter name	**Multiplicity**
CanTpChangeParameterApi	1..1
CanTpDevErrorDetect	1..1
CanTpPaddingByte	1..1
CanTpReadParameterApi	1..1
CanTpVersionInfoApi	1..1
CanTpChangeTxParameterApi	1..1
CanTpPaddingByteCanFD	0..1
CanTpRelocatablePbcfgEnable	1..1
CanTpDynamicNSaEnabled	1..1
CanTpCancelReceiveApi	1..1

Parameters included	
CanTpCancelTransmitApi	1..1
CanTpGptUsageEnable	1..1
CanTpMaxParallelTxChannels	0..1
CanTpMaxParallelRxChannels	0..1
CanTpMaxTxChannels	1..1
CanTpMaxRxChannels	1..1
CanTpMaxRxNSdus	1..1
CanTpMaxTxNSdus	1..1
CanTpMaxFcPdus	1..1
CanTpDynIdSupport	1..1
CanTpFlexibleDataRateSupport	1..1
CanTpGenericConnectionSupport	1..1
CanTpNbrWaitRepeatedSupport	1..1
CanTpStallHandlingSupport	1..1
CanTpMultiCoreSupport	1..1
CanTpDedicatedChannelProcessingSupport	1..1

Parameter Name	CanTpChangeParameterApi	
Description	This parameter, if set to true, enables the CanTp_ChangeParameter and CanTp_ChangeRxParameter APIs for this Module. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **RAM reduction (config):** Disabling this parameter reduces the RAM consumption of the module configuration.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanTpDevErrorDetect
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	**CanTpPaddingByte**
Description	Used for the initialization of unused bytes with a certain value, for CAN 2.0 frames and CAN FD frames (if CANTP_PADDING_BYTE_CANFD is not configured).
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=255 >=0
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanTpReadParameterApi**
Description	This parameter, if set to true, enables the CanTp_ReadParameter API for this Module. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **RAM reduction (config):** Disabling this parameter reduces the RAM consumption of the module configuration.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpVersionInfoApi**	
Description	The function CanTp_GetVersionInfo is configurable (On/Off) by this configuration parameter. **Optimization Effect:** ► **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	AUTOSAR_ECUC	

Parameter Name	**CanTpChangeTxParameterApi**	
Description	This parameter, if set to true, enables the CanTp_ChangeTxParameter and CanTp_ResetTxParameter APIs for this Module. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **RAM reduction (config):** Disabling this parameter reduces the RAM consumption of the module configuration.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpPaddingByteCanFD**
Description	Used for the initialization of unused bytes with a certain value, for CAN FD frames.
Multiplicity	0..1
Type	INTEGER

Range	<=255 >=0	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpRelocatablePbcfgEnable**	
Description	Enables/disable support for relocatable postbuild configuration. ► True: Postbuild configuration relocatable in memory. ► False: Postbuild configuration not relocatable in memory. **Note:** If PbcfgM support is enabled for CanTp, this feature is managed by by the parameter PbcfgMRelocatableCfgEnable.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpDynamicNSaEnabled**
Description	Enables or disables the support for setting of the N_SA values for Rx and Tx N-SDUs during runtime. If the parameter is disabled, normal Autosar behaviour is activated, which means, that the N_SA parameter is defined during the configuration and cannot be changed. If it is enabled, the N_SA values of each Rx and Tx N-SDU can be changed via CanTp_SetNSa() during runtime. In this case, the current value can be read via CanTp_GetNSa(). **Optimization Effect:** ► **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	Elektrobit Automotive GmbH	

Parameter Name	**CanTpCancelReceiveApi**	
Description	Preprocessor switch for enabling the reception cancellation API function. **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	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpCancelTransmitApi**	
Description	Preprocessor switch for enabling the transmit cancellation API function. **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	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpGptUsageEnable**

Description	Preprocessor switch to enable the general purpose timer instead of the main function period for timeout handling of the channels. **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	**CanTpMaxParallelTxChannels**	
Description	Limits the total number of parallel transmit channels.	
Multiplicity	0..1	
Type	INTEGER	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpMaxParallelRxChannels**	
Description	Limits the total number of parallel receive channels.	
Multiplicity	0..1	
Type	INTEGER	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpMaxTxChannels**	
Description	Limits the total number of transmit channels.	
Multiplicity	1..1	
Type	INTEGER	
Default value	10	
Range	<=32767 >=1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Origin	Elektrobit Automotive GmbH

Parameter Name	**CanTpMaxRxChannels**	
Description	Limits the total number of receive channels.	
Multiplicity	1..1	
Type	INTEGER	
Default value	10	
Range	<=32767 >=1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpMaxRxNSdus**	
Description	Limits the total number of RX-PDUs.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=32767 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpMaxTxNSdus**	
Description	Limits the total number of TX-PDUs.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=32767 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpMaxFcPdus**
Description	Limits the total number of flow control PDUs.
Multiplicity	1..1
Type	INTEGER

Range	<=32767 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpDynIdSupport**	
Description	Enable/Disable support for dynamic ID handling via N-PDU MetaData. **Optimization Effect:** ► **RAM reduction (code):** Disabling this parameter reduces the RAM consumption of the module code.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpFlexibleDataRateSupport**	
Description	Enable/Disable support for CAN FD frames up to 64 byte. **Optimization Effect:** ► **RAM reduction (code):** Disabling this parameter reduces the RAM consumption of the module code.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild

Parameter Name	**CanTpGenericConnectionSupport**
Description	Enable/Disable support for handling of generic connection using N-SDUs with MetaData. **Optimization Effect:** ► **RAM reduction (code):** Disabling this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1

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

Parameter Name	**CanTpNbrWaitRepeatedSupport**	
Description	Enable/Disable support of different timeout values for repeated FC WAIT PDUs. **Optimization Effect:** ► **RAM reduction (code):** Disabling this parameter reduces the RAM consumption of the module code.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpStallHandlingSupport**	
Description	Enable/Disable support of stallhandlig feature for received messages. When this feature is enabled, if there is an ongoing reception and another RxIndication occurs before the ending of the first one, RxIndication will be postponed. **Optimization Effect:** ► **RAM reduction (code):** Disabling this parameter reduces the RAM consumption of the module code.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpMultiCoreSupport**
Description	Enable/Disable CanTp support for multicore.
Multiplicity	1..1
Type	BOOLEAN

Default value	false	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpDedicatedChannelProcessingSupport**	
Description	Enables support to map the processing of CanTp channels to different main functions. Also, it can be used besides the multicore feature.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.5.1.3. CanTpChannelProcessing

Parameters included	
Parameter name	**Multiplicity**
CanTpChannelRef	1..n
CanTpTimeBase	1..1
CanTpEcucPartitionRef	1..1

Parameter Name	**CanTpChannelRef**	
Description	Reference to CanTpChannel which is assigned to this MainFunction.	
Multiplicity	1..n	
Type	REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpTimeBase**
Description	Allow to configure the time for the MainFunction (as float in seconds). Please note: This period shall be the same as call cycle time of the periodic task were CanTp Main function is called.
Multiplicity	1..1

Type	FLOAT	
Range	<=3.6 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpEcucPartitionRef**	
Description	This container contains a refference to EcuCPartition which is referenced by a specific Core.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.5.1.4. CanTpJumpTable

Parameters included	
Parameter name	**Multiplicity**
CanTpJumpTableMode	1..1
CanTpUseSchMMacros	1..1
CanTpUseWrapperMacros	1..1
CanTpJumpTableAddress	1..1
CanTpJumpTableIncludeFile	0..1

Parameter Name	**CanTpJumpTableMode**
Description	Switches the jump table support ON/OFF and defines the jump table mode if enabled. ► OFF: Jump table support is off. ► SERVER: Jump table support is enabled and the module acts as jump table server (which means it provides all functionality). ► CLIENT: Jump table support is enabled and the module acts as jump table client (it provides the means to call CanTp server APIs). **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 reduction (code):** Setting this parameter to a value of SERVER for the bootloader and CLIENT for the application (or vice versa) reduces the ROM consumption of the module code compared to disabling this parameter for the application and the bootloader. ► **Execution time increase (code):** Enabling this parameter increases the execution time of the module code.
Multiplicity	1..1
Type	ENUMERATION
Default value	OFF
Range	OFF SERVER CLIENT
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanTpUseSchMMacros**
Description	Defines if the CanTp uses SchM macros or adds the SchM functions to its exit jumptable. ► True: The CanTp uses SchM macros. ► False: The CanTp adds SchM function pointers to the exit jumptable. This parameter is only used if jump table support is enabled. **Please make sure, that this parameter is configured equal for the jump table server and all associated clients.** **Optimization Effect:** ► **ROM reduction (config):** Enabling this parameter reduces the ROM consumption of the module configuration. ► **ROM reduction (code):** Enabling this parameter reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Enabling this parameter reduces 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	**CanTpUseWrapperMacros**
Description	Defines if the CanTp provides wrapper functions or macros for accessing the API functions via the entry jumptable. The names of those wrapper functions/macros are those of the original API functions. ► True: Wrapper macros are defined. ► False: Wrapper functions are defined. This parameter is only used if CanTpJumpTableMode is set to CLIENT. **Optimization Effect:** ► **ROM reduction (config):** Enabling this parameter reduces the ROM consumption of the module configuration. ► **ROM reduction (code):** Enabling this parameter reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Enabling this parameter reduces 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	**CanTpJumpTableAddress**
Description	This parameter defines the base address of the entry jumptable. The entry jumptable is the jumptable, via which an application can access the CanTp API functions. This parameter is only used if CanTpJumpTableMode is set to CLIENT. **Please note, that the value of this parameter is directly written to the C code and must therefore be a correct symbol or address for being compilable.**

Multiplicity	1..1	
Type	STRING	
Default value	0x0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpJumpTableIncludeFile**	
Description	This parameter defines the name of the include file that shall be included, if a symbol is used for parameter CanTpJumpTableAddress. **Please enable this parameter only if CanTpJumpTableAddress is set to a symbol. The named include file must then provide the symbol declaration.**	
Multiplicity	0..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.5.1.5. CanTpConfig

Containers included		
Container name	**Multiplicity**	**Description**
CanTpChannel	1..n	This container contains the configuration parameters of the CanTp channel.

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

Parameter Name	**CanTpMainFunctionPeriod**
Description	Allow to configure the time for the MainFunction (as float in seconds). Please note: This period shall be the same as call cycle time of the periodic task were CanTp Main function is called.
Multiplicity	1..1
Type	FLOAT

Default value	0.005	
Range	<=0.255 >=0.0001	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.5.1.6. CanTpChannel

Containers included		
Container name	**Multiplicity**	**Description**
CanTpRxNSdu	0..n	The following parameters need to be configured for each CAN N-SDU that the CanTp module receives via the CanTpChannel.
CanTpTxNSdu	0..n	The following parameters need to be configured for each CAN N-SDU that the CanTp module transmits via the CanTpChannel.

Parameters included	
Parameter name	**Multiplicity**
CanTpChannelMode	1..1
CanTpSTminTimeoutHandling	1..1
CanTpGptChannelId	1..1
CanTpGptChannelResolution	1..1
CanTpGptChannelCallbackName	1..1

Parameter Name	**CanTpChannelMode**
Description	The CAN Transport Layer supports half and full duplex channel modes. **Optimization Effect:** ► **ROM reduction (config):** Choosing the value CANTP_MODE_HALF_DUPLEX for this parameter reduces the ROM consumption of the module configuration. ► **RAM reduction (config):** Choosing the value CANTP_MODE_HALF_DUPLEX for this parameter reduces the RAM consumption of the module.
Multiplicity	1..1

Type	ENUMERATION	
Default value	CANTP_MODE_HALF_DUPLEX	
Range	CANTP_MODE_FULL_DUPLEX CANTP_MODE_HALF_DUPLEX	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpSTminTimeoutHandling**
Description	Decides whether waiting for STmin timing is done via the GPT or from the CanTp_MainFunction. **Optimization Effect:** ► **ROM increase (config):** Setting this parameter to Gpt increases the ROM consumption of the module configuration. ► **ROM increase (code):** Setting this parameter to Gpt increases the ROM consumption of the module code. ► **Execution time increase (code):** Setting this parameter to Gpt increases the execution time of the module code.
Multiplicity	1..1
Type	ENUMERATION
Default value	CanTpMainFunction
Range	Gpt CanTpMainFunction
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanTpGptChannelId**
Description	Identifier/Channel Handle of the GPT timer to be used for this channel.
Multiplicity	1..1
Type	STRING
Configuration class	**PostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**CanTpGptChannelResolution**

Description	The resolution of the associated GPT channel (in ns per tick).	
Multiplicity	1..1	
Type	INTEGER	
Range	<=1000000000 >0	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**CanTpGptChannelCallbackName**	
Description	Identifier of the GPT callback routine for this channel. An appropriate callback function with this name will be created by the CanTp.	
Multiplicity	1..1	
Type	STRING	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.5.1.7. CanTpRxNSdu

Containers included		
Container name	**Multiplicity**	**Description**
CanTpNAe	1..1	Contains the parameters needed to configure each Rx N-SDU with CanTpRxAddressingFormat set to CANTP_-MIXED.
CanTpNSa	1..1	Contains the parameters needed to configure each Rx N-SDU with: - CanTpRxAddressingFormat set to CANTP_EX-TENDED or - CanTpRxAddressingFormat set to CANTP_-NORMALFIXED and CanTpDynIdSupport enabled and CanTpGenericConnectionSupport disabled.
CanTpNTa	1..1	The following parameters need to be configured for each RxNsdu with: - the CanTpRxAddressingFormat set to CANTP_EXTENDED. - the CanTpRxAddressingFormat set to CANTP_NORMALFIXED and CanTpDynIdSupport en-abled.
CanTpRxNPdu	1..1	Used for grouping of the ID of a PDU and the Reference to a PDU.

Containers included		
CanTpTxFcNPdu	1..1	Used for grouping of the ID of a PDU and the reference to a PDU.

Parameters included	
Parameter name	**Multiplicity**
CanTpBs	1..1
CanTpNar	1..1
CanTpNbr	1..1
CanTpNbrWaitRepeated	0..1
CanTpNcr	1..1
CanTpRxAddressingFormat	1..1
CanTpRxDl	1..1
CanTpRxNSduId	1..1
CanTpRxPaddingActivation	0..1
CanTpRxTaType	1..1
CanTpRxWftMax	1..1
CanTpSTmin	1..1
CanTpRxNSduRef	1..1

Parameter Name	CanTpBs	
Description	Sets the maximum number of N-PDUs the CanTp receiver allows the sender to send, before waiting for an authorization to continue transmission of the following N-PDUs. For further details on this parameter value see ISO 15765-2 specification. **Note:** For reasons of buffer length, the CAN Transport Layer can adapt the BS value within the limit of this maximum BS.	
Multiplicity	1..1	
Type	INTEGER	
Default value	16	
Range	<=255 >=0	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanTpNar	
Description	Value in seconds of the N_Ar timeout. N_Ar is the time for transmission of a CAN frame (any N_PDU) on the receiver side.	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.1	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanTpNbr	
Description	Value in seconds of the performance requirement for (N_Br + N_Ar). N_Br is the elapsed time between the receiving indication of a FF or CF or the transmit confirmation of a FC, until the transmit request of the next FC. **Note:** N_Br is only respected when sending FC(WT) frames, other flow control status (CTS, OVFLW) are sent at once when their conditions are met.	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanTpNbrWaitRepeated	
Description	Vendor specific configuration parameter for handling of dedicated N_Br time-out values for repeated FC WAIT PDUs. Value in seconds of the performance requirement for(N_Br_WT + N_Ar). N_Br_WT is the elapsed time until the next Flow Control transmission when sending subsequent FC WAIT PDUs. **Note:** N_Br_WT is only respected when sending FC(WT) frames, starting with the second FC(WT).	
Multiplicity	0..1	
Type	FLOAT	
Default value	0.0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanTpNcr

Description	Value in seconds of the N_Cr timeout. N_Cr is the time until reception of the next Consecutive Frame N_PDU is expected.	
Multiplicity	1..1	
Type	FLOAT	
Default value	1.0	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpRxAddressingFormat**
Description	Declares which communication addressing format is supported for this Rx N-SDU. Enum values: ► CANTP_STANDARD: Use normal addressing format. ► CANTP_EXTENDED: Use extended addressing format. ► CANTP_NORMALFIXED: Use normal fixed addressing format. ► CANTP_MIXED: Use mixed addressing format.
Multiplicity	1..1
Type	ENUMERATION
Default value	CANTP_STANDARD
Range	CANTP_EXTENDED CANTP_MIXED CANTP_NORMALFIXED CANTP_STANDARD
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**CanTpRxDl**
Description	**This parameter has been deprecate according to AUTOSAR RFC53101.** Data Length Code of this RxNsdu. In case of variable message length, this value indicates the minimum data length. Depending on SF or FF N-SDU the value will be limited to 7 (6 for an extended addressing format) and 4095 respectively.
Multiplicity	1..1

Type	INTEGER	
Default value	1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpRxNSduId**	
Description	Unique identifier used by the upper layer to call CanTp_CancelReceive, CanTp_ChangeParameter and CanTp_ReadParameter.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpRxPaddingActivation**	
Description	Defines if the receive frame uses padding or not. Definition of enumeration values: ► CANTP_OFF: The transmit N-PDU does not use padding for SF, CF and the last CF. (N-PDU length is dynamic) ► CANTP_ON: Enabled mandatory padding to 8 bytes for CAN 2.0 PDUs only (SF, FC and last CF). ► CANTP_ON_CAN_CAN_FD: Enable mandatory padding to 8 bytes for CAN 2.0 PDUs and 64 bytes for CAN FD PDUs	
Multiplicity	0..1	
Type	ENUMERATION	
Default value	CANTP_ON	
Range	CANTP_OFF CANTP_ON CANTP_ON_CAN_CAN_FD	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpRxTaType**
Description	Declares the communication type of this Rx N-SDU.
Multiplicity	1..1

Type	ENUMERATION	
Default value	CANTP_PHYSICAL	
Range	CANTP_FUNCTIONAL CANTP_PHYSICAL	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpRxWftMax**	
Description	This parameter indicates how many Flow Control wait N-PDUs can be consecutively transmitted by the receiver. It is local to the node and is not transmitted inside the FC protocol data unit. CanTpRxWftMax is used to avoid sender nodes being potentially hooked-up in case of a temporarily reception inability on the part of the receiver nodes, whereby the sender could be waiting continuously.	
Multiplicity	1..1	
Type	INTEGER	
Default value	4	
Range	<=255 >=0	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpSTmin**	
Description	Sets the duration of the minimum time (in seconds) the CanTp sender shall wait between the transmissions of two CF N-PDUs. For further details on this parameter value see ISO 15765-2 specification.	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.0	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpRxNSduRef**
Description	Reference to a PDU in the COM-Stack.

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

5.5.1.8. CanTpNAe

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

Parameter Name	**CanTpNAe**	
Description	If an Rx N-SDU is configured for mixed addressing format, this parameter contains the value of the transport protocol address extension.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.5.1.9. CanTpNSa

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

Parameter Name	**CanTpNSa**	
Description	If an Rx N-SDU is configured for extended addressing format or normal fixed addressing format (CanTpDynIdSupport enabled and CanTpGenericConnectionSupport disabled), this parameter contains the value of the transport protocol address of the local node (e.g., this ECU).	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.5.1.10. CanTpNTa

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

Parameter Name	CanTpNTa	
Description	If a Rx N-SDU is configured for extended addressing format or normal fixed addressing format (CanTpDynIdSupport enabled), this parameter contains the value of the transport protocol address of the remote node (e.g., the diagnostic tester), if an Rx N-SDU is configured with: - extended addressing format - Normal fixed addressing format and CanTpDynIdSupport is enabled. .	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.5.1.11. CanTpRxNPdu

Parameters included	
Parameter name	**Multiplicity**
CanTpRxNPduId	1..1
CanTpRxNPduRef	1..1

Parameter Name	CanTpRxNPduId	
Description	The N-PDU identifier attached to the RxNsdu is identified by CanTpRxNSduId. Each RxNsdu identifier is linked to only one SF/FF/CF N-PDU identifier. Nevertheless, in the case of extended or mixed addressing format, the same N-PDU identifier can be used for several N-SDU identifiers. The distinction is made by the N_TA or N_AE value (first data byte of SF or FF).	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanTpRxNPduRef

Description	Reference to a PDU in the COM-Stack.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.5.1.12. CanTpTxFcNPdu

Parameters included	
Parameter name	**Multiplicity**
CanTpTxFcNPduConfirmationPduId	1..1
CanTpTxFcNPduRef	1..1

Parameter Name	**CanTpTxFcNPduConfirmationPduId**	
Description	Handle ID to be used by the CanIf to confirm the transmission of the CanTp-TxFcNPdu to the CanIf module.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpTxFcNPduRef**	
Description	Reference to a PDU in the COM-Stack.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.5.1.13. CanTpTxNSdu

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

Containers included		
CanTpNAe	1..1	Contains the parameters needed to configure each Tx N-SDU with CanTpTxAddressingFormat set to CANTP_-MIXED.
CanTpNSa	1..1	Contains the parameters needed to configure each Tx N-SDU with: - CanTpTxAddressingFormat set to CANTP_EX-TENDED. - CanTpTxAddressingFormat set to CANTP_NOR-MALFIXED and CanTpDynIdSupport enabled.
CanTpNTa	1..1	The following parameters need to be configured for each Tx N-SDU with: - CanTpTxAddressingFormat set to CANTP_-EXTENDED. - CanTpTxAddressingFormat set to CANTP_-NORMALFIXED, CanTpDynIdSupport enabled and CanTp-GenericConnectionSupport disabled.
CanTpRxFcNPdu	1..1	Used for grouping of the ID of a PDU and the Reference to a PDU.
CanTpTxNPdu	1..1	Used for grouping of the ID of a PDU and the Reference to a PDU.

Parameters included	
Parameter name	**Multiplicity**
CanTpNas	1..1
CanTpNbs	1..1
CanTpNcs	1..1
CanTpTc	1..1
CanTpTxAddressingFormat	1..1
CanTpTxDl	1..1
CanTpTxNSduId	1..1
CanTpTxPaddingActivation	0..1
CanTpTxTaType	1..1
CanTpTxNSduRef	1..1

Parameter Name	CanTpNas
Description	Value in second of the N_As timeout. N_As is the time for the transmission of a CAN frame (any N-PDU) on the part of the sender.
Multiplicity	1..1
Type	FLOAT

Default value	0.1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpNbs**	
Description	Value in seconds of the N_Bs timeout. N_Bs is the time of transmission until reception of the next Flow Control N-PDU.	
Multiplicity	1..1	
Type	FLOAT	
Default value	1.0	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpNcs**	
Description	Value in seconds of the performance requirement of (N_Cs + N_As). N_Cs is the time which elapses between the transmit request of a CF N-PDU until the transmit request of the next CF N-PDU. **Note:** N_Cs is used as buffer request timeout by Autosar. Therefore it should be greater than the minimum separation time (STmin) of connections allowing segmented frames. Otherwise, the transmission of following consecutive frames is prevented by this timeout if the connections block size allows the transmission of several consecutive frames between two flow control frames.	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.9	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpTc**	
Description	Switch for enabling Transmit Cancellation for a certain Tx N-Sdu.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanTpTxAddressingFormat	
Description	Declares which communication addressing format is supported for this Tx N-SDU. Definition of Enumeration values: ► CANTP_STANDARD: Use normal addressing format. ► CANTP_EXTENDED: Use extended addressing format (the N_TA container of this TxNsdu will be used). ► CANTP_NORMALFIXED: Use normal fixed addressing format. ► CANTP_MIXED: Use mixed addressing format (the N_AE container of this TxNsdu will be used).	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	CANTP_STANDARD	
Range	CANTP_EXTENDED CANTP_MIXED CANTP_NORMALFIXED CANTP_STANDARD	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanTpTxDl	
Description	**This parameter has been deprecate according to AUTOSAR RFC53101.** Data Length Code of this Tx N-SDU. In case of variable length message, this value indicates the minimum data length.	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanTpTxNSduId
Description	Unique identifier to a structure that contains all useful information to process the transmission of a Tx N-SDU.

Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpTxPaddingActivation**	
Description	Defines if the transmit frame use padding or not. Definition of Enumeration values: ► CANTP_OFF: ► No padding needed for CAN 2.0 PDUs ► Enable mandatory padding with variable lengths (8, 12, 16, 20, 24, 32, 48, 64 based on configured EcuC maximum length) for CAN FD PDUs ► CANTP_ON: ► Enabled mandatory padding to 8 bytes for CAN 2.0 PDUs (SF, FC and last CF). ► Enable mandatory padding with variable lengths (8, 12, 16, 20, 24, 32, 48, 64 based on configured EcuC maximum length) for CAN FD PDUs ► CANTP_ON_CAN_CAN_FD: Enable mandatory padding to 8 bytes for CAN 2.0 PDUs and 64 bytes for CAN FD PDUs	
Multiplicity	0..1	
Type	ENUMERATION	
Default value	CANTP_ON	
Range	CANTP_OFF CANTP_ON CANTP_ON_CAN_CAN_FD	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpTxTaType**
Description	Declares the communication type of this Tx N-SDU. Enumeration values: ► CANTP_PHYSICAL: Used for 1:1 communication. ► CANTP_FUNCTIONAL: Used for 1:n communication (SFs only).

Multiplicity	1..1	
Type	ENUMERATION	
Default value	CANTP_PHYSICAL	
Range	CANTP_FUNCTIONAL CANTP_PHYSICAL	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**CanTpTxNSduRef**	
Description	Reference to a PDU in the COM-Stack.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.5.1.14. CanTpNAe

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

Parameter Name	**CanTpNAe**	
Description	If a Tx N-SDU is configured for mixed addressing format, this parameter contains the value of the transport protocol address extension.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.5.1.15. CanTpNSa

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
CanTpNSa	1..1

Parameter Name	CanTpNSa	
Description	This parameter contains the value of the transport protocol address of the local node (i.e., this ECU), if a Tx N-SDU is configured with: - extended addressing format - normal fixed addressing format and CanTpDynIdSupport is enabled. .	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.5.1.16. CanTpNTa

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

Parameter Name	CanTpNTa	
Description	This parameter contains the value of the transport protocol address of the remote node (e.g., the diagnostic tester), if Tx N-SDU is configured with: - extended addressing format - normal fixed addressing format, CanTpDynIdSupport is enabled and CanTpGenericConnectionSupport is disabled.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.5.1.17. CanTpRxFcNPdu

Parameters included	
Parameter name	**Multiplicity**
CanTpRxFcNPduId	1..1
CanTpRxFcNPduRef	1..1

Parameter Name	CanTpRxFcNPduId	
Description	N-PDU identifier attached to the FC N-PDU of this Tx N-SDU identified by CanTpTxNSduId. Each Tx N-SDU identifier is linked to one Rx FC N-PDU identifier only. However, in the case of extended or mixed addressing format, the same FC N-PDU identifier can be used for several N-SDU identifiers. The distinction is made by means of the N_TA value (extended addressing format) or the N_AE value (mixed addressing format). In both cases, this is the first data byte of FC frame.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanTpRxFcNPduRef	
Description	Reference to a PDU in the COM-Stack.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.5.1.18. CanTpTxNPdu

Parameters included	
Parameter name	**Multiplicity**
CanTpTxNPduConfirmationPduId	1..1
CanTpTxNPduRef	1..1

Parameter Name	CanTpTxNPduConfirmationPduId	
Description	Handle ID to be used by the CanIf to confirm the transmission of the CanTpTxNPdu to the CanIf module.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	CanTpTxNPduRef	
Description	Reference to a PDU in the COM-Stack.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.5.1.19. 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	6
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	45
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	35
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

5.5.1.20. PublishedInformation

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

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

5.5.2. Application programming interface (API)

5.5.2.1. Macro constants

5.5.2.1.1. CANTP_PDU_DIR_RECEIVE

Purpose	Define for CanTp N-SDU direction (Rx N_SDU).

Value	1U

5.5.2.1.2. CANTP_PDU_DIR_TRANSMIT

Purpose	Define for CanTp N-SDU direction (Tx N_SDU).
Value	0U

5.5.2.2. Functions

5.5.2.2.1. CanTp_CancelReceive

Purpose	Cancel an ongoing reception.	
Synopsis	`Std_ReturnType` **`CanTp_CancelReceive`** `( PduIdType CanTpRxSduId );`	
Service ID	0x09	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`CanTpRxSduId`	Identifier of the received N-SDU.
Return Value	Std_ReturnType	
	`E_OK`	Cancellation request of the specified N-SDU is accepted.
	`E_NOT_OK`	Cancellation request is rejected; the reason can be that request is issued for an N-SDU that is not segmented or request is issued for an N-SDU that is not in the reception process.
Description	This service is used to cancel an ongoing segmented reception. ** Preconditions: ► Related N-SDU is segmented and not waiting for the last CF. ► N-SDU id must be valid. ► Module must be initialized. ► Corresponding channel is not waiting for a response of the lower layer.	

5.5.2.2.2. CanTp_CancelTransmit

Purpose	Cancel a pending transfer.	
Synopsis	`Std_ReturnType` **`CanTp_CancelTransmit`** `( PduIdType CanTpTxSduId );`	
Service ID	0x08	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`CanTpTxSduId`	Tx N-SDU ID
Return Value	Std_ReturnType	
	`E_OK`	cancellation request accepted
	`E_NOT_OK`	cancellation request rejected
Description	This API-Service is used to cancel the transfer of pending Can N-SDUs.	

5.5.2.2.3. CanTp_ChangeParameter

Purpose	Change parameter BS or STmin.	
Synopsis	`Std_ReturnType` **`CanTp_ChangeParameter`** `( PduIdType Id , TPParameterType Parameter , uint16 Value );`	
Service ID	0x0A	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`Id`	Identifier of the received N-SDU on which the parameter has to be changed.
	`Parameter`	Specify the parameter to which the value has to be changed (BS or STmin).
	`Value`	The new value of the parameter.
Return Value	Std_ReturnType	
	`E_OK`	request is accepted.
	`E_NOT_OK`	request is not accepted.
Description	This API-Service is used to request the change of reception parameters BS and STmin for a specified N-SDU. Preconditions: ► Related N-SDU must not be in the process of reception	

	▶ Function parameter must be valid

5.5.2.2.4. CanTp_ChangeRxParameter

Purpose	Change parameter BS or STmin.	
Synopsis	`Std_ReturnType` **`CanTp_ChangeRxParameter`** `( PduIdType Id , TPParameterType Parameter , uint16 Value );`	
Service ID	0x0A	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`Id`	Identifier of the received N-SDU on which the parameter has to be changed.
	`Parameter`	Specify the parameter to which the value has to be changed (BS or STmin).
	`Value`	The new value of the parameter.
Return Value	Std_ReturnType	
	`E_OK`	request is accepted.
	`E_NOT_OK`	request is not accepted.
Description	This API-Service is used to request the change of reception parameters BS and STmin for a specified N-SDU. Preconditions: ▶ Related N-SDU must not be in the process of reception ▶ Function parameter must be valid	

5.5.2.2.5. CanTp_ChangeTxParameter

Purpose	Change parameter STmin.
Synopsis	`Std_ReturnType` **`CanTp_ChangeTxParameter`** `( PduIdType Id , TPParameterType Parameter , uint16 Value );`
Service ID	0x0C
Sync/Async	Synchronous
Reentrancy	Reentrant

Parameters (in)	`Id`	Identifier of the transmitted N-SDU on which the parameter has to be changed.
	`Parameter`	Specify the parameter to which the value has to be changed (STmin).
	`Value`	The new value of the parameter.
Return Value	Std_ReturnType	
	`E_OK`	request is accepted.
	`E_NOT_OK`	request is not accepted.
Description	This API-Service is used to request the change of transmission parameter STmin for a specified N-SDU. Preconditions: ► Function parameter must be valid	

5.5.2.2.6. CanTp_ChannelHandling

Purpose	CanTp_ChannelHandling().
Synopsis	`void` **`CanTp_ChannelHandling`** `( CanTp_ChType Channel );`
Description	This function is to handle each channel Stall,Timeout and State by calling the following functions: CanTp_StallHandling(), CanTp_TimeoutHandling() , CanTp_RxStateHandling(), CanTp_TxStateHandling()

5.5.2.2.7. CanTp_GetNSa

Purpose	Get N_SA value for a specific N-SDU ID.	
Synopsis	`Std_ReturnType` **`CanTp_GetNSa`** `( PduIdType CanTpPduId , uint8 CanTpDirection , uint8 * CanTpNSaPtr );`	
Service ID	0x1F	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`CanTpPduId`	Contains the N-SDU ID
	`CanTpDirection`	CANTP_PDU_DIR_TRANSMIT and CANTP_PDU_DIR_RECEIVE
Parameters (out)	`CanTpNSaPtr`	A pointer to a N_SA value, where the current N_SA value can be written to.

Return Value	Std_ReturnType	
	`E_OK`	No Errors.
	`E_NOT_OK`	Det error occured or N-SDU is not configured.
Description	This service gets the N_SA value for the given N-SDU if it is configured with extended addressing format. N_SA in this context always refers to the address of this (the own) ECU, independent of the N-SDU direction. For CanTpDirection CANTP_PDU_DIR_RECEIVE it can be found in CanTp configuration in the configuration parameter CanTpRxNSdu/CanTpRxNSduId. For CanTpDirection CANTP_PDU_DIR_TRANSMIT it can be found in CanTp configuration in the configuration parameter CanTpTxNSdu/CanTpTxNSduId.	

5.5.2.2.8. CanTp_GetVersionInfo

Purpose	Get version information of the CanTP module.	
Synopsis	`void` **`CanTp_GetVersionInfo`** `( Std_VersionInfoType * VersionInfoPtr );`	
Service ID	0x07	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (out)	`VersionInfoPtr`	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: ► Module Id ► Vendor Id ► Vendor specific version numbers	

5.5.2.2.9. CanTp_Init

Purpose	Initialize the CanTP module.
Synopsis	`void` **`CanTp_Init`** `( const CanTp_ConfigType * CfgPtr );`
Service ID	0x01

Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`CfgPtr`	Pointer to the CanTp post-build configuration data. This parameter is ignored because the CanTp does not support post-build configuration. Please use NULL_PTR as parameter for the initialization.
Description	This function initializes the CanTP module.	

5.5.2.2.10. CanTp_IsValidConfig

Purpose	Validate configuration.
Synopsis	`Std_ReturnType` **`CanTp_IsValidConfig`** `( const void * voidConfigPtr );`
Service ID	0x60
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.

5.5.2.2.11. CanTp_MainFunction

Purpose	Main function of the CanTp.
Synopsis	`void` **`CanTp_MainFunction`** `( void );`
Service ID	0x06
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This function is the main function for scheduling CanTP.

5.5.2.2.12. CanTp_ReadParameter

Purpose	Read parameter BS or STmin.

Synopsis	`Std_ReturnType` **`CanTp_ReadParameter`** `( PduIdType id , TPParameterType parameter , uint16 * value );`	
Service ID	0x0B	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	`id`	Identifier of the received N-SDU on which the reception parameters are read.
	`parameter`	Specify the parameter to which the value has to be read (BS or STmin).
	`value`	Pointer where the parameter value will be provided.
Return Value	Std_ReturnType	
	`E_OK`	request is accepted.
	`E_NOT_OK`	request is not accepted.
Description	This API-Service is used to read reception parameters BS and STmin for a specified N-SDU. Preconditions: ► Function parameter must be valid	

5.5.2.2.13. CanTp_ResetTxParameter

Purpose	Reset parameter STmin.	
Synopsis	`void` **`CanTp_ResetTxParameter`** `( PduIdType Id );`	
Service ID	0x0D	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`Id`	Identifier of the transmitted N-SDU on which the parameter has to be reset.
Description	This API-Service is used to request the reset of transmission parameter STmin for a specified N-SDU. Preconditions: ► Function parameter must be valid	

5.5.2.2.14. CanTp_RxIndication

Purpose	Indicate a successful reception.	
Synopsis	`void CanTp_RxIndication ( PduIdType CanTpRxPduId , PduInfoType * CanTpRxPduPtr );`	
Service ID	0x42	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`CanTpRxPduId`	ID of CAN L-PDU that has been received. Identifies the data that has been received. Range: 0..(maximum number of L-PDU IDs received) - 1.
	`CanTpRxPduPtr`	Indicator of structure with received L-SDU (payload) and data length.
Description	This function is called by the CAN Interface after a successful reception of a Rx CAN L-PDU.	

5.5.2.2.15. CanTp_SetNSa

Purpose	Set N_SA value for a specific N-SDU ID.	
Synopsis	`Std_ReturnType CanTp_SetNSa ( PduIdType CanTpPduId , uint8 CanTpDirection , uint8 CanTpNSa );`	
Service ID	0x1E	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`CanTpPduId`	Contains the N-SDU ID
	`CanTpDirection`	CANTP_PDU_DIR_TRANSMIT and CANTP_PDU_DIR_RECEIVE
	`CanTpNSa`	N_SA value to be set
Return Value	Std_ReturnType	
	`E_OK`	No Errors.
	`E_NOT_OK`	Det error occured or N-SDU is not configured.
Description	This service sets the N_SA value for the given N-SDU if it is configured with extended addressing format. N_SA in this context always refers to the address of this (the own) ECU, independent of the N-SDU direction.	

	For CanTpDirection CANTP_PDU_DIR_RECEIVE it can be found in CanTp configuration in the configuration parameter CanTpRxNSdu/CanTpRxNSduId. For CanTpDirection CANTP_PDU_DIR_TRANSMIT it can be found in CanTp configuration in the configuration parameter CanTpTxNSdu/CanTpTxNSduId.

5.5.2.2.16. CanTp_Transmit

Purpose	Transfer segmented data.	
Synopsis	`Std_ReturnType` **`CanTp_Transmit`** `( PduIdType CanTpTxSduId , const PduInfoType * CanTpTxInfoPtr );`	
Service ID	0x03	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`CanTpTxSduId`	Contains the unique CanTp module identifier of the CAN N-SDU to be transmitted. Range: - 0..(maximum number of L-PDU IDs received) - 1.
	`CanTpTxInfoPtr`	A pointer to a structure with CAN N-SDU related data (CAN N-SDU buffer and the length of this buffer)
Return Value	Std_ReturnType	
	`E_OK`	No Errors.
	`E_NOT_OK`	The request cannot be started (e.g. a transmit request is in progress with the same N-SDU identifier).
Description	This service is used to request the transfer of segmented data.	

5.5.2.2.17. CanTp_TxConfirmation

Purpose	Confirm transmitted frame.
Synopsis	`void` **`CanTp_TxConfirmation`** `( PduIdType CanTpTxPduId );`
Service ID	0x40
Sync/Async	Synchronous
Reentrancy	Reentrant

Parameters (in)	`CanTpTxPduId`	ID of CAN L-PDU that has been transmitted. Range: 0...(maximum number of L-PDU IDs received) - 1.
Description	This function confirms all transmitted CAN frames belonging to the CAN Transport Layer.	

5.5.3. Integration notes

5.5.3.1. Exclusive areas

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

5.5.3.1.1. SCHM_CANTP_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.

5.5.3.2. Production errors

Production errors are not reported by the CanTp module.

5.5.3.3. 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
CONST_ENTRY_JUMP_TABLE
CONST_UNSPECIFIED
CONST_EXIT_JUMP_TABLE
VAR_INIT_JUMP_TABLE_SHARED_UNSPECIFIED
VAR_CLEARED_8
VAR_CLEARED_16
VAR_CLEARED_UNSPECIFIED
CONFIG_DATA_UNSPECIFIED
VAR_INIT_UNSPECIFIED
CODE
CONST_32
CODE_CC_BLOCK

5.5.3.4. 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.

5.5.3.4.1. lim.CanTp.EB_INTREQ_CanTp_0001

Description	Limitation on multiple invocations of functions The module postpones invocations of CanTp_RxIndication, CanTp_TxConfirmation or the Gpt callback function in case that the channel is occupied to handle a previous call of these functions. In this case the incoming call is postponed. In case of multiple calls on an occupied channel only the last call is stored and all previous pending calls are discarded.
Rationale	API functions have to occupy the CanTp Channel for consistent operations. During execution the API function might get interrupted. Especially the three functions CanTp_RxIndication, CanTp_TxConfirmation and the Gpt callback function are relevant because they are most likely called in interrupt mode. The CanTp addresses this issue. One incoming call of CanTp_RxIndication, CanTp_TxConfirmation and the Gpt callback function per channel are stored and resolved at the end of the initial API function call. In the unlikely event of multiple calls take place while the channel is oc-

	cupied. The last incoming frame is stored. Discarded frames segmented messages are detected through the sequence number check provided for this type of frames.

5.5.3.4.2. lim.CanTp.EB_INTREQ_CanTp_0002

Description	Limitation on API CanTp_CancelReceive and CanTp_CancelTransmit The API services CanTp_CancelReceive and CanTp_CancelTransmit do not cancel an ongoing reception/transmission of a message if the API call interrupts data processing. In this case the APIs signalize the disability to cancel the communication by returning E_NOT_OK. For a successful receive cancellation following preconditions must be fulfilled: - Related N-SDU is in state of reception. - Receive cancellation is applied to a segmented message. - Channel is not locked. - CanTp is not waiting for the last consecutive frame. Channel is not waiting for a TX confirmation response from lower layer. For a successful transmit cancellation following preconditions must be fulfilled: - Related N-SDU is in state of transmission. - Channel is not locked. Channel is not waiting for a TX confirmation response from lower layer. Channel is not waiting for a flow control message.
Rationale	To ensure internal data consistency of a communication channel it is advisable to wait with the cancellation until the data handling mechanism frees (unlocks) the channel. A storage of the cancellation event to process it after the channel unlock is not possible because the API service shall return immediately with the correct return status which is not yet known.

5.5.3.4.3. lim.CanTp.EB_INTREQ_CanTp_0003

Description	The effect of CanTpGeneral/CanTpMainFunctionPeriod on the accuracy of the callback functions If CanTpSTminTimeoutHandling is configured to CanTpMainFunction, the module uses an internal counter to trigger the callback routine used for the STmin delay. Note that the accuracy of this method depends highly on the time between subsequent CanTp_MainFunction calls as specified in CanTpGeneral/CanTpMainFunc-

	tionPeriod. This value is also used to calculate the counter values for the CanTp_-MainFunction timer.
Rationale	

5.5.3.4.4. lim.CanTp.EB_INTREQ_CanTp_0004

Description	The reinitialization process must not interrupt other module functions If reinitialization of the module is required, the call of CanTp_Init must not interrupt other module functions.
Rationale	The reinitialization process resets all internal variables. Continuing and interrupted module function after reinitialization can lead to undefined module behavior.

5.5.3.4.5. lim.CanTp.EB_INTREQ_CanTp_0005

Description	CanTp_Init() shall not be preempted by any other module API calls. It needs to be ensured that the function call CanTp_Init() is not preempted by any other module API calls.
Rationale	During the call of CanTp_Init() global variables and pointers get initialized. It is easy for the integrator to avoid this preemption, thus no data protection mechanism has been implemented for function CanTp_Init().

5.5.3.4.6. lim.CanTp.EB_INTREQ_CanTp_0006

Description	Functions Gpt_StopTimer() and Gpt_StartTimer() must be concurrently callable from different partitions/cores for different Gpt channels if Gpt (CanTpGptUsageEnable) is used with the multicore channel distribution feature (CanTpMultiCoreSupport).
Rationale	

6. Bibliography

Bibliography

[1] *AUTOSAR consortium homepage*, URL: http://www.autosar.org/, Publisher: AUTOSAR