AUTOSAR CanIf User Manual

Document Change History			
Date (YYYY-MM-DD)	Ver.	Editor	Content
2021-Aug-23	1.0.0	HieuTM8	Initial Version
2021-Aug-26	1.0.0.6	Seungjae Kim	Update Change Log
2022-Feb-21	1.0.1.0	HieuTM8	Update Change Log
2022-Mar-18	1.0.2.0	HieuTM8	Update Module version Update Change Log Update default value CanIfRxPduCanIdMask Update default value CanIfTxPduCanIdMask
2022-Apr-28	1.0.3.0	HieuTM8	Update Module version Update Change Log
2022-May-09	1.0.4.0	Jiwon Oh	Update Module version Update Change Log
2022-June-30	1.0.5.0	Jiwon Oh	Update Module version Update Change Log
2022-Aug-02	1.0.6.0	HieuTM8	Update Module version Update Change Log
2022-Aug-22	1.0.7.0	Jiwon Oh	Update Module version Update Change Log
2022-Aug-31	1.0.8.0	HieuTM8	Update Module version Update Change Log Update ERR0600013
2022-Oct-4	1.0.9.0	HieuTM8	Update Module version Update Change Log
2022-Oct-28	1.0.10.0	HieuTM8	Update Module version Update Change Log Update ERR0600027
2022-Nov-2	1.0.11.0	HieuTM8	Update Module version Update Change Log
2022-Dec-16	1.0.12.0	HieuTM8	Update Module version Update Change Log

2023-Jan-18	1.0.13.0	HieuTM8	Update Module version Update Change Log Add ERR0600032, ERR0600033, ERR0600035
2023-Jun-13	1.0.14.0	HieuTM8	Update Module version Update Change Log Update ERR0600018, ERR0600033, ERR0600035
2024-Jan-23	1.0.15.0	JH Jang	Update Module version Update Change Log
2024-Mar-14	1.1.0.0	JH Yang	Add to Configuration (CanIfTxRxMonitoringSupport) Add ERR0600036, ERR0600037 Remove WRN0600003, WRN0600006 Update Limitation and Deviations

Table of Contents

1 Overview

It is written based on AUTOSAR standard SRS / SWS. If more detailed functional explanation is needed when using the module, see the Reference Manual. The interpretation of setting related category is as follows:

- Changeable (C): Items that can be set by the user
- Fixed (F): Items that cannot be changed by the user.
- Not Supported (N): Deprecated item

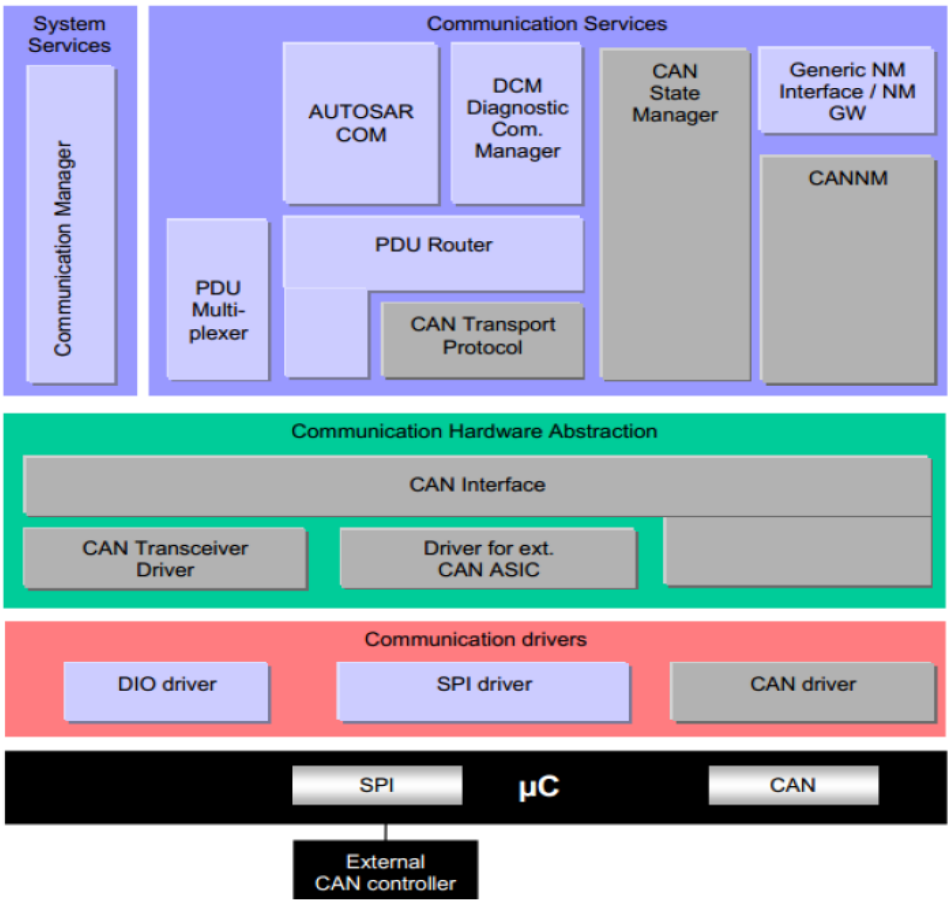
2 Reference

Sl. No.	Title	Version
1	AUTOSAR_SWS_CanInterface.pdf	4.4.0

3 AUTOSAR System

3.1 Overview of Software Layers

CanIf module is an interface module for CAN communication.



4 Limitations and Deviations

4.1 Limitations

- Software Filter Type

  In case of receiving CAN message by BASIC CAN method, Table and Index methods are not supported among the filter types that select valid messages among the received areas.

- TTCAN Support Not Supported

  Time-Triggered CAN is not supported.

4.2 Deviations

- CAN Controller ID

  The CanIfCtrlId value is set to the same value as the CanControllerId of the connected MCAL CAN controller.

- CAN Transceiver ID

  The CanIfTrcvId value is set to the same value as the CanTrcvChannelId of the connected CAN Transceiver module.

- CanCM Support

  In order to limit the transmission function according to the voltage, an interface called by the CanCM module is provided.

- IdsM Support

  IdsM module provides the function of receiving message and transmitting CAN controller status information.

- Hrh Range

  Only one CanIfHrhRangeCfg can be supported in each CanIfHrhCfg.

5 Configuration Guide

The CanIf setting of the AUTOSAR platform distributed by Hyundai Auto is a setting reflecting Hyundai Auto Policy's policy. Therefore, you should consult with Hyundai Auto.

5.0 General Settings

Implementation Config Variant:

- VARIANT-PRE-COMPILE: Run on Pre-compile
- VARIANT-POST-BUILD / VARIANT-POST-BUILD-SELECTABLE: Run on PostBuild Selectable

  + Post Build Variant Used : True

5.1 CanIfCtrlDrvCfg Settings

Parameter Name	Value	Category
1) CanIfCtrlDrvInitHohConfigRef	Automated	F
2) CanIfCtrlDrvNameRef	Automated	F

1) CanIfCtrlDrvInitHohConfigRef
   - Reference to the Init Hoh Configuration
2) CanIfCtrlDrvNameRef
   - CAN Interface Driver Reference

5.2 CanIfCtrlCfg Settings

Parameter Name	Value	Category
1) CanIfCtrlId	Automated	F
2) CanIfCtrlWakeupSupport	True / False	C
3) CanIfCtrlCanCtrlRef	Automated	F

1) CanIfCtrlId
   - This parameter 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
2) CanIfCtrlWakeupSupport
   - This parameter defines if a respective controller of the referenced CAN Driver modules is queriable for wake-up events.
3) CanIfCtrlCanCtrlRef
   - This parameter 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

5.3 CanIfDispatchCfg Settings

Parameter Name	Value	Category
1) CanIfDispatchUserCheckTrcvWakeFlagIndicationUL	CAN_SM	C
2) CanIfDispatchUserCheckTrcvWakeFlagIndicationName	-	C

3) CanIfDispatchUserClearTrcvWufFlagIndicationUL	CAN_SM	C
4) CanIfDispatchUserClearTrcvWufFlagIndicationName	-	C
5) CanIfDispatchUserConfirmPnAvailabilityUL	CAN_SM	C
6) CanIfDispatchUserConfirmPnAvailabilityName	-	C
7) CanIfDispatchUserCtrlBusOffUL	CAN_SM	C
8) CanIfDispatchUserCtrlBusOffName	-	C
9) CanIfDispatchUserCtrlModeIndicationUL	CAN_SM	C
10) CanIfDispatchUserCtrlModeIndicationName	-	C
11) CanIfDispatchUserTrcvModeIndicationUL	CAN_SM	C
12) CanIfDispatchUserTrcvModeIndicationName	-	C
13) CanIfDispatchUserValidateWakeupEventUL	ECUM	C
14) CanIfDispatchUserValidateWakeupEventName	-	C

1) CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
   - This parameter defines the upper layer module to which the CheckTrcvWakeFlagIndication from the Driver modules have to be routed via 〈User_CheckTrcvWakeFlagIndication〉
2) CanIfDispatchUserCheckTrcvWakeFlagIndicationName
   - This parameter defines the name of 〈User_CheckTrcvWakeFlagIndication〉

   (This parameter is a required setting only if CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is selected as a CDD. Write the function name of the CDD to be called when CanIf_CheckTrcvWakeFlagIndication occurs. In CDD, after performing the required logic in that function, the original parent function of CanIf, CanSM_CheckTrcvWakeFlagIndication, must be called and delivered.)

3) CanIfDispatchUserClearTrcvWufFlagIndicationUL
   - This parameter defines the upper layer module to which the ClearTrcvWufFlagIndication from the Driver modules have to be routed via 〈User_ClearTrcvWufFlagIndication〉
4) CanIfDispatchUserClearTrcvWufFlagIndicationName
   - This parameter defines the name of 〈User_ClearTrcvWufFlagIndication〉

   (This parameter is a required setting only if CanIfDispatchUserClearTrcvWufFlagIndicationUL is selected as a CDD. Write the function name of the CDD to be called when CanIf_ClearTrcvWufFlagIndication occurs. In CDD, after performing the required logic in that function, the original parent function of CanIf, CanSM_ClearTrcvWufFlagIndication, must be called and delivered.)

5) CanIfDispatchUserConfirmPnAvailabilityUL
   - This parameter defines the upper layer module to which the ConfirmPnAvailability notification from the Driver modules have to be routed via 〈User_ConfirmPnAvailability〉
6) CanIfDispatchUserConfirmPnAvailabilityName
   - This parameter defines the name of 〈User_ConfirmPnAvailability〉

   (This parameter is a required setting only if CanIfDispatchUserConfirmPnAvailabilityUL is selected as a CDD. Write the function name of the CDD to be called when CanIf_ConfirmPnAvailability occurs. In CDD, after performing the required logic in that function, the original parent function of CanIf, CanSM_ConfirmPnAvailability, must be called and delivered.)

7) CanIfDispatchUserCtrlBusOffUL

- This parameter defines the upper layer module to which the notifications of all ControllerBusOff events from the CAN Driver modules have to be routed via <User_ControllerBusOff>

8) CanIfDispatchUserCtrlBusOffName
   - This parameter defines the name of <User_ControllerBusOff>

   (This parameter is a required setting only if CanIfDispatchUserCtrlBusOffUL is selected as a CDD. Write the function name of the CDD to be called when CanIf_ControllerBusOff occurs. In CDD, after performing the required logic in that function, the original parent function of CanIf, CanSM_ControllerBusOff, must be called and delivered.)

9) CanIfDispatchUserCtrlModeIndicationUL
   - This parameter defines the upper layer module to which the notifications of all ControllerTransition events from the CAN Driver modules have to be routed via <User_ControllerModeIndication>

10) CanIfDispatchUserCtrlModeIndicationName
   - This parameter defines the name of <User_ControllerModeIndication>

   (This parameter is a required setting only if CanIfDispatchUserCtrlModeIndicationUL is selected as a CDD. Write the function name of the CDD to be called when CanIf_ControllerModeIndication occurs. In CDD, after performing the required logic in that function, the original parent function of CanIf, CanSM_ControllerModeIndication, must be called and delivered.)

11) CanIfDispatchUserTrcvModeIndicationUL
   - This parameter defines the upper layer module to which the notifications of all TransceiverTransition events from the CAN Transceiver Driver modules have to be routed via <User_TrcvModeIndication>
   - If no UL module is configured, no upper layer callback function will be called.

12) CanIfDispatchUserTrcvModeIndicationName
   - This parameter defines the name of <User_TrcvModeIndication>

   (This parameter is a required setting only if CanIfDispatchUserTrcvModeIndicationUL is selected as a CDD. Write the function name of the CDD to be called when CanIf_TrcvModeIndication occurs. In CDD, after performing the required logic in that function, the original parent function of CanIf, CanSM_TrcvModeIndication, must be called and delivered.)

13) CanIfDispatchUserValidateWakeupEventUL
   - This parameter defines the upper layer module to which the notifications about positive former requested wake up sources have to be routed via <User_ValidateWakeupEvent>

14) CanIfDispatchUserValidateWakeupEventName
   - This parameter defines the name of <User_ValidateWakeupEvent>

   (This parameter is a required setting only if CanIfDispatchUserValidateWakeupEventUL is selected as a CDD. Write the function name of the CDD to be called when CanIf_ValidateWakeupEvent occurs. In CDD, after performing the required logic in that function, the original parent function of CanIf, EcuM_ValidateWakeupEvent, must be called and delivered.)

5.4 CanIfPrivateCfg Settings

Parameter Name	Value	Category
1) CanIfFixedBuffer	True / False	C
2) CanIfPrivateDataLengthCheck	True / False	C
3) CanIfPrivateSoftwareFilterType	BINARY	F
4) CanIfSupportTTCAN	False	F
5) CanIfSupportCanCM	True	F

6) CanIfSupportCANFD	True	F

1) CanIfFixedBuffer
   - This parameter defines if the buffer element length shall be fixed to 8 Bytes for buffers to which only PDUs < 8 Bytes are assigned.
2) CanIfPrivateDataLengthCheck
   - Selects whether Data Length Check is supported.
   - Set DLC check for received CAN frame (requires use of CanIfPublicDevErrorDetect function)
3) CanIfPrivateSoftwareFilterType
   - Selects the desired software filter mechanism for reception only.
   - Set the filter type to be used when BASIC CAN method is used for CAN frame reception.
4) CanIfSupportTTCAN
   - Defines whether TTCAN is supported.
5) CanIfSupportCanCM
   - Defines whether CanCM is supported.
   - Set true to control transmission according to battery voltage using CanCM module.
6) CanIfSupportCANFD
   - Defines whether CAN-FD is supported.
   - Set to true if CAN-FD frame transmission / reception is required.

5.5 CanIfPublicCfg Settings

Parameter Name	Value	Category
1) CanIfBusMirroringSupport	True / False	C
2) CanIfPublicCddHeaderFile	-	C
3) CanIfDevErrorDetect	True / False	C
4) CanIfPublicHandleTypeEnum	UINT16	F
5) CanIfPublicIcomSupport	True/False	C
6) CanIfPublicMultipleDrvSupport	True/False	C
7) CanIfPublicPnSupport	True/False	C
8) CanIfPublicReadRxPduDataApi	True/False	C
9) CanIfPublicReadRxPduNotifyStatusApi	True/False	C
10) CanIfPublicReadTxPduNotifyStatusApi	True/False	C
11) CanIfPublicSetDynamicTxIdApi	True/False	C
12) CanIfPublicTxBuffering	True/False	C
13) CanIfPublicTxConfirmPollingSupport	True/False	C
14) CanIfVersionInfoApi	True/False	C
15) CanIfWakeupSupport	True/False	C
16) CanIfPublicWakeupCheckValidByNM	True/False	C
17) CanIfPublicWakeupCheckValidSupport	True/False	C
18) CanIfSetBaudrateApi	True/False	C
19) CanIfTriggerTransmitSupport	True/False	C
20) CanIfTxOfflineActiveSupport	True/False	C
21) CanIfMetaDataSupport	True/False	C
22) CanIfAutronTrcvDrvSupport	True	F
23) CanIfExternalTrcvDrvSupport	False	F

24) CanIfCANFDID16BitSupport	False	F
25) CanIfCANFDDiscreteDlcSupport	False	F
26) CanIfBusloadDetectingSupport	-	N
27) CanIfTxRxMonitoringSupport	True/False	C

1) CanIfBusMirroringSupport
   - Enable support for Bus Mirroring.
2) CanIfPublicCddHeaderFile
   - Defines header files for callback functions which shall be included in case of CDDs. Range of characters is 1.. 32.
   - If the user develops a CDD that communicates directly with the CanIf module, the CDD must provide an indication function to the CanIf module, so add the header file in which the function is declared.
3) CanIfDevErrorDetect
   - Switches the development error detection and notification on or off.
   - DET function On / Off setting. It is basically used as True and can be set as False by user request and agreement.
4) CanIfPublicHandleTypeEnum
   - This parameter is used to configure the Can_HwHandleType.
   - The Can_HwHandleType represents the hardware object handles of a CAN hardware unit. For CAN hardware units with more than 255 HW objects the extended range shall be used (UINT16).
5) CanIfPublicIcomSupport
   - Selects support of Pretended Network features in CanIf.
6) CanIfPublicMultipleDrvSupport
   - Selects support for multiple CAN Drivers.
7) CanIfPublicPnSupport
   - Selects support of Partial Network features in CanIf.
8) CanIfPublicReadRxPduDataApi
   - Enables / Disables the API CanIf_ReadRxPduData() for reading received L-SDU data.
   - Configuration when reading CAN frame data received by CanIf module directly from user developed CDD.
9) CanIfPublicReadRxPduNotifyStatusApi
   - Enables and disables the API for reading the notification status of receive L-PDUs.
   - Provide API to check whether CanIf module can receive CAN frame in user developed CDD.
10) CanIfPublicReadTxPduNotifyStatusApi
   - Enables and disables the API for reading the notification status of transmit L-PDUs.
   - Provide API to check whether CanIf module can transmit CAN frame to user developed CDD.
11) CanIfPublicSetDynamicTxIdApi
   - Enables and disables the API for reconfiguration of the CAN Identifier for each Transmit L-PDU.
12) CanIfPublicTxBuffering
   - Enables and disables the buffering of transmit L-PDUs (rejected by the CanDrv) within the CAN Interface module.
   - Used when the number of CAN hardware buffers available for transmission is smaller than the number of messages to transmit.
13) CanIfPublicTxConfirmPollingSupport
   - Configuration parameter to enable/disable the API to poll for Tx Confirmation state.
14) CanIfVersionInfoApi
   - Enables and disables the API for reading the version information about the CAN Interface.

15) CanIfWakeupSupport
- Enables the CanIf_CheckWakeup API at Pre-Compile-Time. Therefore, this parameter defines if there shall be support for wake-up.

16) CanIfPublicWakeupCheckValidByNM
- If enabled, only NM messages shall validate a detected wake-up event in CanIf.

17) CanIfPublicWakeupCheckValidSupport
- Selects support for wake-up validation.
- Setting whether to perform Wake Up validation by CAN.

18) CanIfSetBaudrateApi
- Configuration parameter to enable/disable the CanIf_SetBaudrate API to change the baud rate of a CAN Controller.

19) CanIfTriggerTransmitSupport
- Enables the CanIf_TriggerTransmit API at Pre-Compile-Time. Therefore, this parameter defines if there shall be support for trigger transmit transmissions.

20) CanIfTxOfflineActiveSupport
- Determines whether TxOffLineActive feature (see SWS_CANIF_00072) is supported by CanIf.

21) CanIfMetaDataSupport
- Enable support for dynamic ID handling using L-SDU MetaData.
- Whether to use Meta Data when using J1939 type CAN communication

22) CanIfAutronTrcvDrvSupport
- Selects support for Autron CanTrcv Drv

23) CanIfExternalTrcvDrvSupport
- Selects support for External CanTrcv Drv

24) CanIfCANFDID16BitSupport
- Selects support for CAN FD 16Bit CAN ID
- Set by the platform distributor, check if the MCAL used in the platform uses 31bit as the CAN-FD MASK Bit for CAN-FD processing.

25) CanIfCANFDDiscreteDlcSupport
- Selects support for CAN FD discrete DLC.
- Set by platform distributor and check if it is uploaded after using DLC for CAN-FD in MCAL.

26) CanIfBusloadDetectingSupport
- Selects support for CAN Busload Detecting

27) CanIfTxRxMonitoringSupport
- Set this if you want to know whether message transmission and reception for each Pdu are performed normally.
- Variables to confirm receipt: CanIf_ReceiveCnt
- Variables to confirm transmission: CanIf_TransmitCnt

5.6 CanIfRxPduCfg Settings

Parameter Name	Value	Category
1) CanIfRxPduCanId	Automated	F
2) CanIfRxPduCanIdType	Automated	F
3) CanIfRxPduDataLength	Automated	C
4) CanIfRxPduDataLengthCheck	True / False	C
5) CanIfRxPduId	Automated	F
6) CanIfRxPduReadData	False	F

7) CanIfRxPduReadNotifyStatus	False	F
8) CanIfRxPduUserRxIndicationUL	Automated	C
9) CanIfRxPduUserRxIndicationName	Automated	C
10) CanIfRxPduHrhIdRef	Automated	F
11) CanIfRxPduRef	Automated	F
12) CanIfRxPduCanIdMask	Automated	C
13) CanIfPduReportIdsMEnable	Automated	C

1) CanIfRxPduCanId
   - CAN Identifier of Receive CAN L-PDUs used by the CAN Interface. Exa: Software Filtering.
   - This parameter is used if exactly one Can Identifier is assigned to the Pdu. If a range is assigned, then the CanIfRxPduCanIdRange parameter shall be used.
2) CanIfRxPduCanIdType
   - CAN Identifier of receive CAN L-PDUs used by the CAN Driver for CAN L-PDU reception.
3) CanIfRxPduDataLength
   - Data length of the received CAN L-PDUs used by the CAN Interface.
4) CanIfRxPduDataLengthCheck
   - This parameter switches the message specific data length check.
5) CanIfRxPduId
   - ECU wide unique, symbolic handle for receive CAN L-SDU. It shall fulfill ANSI/AUTOSAR definitions for constant defines.
6) CanIfRxPduReadData
   - Enables and disables the Rx buffering for reading of received L-SDU data.
   - Whether to use an API to read PDU data
7) CanIfRxPduReadNotifyStatus
   - Enables and disables receive indication for each receive CAN L-SDU for reading its notification status.
   - Whether to use API to check PDU reception status
8) CanIfRxPduUserRxIndicationUL
   - This parameter defines the upper layer module to which the indication of the successfully received CANRXPDUID has to be routed via <User_RxIndication>.
   - This <User_RxIndication> has to be invoked when the indication of the configured CANRXPDUID will be received by an Rx indication event from the CAN Driver module. If no upper layer module is configured, no <User_RxIndication> has to be called in case of an Rx indication event of the CANRXPDUID from the CAN Driver module.
   - Specifies the upper module to which the corresponding PDU notification is delivered. It is set automatically at CANDB Import but can be designated as CDD when user develops CDD that needs PDU reception.
9) CanIfRxPduUserRxIndicationName
   - This parameter defines the name of the <User_RxIndication>.
   - Designate the function provided by CDD.
10) CanIfRxPduHrhIdRef
   - The HRH to which Rx L-PDU belongs to, is referred through this parameter.
11) CanIfRxPduRef
   - Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.
12) CanIfRxPduCanIdMask
   - 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 bits for Standard CAN Identifier, 29 bits for Extended CAN Identifier.

13) CanIfPduReportIdsMEnable
- If true CanIf Module shall report to IdsM Module when PDU received.
- Setting that decides whether to transmit message reception information to IdsM module.

5.7 CanIfTxPduCfg Settings

Parameter Name	Value	Category
1) CanIfTxPduCanId	Automated	F
2) CanIfTxPduCanIdType	Automated	F
3) CanIfTxPduId	Automated	F
4) CanIfTxPduReadNotifyStatus	True / False	C
5) CanIfTxPduTriggerTransmit	Automated	C
6) CanIfTxPduTruncation	True / False	C
7) CanIfTxPduType	Automated	F
8) CanIfTxPduUserTriggerTransmitName	-	C
9) CanIfTxPduUserTxConfirmationUL	Automated	C
10) CanIfRxPduUserTxConfirmationName	-	C
11) CanIfTxPduBufferRef	Automated	F
12) CanIfTxPduRef	Automated	F
13) CanIfTxPduPnFilterPdu	True / False	C
14) CanIfTxPduCanIdMask	Automated	C

1) CanIfTxPduCanId
- CAN Identifier of transmit CAN L-PDUs used by the CAN Driver for CAN L-PDU transmission.

2) CanIfTxPduCanIdType
- Type of CAN Identifier of the transmit CAN L-PDU used by the CAN Driver module for CAN L-PDU transmission.

3) CanIfTxPduId
- ECU wide unique, symbolic handle for transmit CAN L-SDU.

4) CanIfTxPduReadNotifyStatus
- Enables and disables transmit confirmation for each transmit CAN L-SDU for reading its notification status.

5) CanIfTxPduTriggerTransmit
- Determines if or if not CanIf shall use the trigger transmit API for this PDU.

6) CanIfTxPduTruncation
- Enables/disables truncation of PDUs that exceed the configured size.

7) CanIfTxPduType
- Defines the type of each transmit CAN L-PDU.

8) CanIfTxPduUserTriggerTransmitName
- This parameter defines the name of the <User_TriggerTransmit>.
- This parameter depends on the parameter CanIfTxPduUserTxConfirmationUL.
- Specify a function provided by CDD.

9) CanIfTxPduUserTxConfirmationUL

- This parameter defines the upper layer (UL) module to which the confirmation of the successfully transmitted CanTxPduId has to be routed via the <User_TxConfirmation>.
- Specifies the upper module to which the corresponding PDU transmission completion notification is delivered. It is set automatically at CANDB Import but can be designated as CDD when user develops CDD that needs PDU transmission.

10) CanIfRxPduUserTxConfirmationName
- This parameter defines the name of the <User_TxConfirmation>.
- This parameter depends on the parameter CanIfTxPduUserTxConfirmationUL.
- Specify a function provided by CDD.

11) CanIfTxPduBufferRef
- Configurable reference to a CanIf buffer configuration.

12) CanIfTxPduRef
- Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.

13) CanIfTxPduPnFilterPdu
- If CanIfPublicPnFilterSupport is enabled, by this parameter PDUs could be configured which will pass the CanIfPnFilter.

14) CanIfTxPduCanIdMask
- Type of CAN Identifier of the transmit CAN L-PDU used by the CAN Driver module for CAN L-PDU transmission.

5.8 CanIfBufferCfg Settings

Parameter Name	Value	Category
1) CanIfBufferSize	Automated	C
2) CanIfBufferHthRef	Automated	F

1) CanIfBufferSize
- 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.

(User can change the value as needed after execution of configuration automation.)

2) CanIfBufferHthRef
- 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.

(User can change the value as needed after execution of configuration automation.)

5.9 CanIfTrcvDrvCfg Settings

Parameter Name	Value	Category
1) CanIfTrcvId	Automated	F
2) CanIfTrcvWakeupSupport	True / False	C
3) CanIfTrcvCanTrcvRef	Automated	F

1) CanIfTrcvId
   - This parameter 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.
2) CanIfTrcvWakeupSupport
   - This parameter defines if a respective transceiver of the referenced CAN Transceiver Driver modules is queriable for wake-up events.
3) CanIfTrcvCanTrcvRef
   - This parameter references to the logical handle of the underlying CAN
   - transceiver from the CAN transceiver driver module to be served by the CAN Interface module.

5.10 CanIfInitCfg Settings

Parameter Name	Value	Category
1) CanIfInitCfgSet	-	C
2) CanIfMaxBufferSize	-	N
3) CanIfMaxRxPduCfg	-	N
4) CanIfMaxTxPduCfg	-	N

1) CanIfInitCfgSet
   - Selects the CAN Interface specific configuration setup. This type of the external data structure shall contain the post build initialization data for the CAN Interface for all underlying CAN Dirvers.
2) CanIfMaxBufferSize
   - Maximum total size of all Tx buffers. This parameter is needed only in case of post-build loadable implementation using static memory.
3) CanIfMaxRxPduCfg
   - Maximum number of Pdus. This parameter is needed only in case of post-build loadable implementation using static memory allocation.
4) CanIfMaxTxPduCfg
   - Maximum number of Pdus. This parameter is needed only in case of post-build loadable implementation using static memory allocation.

5.11 PostBuild Settings

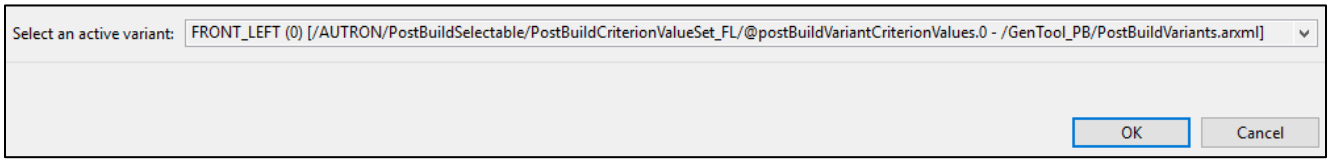
- Change Mode to PostBuild:

Implementation Config Variant set to <VARIANT_POST_BUILD/ VARIANT_POST_BUILD _SELECTABLE>

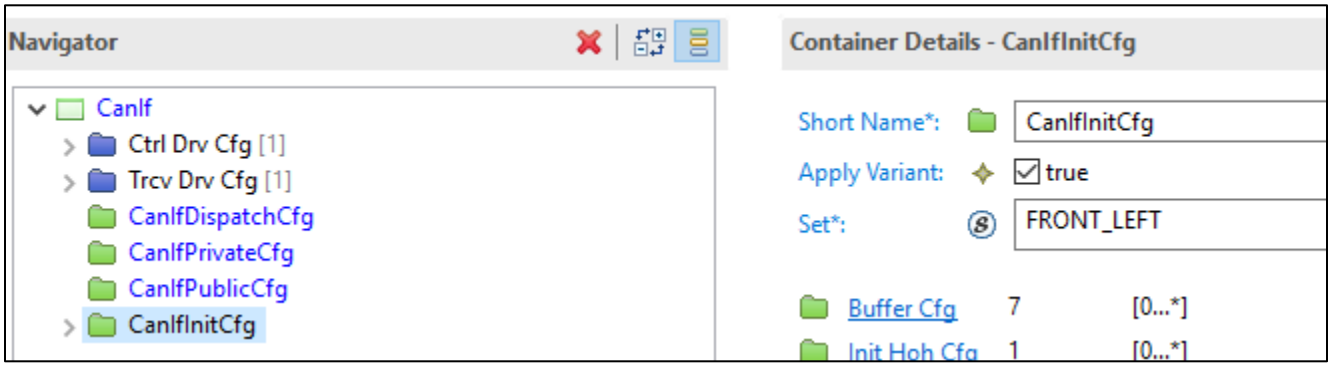
Post Build Variant Used set to <True>

- Apply variant:

+ Select Variant on RTU.



+ CanIfInitCfg – Apply Variant set to <True> (all sub container of this will be belong to this Variant).



6 Application Programming Interface (API)

6.1 Type Definitions

6.1.1 PduModeType

Range	
CANIF_OFFLINE	= 0 Transmit and receive path of the corresponding channel are disabled => no communication mode.
CANIF_TX_OFFLINE	Transmit path of the corresponding channel is disabled. The receive path is enabled.
CANIF_TX_OFFLINE_ACTIVE	Transmit path of the corresponding channel is in offline active mode. The receive path is disabled. This mode requires CanIfTxOfflineActiveSupport = TRUE.
CANIF_ONLINE	Transmit and receive path of the corresponding channel are enabled => full operation mode

6.1.2 NotifStatusType

Range	
CANIF_TX_RX_NOTIFICATION	The requested Rx/Tx CAN L-PDU was successfully transmitted or received.
CANIF_NO_NOTIFICATION	No transmit or receive event occurred for the requested L-PDU.

6.2 Macro Constants

None

6.3 Functions

6.3.1 Initialization

Function Name	CanIf_Init	
Syntax	FUNC(void, CANIF_CODE) CanIf_Init (P2CONST(CanIf_ConfigType, AUTOMATIC, CANIF_APPL_CONST) ConfigPt)	
Service ID [Hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (In)	ConfigPtr	Pointer to configuration parameter set, used e.g. for post build parameters

Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	This service Initializes internal and external interfaces of the CAN Interface for the further processing. This function is used by BSW.
Preconditions	None
Configuration Dependency	None
Available via	CanIf.h

Function Name	CanIf_DeInit
Syntax	FUNC(void, CANIF_CODE) CanIf_DeInit ()
Service ID [Hex]	0x02
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	De-initializes the CanIf module. This function is used by BSW.
Preconditions	None
Configuration Dependency	None
Available via	CanIf.h

Function Name	CanIf_GetVersionInfo	
Syntax	FUNC(void, CANIF_CODE)CanIf_GetVersionInfo(P2VAR(Std_VersionInfoType, AUTOMATIC, CANIF_APPL_DATA)VersionInfo)	
Service ID [Hex]	0x0b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	None	
Parameters (Inout)	None	
Parameters (Out)	VersionInfo	Pointer to where to store the version information of this module.
Return Value	None	
Description	This service returns the version information of the called CAN Interface module. This function is used by user. But it needs configuration. (It cannot be called directly by user)	
Preconditions	None	

Configuration Dependency	None
Available via	CanIf.h

6.3.2 Transmission

Function Name	CanIf_TriggerTransmit	
Syntax	FUNC(Std_ReturnType, CANIF_CODE) CanIf_TriggerTransmit (VAR(PduIdType, CANIF_VAR) TxPduId, P2CONST(PduInfoType, AUTOMATIC, CANIF_APPL_CONST) PduInfoPtr)	
Service ID [Hex]	0x41	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (In)	TxPduId	ID of the SDU that is requested to be transmitted.
Parameters (Inout)	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLengh. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU data has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description	Within this API, the upper layer module (called module) shall check whether the available data fits into the buffer size reported by PduInfoPtr->SduLength. If it fits, it shall copy its data into the buffer provided by PduInfoPtr -> SduDataPtr and update the length of the actual copied data in PduInfoPtr -> SduLength. If not, it returns E_NOT_OK without changingPduInfoPtr. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one TxPdu should be configured.	
Available via	CanIf.h	

Function Name	CanIf_Transmit
Syntax	FUNC(Std_ReturnType, CANIF_CODE) CanIf_Transmit (

	VAR(PduIdType, CANIF_VAR) TxPduId, P2CONST(PduInfoType, AUTOMATIC, CANIF_APPL_CONST) PduInfoPtr)	
Service ID [Hex]	0x49	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (In)	TxPduId	Identifier of the PDU to be transmitted.
	PduInfoPtr	Length of and pointer to the PDU data and pointer to MetaData.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: Transmit request has been accepted. E_NOT_OK: Transmit request has not been accepted.
Description	Requests transmission of a PDU. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one TxPdu should be configured.	
Available via	CanIf.h	

Function Name	CanIf_TxConfirmation	
Syntax	FUNC(void, CANIF_CODE) CanIf_TxConfirmation (VAR(PduIdType, CANIF_VAR) CanIfTxSduId)	
Service ID [Hex]	0x13	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	CanTxPduId	L-PDU handle of CAN L-PDU successfully transmitted. This ID specifies the corresponding CAN L-PDU ID and implicitly the CAN Driver instance as well as the corresponding CAN controller device.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	
Description	This service confirms a previously successfully processed transmission of a CAN TxPDU. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one TxPdu should be configured.	

Available via	CanIf_Can.h

Function Name	CanIf_GetTxConfirmationState	
Syntax	FUNC(CanIf_NotifStatusType, CANIF_CODE)CanIf_GetTxConfirmationState (VAR(uint8, CANIF_VAR) ControllerId)	
Service ID [Hex]	0x19	
Sync/Async	Synchronous	
Reentrancy	Reentrant (Not for the same controller)	
Parameters (In)	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	CanIf_NotifStatusType	Combined TX confirmation status for all TX PDUs of the CAN controller.
Description	This service reports, if any TX confirmation has been done for the whole CAN controller since the last CAN controller start. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	CANIF_TXCONFIRM_POLLING_SUPPORT should be configured as STD_ON.	
Available via	CanIf.h	

6.3.3 Reception

Function Name	CanIf_ReadRxPduData	
Syntax	FUNC(Std_ReturnType, CANIF_CODE)CanIf_ReadRxPduData (VAR(PduIdType, CANIF_VAR) CanIfRxSduId, P2VAR(PduInfoType, AUTOMATIC, CANIF_APPL_DATA) PduInfoPtr)	
Service ID [Hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (In)	CanIfRxSduId	Receive L-SDU handle specifying the corresponding CAN L-SDU ID and implicitly the CAN Driver instance as well as the corresponding CAN controller device.
Parameters (Inout)	None	
Parameters (Out)	PduInfoPtr	Contains the length (SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU, and the MetaData related to this PDU.

Return Value	Std_ReturnType	E_OK: Request for L-SDU data has been accepted E_NOT_OK: No valid data has been received
Description	This service provides the Data Length and the received data of the requested CanIfRxSduId to the calling upper layer. This function is used by user. But it needs configuration. (It cannot be called directly by user)	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one RxPdu should be configured and CANIF_READRXPDU_DATA_API should be configured as STD_ON.	
Available via	CanIf.h	

Function Name	CanIf_ReadRxNotifStatus	
Syntax	FUNC(CanIf_NotifStatusType, CANIF_CODE) CanIf_ReadRxNotifStatus (VAR(PduIdType, CANIF_VAR) CanIfRxSduId)	
Service ID [Hex]	0x08	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (In)	CanIfRxSduId	Receive L-SDU handle specifying the corresponding CAN L-SDU ID and implicitly the CAN Driver instance as well as the corresponding CAN controller device.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	CanIf_NotifStatusType	Current indication status of the corresponding CAN Rx L-PDU.
Description	This service returns the indication status (indication occurred or not) of a specific CAN Rx L-PDU, requested by the CanIfRxSduId. This function is used by user. But it needs configuration. (It cannot be called directly by user)	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one RxPdu should be configured and CANIF_READRXPDU_NOTIFY_STATUS_API should be configured as STD_ON.	
Available via	CanIf.h	

Function Name	CanIf_ReadTxNotifStatus
Syntax	FUNC(CanIf_NotifStatusType, CANIF_CODE) CanIf_ReadTxNotifStatus (VAR(PduIdType, CANIF_VAR) CanIfTxSduId)
Service ID [Hex]	0x07

Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (In)	CanIfTxSduId	L-SDU handle to be transmitted. This handle specifies the corresponding CAN LSDU ID and implicitly the CAN Driver instance as well as the corresponding CAN controller device.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	CanIf_NotifStatusType	Current confirmation status of the corresponding CAN Tx L-PDU.
Description	This service returns the confirmation status (confirmation occurred or not) of a specific static or dynamic CAN Tx L-PDU, requested by the CanIfTxSduId. This function is used by user. But it needs configuration. (It cannot be called directly by user)	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one TxPdu should be configured and CANIF_READTXPDU_NOTIFY_STATUS_API should be configured as STD_ON.	
Available via	CanIf.h	

Function Name	CanIf_RxIndication	
Syntax	FUNC(void, CANIF_CODE) CanIf_RxIndication(P2CONST(Can_HwType, AUTOMATIC, CANIF_APPL_CONST) Mailbox, P2CONST(PduInfoType, AUTOMATIC, CANIF_APPL_CONST) PduInfoPtr)	
Service ID [Hex]	0x14	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	Mailbox	Identifies the HRH and its corresponding CAN Controller.
	PduInfoPtr	Pointer to the received L-PDU
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	
Description	This service indicates a successful reception of a received CAN Rx LPDU to the CanIf after passing all filters and validation checks. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	None	
Available via	CanIf_Can.h	

6.3.4 Control Mode

Function Name	CanIf_SetControllerMode	
Syntax	FUNC(Std_ReturnType, CANIF_CODE)CanIf_SetControllerMode (VAR(uint8, CANIF_VAR) ControllerId, VAR(Can_ControllerStateType, CANIF_VAR) ControllerMode)	
Service ID [Hex]	0x03	
Sync/Async	Asynchronous	
Reentrancy	Reentrant (Not for the same controller)	
Parameters (In)	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller, which is requested for mode transition.
	ControllerMode	Requested mode transition.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: Controller mode request has been accepted E_NOT_OK: Controller mode request has not been accepted
Description	This service calls the corresponding CAN Driver service for changing of the CAN controller mode. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	None	
Available via	CanIf.h	

Function Name	CanIf_GetControllerMode	
Syntax	FUNC(Std_ReturnType, CANIF_CODE)CanIf_GetControllerMode (VAR(uint8, CANIF_VAR) ControllerId, P2VAR(Can_ControllerStateType, AUTOMATIC, CANIF_APPL_DATA) ControllerModePtr)	
Service ID [Hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (In)	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller, which is requested for current operation mode.
Parameters (Inout)	None	

Parameters (Out)	ControllerModePtr	Pointer to a memory location, where the current mode of the CAN controller will be stored.
Return Value	Std_ReturnType	E_OK: Controller mode request has been accepted. E_NOT_OK: Controller mode request has not been accepted.
Description	This service calls the corresponding CAN Driver service for obtaining the current status of the CAN controller. This function is used by user. But it needs configuration. (It cannot be called directly by user)	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	None	
Available via	CanIf.h	

Function Name	CanIf_SetPduMode	
Syntax	FUNC(Std_ReturnType, CANIF_CODE) CanIf_SetPduMode(VAR(uint8, CANIF_VAR) ControllerId, VAR(CanIf_PduModeType, CANIF_VAR) PduModeRequest)	
Service ID [Hex]	0x09	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (In)	ControllerId	All PDUs of the own ECU connected to the corresponding CanIf ControllerId, which is assigned to a physical CAN controller are addressed.
	PduModeRequest	Requested PDU mode change.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: Request for mode transition has been accepted. E_NOT_OK: Request for mode transition has not been accepted.
Description	This service sets the requested mode at the L-PDUs of a predefined logical PDU channel. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	None	
Available via	CanIf.h	

Function Name	CanIf_GetPduMode
Syntax	FUNC(Std_ReturnType, CANIF_CODE)CanIf_GetPduMode (

	VAR(uint8, CANIF_VAR) ControllerId, P2VAR(CanIf_PduGetModeType, AUTOMATIC, CANIF_APPL_DATA) PduModePtr)	
Service ID [Hex]	0x0a	
Sync/Async	Synchronous	
Reentrancy	Reentrant (Not for the same channel)	
Parameters (In)	ControllerId	All PDUs of the own ECU connected to the corresponding CanIf ControllerId, which is assigned to a physical CAN controller are addressed.
Parameters (Inout)	None	
Parameters (Out)	PduModePtr	Pointer to a memory location, where the current mode of the logical PDU channel will be stored.
Return Value	Std_ReturnType	E_OK: PDU mode request has been accepted E_NOT_OK: PDU mode request has not been accepted
Description	This service reports the current mode of a requested PDU channel. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	None	
Available via	CanIf.h	

Function Name	CanIf_SetDynamicTxId	
Syntax	FUNC(Std_ReturnType, CANIF_CODE) CanIf_SetDynamicTxId (VAR(PduIdType, CANIF_VAR) CanIfTxSduId, VAR(Can_IdType, CANIF_VAR) CanId)	
Service ID [Hex]	0x0c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (In)	CanIfTxSduId	L-SDU handle to be transmitted. This handle specifies the corresponding CAN LSDU ID and implicitly the CAN Driver instance as well as the corresponding CAN controller device.
	CanId	Standard/Extended CAN ID of CAN L-SDU that shall be transmitted as FD or conventional CAN frame.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	

Description	This service reconfigures the corresponding CAN identifier of the requested CAN L-PDU. This function is used by user. But it needs to configuration. (It cannot be called directly by user)
Preconditions	CAN Interface module should be initialized
Configuration Dependency	At least one TxPdu should be configured and CANIF_SETDYNAMICTXID_API should be configured as STD_ON.
Available via	CanIf.h

Function Name	CanIf_SetTrcvMode	
Syntax	FUNC(Std_ReturnType, CANIF_CODE) CanIf_SetTrcvMode (VAR(uint8, CANIF_VAR) TransceiverId, VAR(CanTrcv_TrcvModeType, CANIF_VAR) TransceiverMode)	
Service ID [Hex]	0x0d	
Sync/Async	Asynchronous	
Reentrancy	Non-Reentrant	
Parameters (In)	TransceiverId	Abstracted CanIf TransceiverId, which is assigned to a CAN transceiver, which is requested for mode transition.
	TransceiverMode	Requested mode transition.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: Transceiver mode request has been accepted. E_NOT_OK: Transceiver mode request has not been accepted.
Description	This service changes the operation mode of the tansceiver TransceiverId, via calling the corresponding CAN Transceiver Driver service. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one Transceiver should be configured	
Available via	CanIf.h	

Function Name	CanIf_GetTrcvMode
Syntax	FUNC(Std_ReturnType, CANIF_CODE) CanIf_GetTrcvMode(P2VAR(CanTrcv_TrcvModeType, AUTOMATIC, CANIF_APPL_DATA) TransceiverModePtr, VAR(uint8, CANIF_VAR) TransceiverId)
Service ID [Hex]	0x0e
Sync/Async	Synchronous
Reentrancy	Non-Reentrant

Parameters (In)	TransceiverId	Abstracted CanIf TransceiverId, which is assigned to a CAN transceiver, which is requested for current operation mode.
Parameters (Inout)	None	
Parameters (Out)	TransceiverModePtr	Requested mode of requested network the Transceiver is connected to.
Return Value	Std_ReturnType	E_OK: Transceiver mode request has been accepted. E_NOT_OK: Transceiver mode request has not been accepted.
Description	This function invokes CanTrcv_GetOpMode and updates the parameter TransceiverModePtr with the value OpMode provided by CanTrcv. This function is used by user. But it needs configuration. (It cannot be called directly by user)	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one Transceiver should be configured	
Available via	CanIf.h	

Function Name	CanIf_GetTrcvWakeupReason	
Syntax	FUNC(Std_ReturnType, CANIF_CODE) CanIf_GetTrcvWakeupReason (VAR(uint8, CANIF_VAR) TransceiverId, P2VAR(CanTrcv_TrcvWakeupReasonType, AUTOMATIC, CANIF_VAR)TrcvWuReasonPtr)	
Service ID [Hex]	0x0f	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (In)	TransceiverId	Abstracted CanIf TransceiverId, which is assigned to a CAN transceiver, which is requested for wake up reason.
Parameters (Inout)	None	
Parameters (Out)	TrcvWuReasonPtr	Provided pointer to where the requested transceiver wake up reason shall be returned.
Return Value	Std_ReturnType	E_OK: Transceiver wake up reason request has been accepted. E_NOT_OK: Transceiver wake up reason request has not been accepted.
Description	This service returns the reason for the wake up of the transceiver TransceiverId, via calling the corresponding CAN Transceiver Driver service.	

	This function is used by user. But it needs configuration. (It cannot be called directly by user)
Preconditions	CAN Interface module should be initialized
Configuration Dependency	At least one Transceiver should be configured
Available via	CanIf.h

Function Name	CanIf_SetTrcvWakeupMode	
Syntax	FUNC(Std_ReturnType, CANIF_CODE) CanIf_SetTrcvWakeupMode (VAR(uint8, CANIF_VAR) TransceiverId, VAR(CanTrcv_TrcvWakeupModeType, CANIF_VAR) TrcvWakeupMode)	
Service ID [Hex]	0x10	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (In)	TransceiverId	Abstracted CanIf TransceiverId, which is assigned to a CAN transceiver, which is requested for wake up notification mode transition.
	TrcvWakeupMode	Requested transceiver wake up notification mode.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: Will be returned, if the wake up notifications state has been changed to the requested mode. E_NOT_OK: Will be returned, if the wake up notifications state change has failed or the parameter is out of the allowed range. The previous state has not been changed.
Description	This function shall call CanTrcv_SetTrcvWakeupMode. This function is used by user. But it needs configuration. (It cannot be called directly by user)	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one Transceiver should be configured	
Available via	CanIf.h	

Function Name	CanIf_CheckWakeup
Syntax	FUNC(Std_ReturnType, CANIF_CODE) CanIf_CheckWakeup (VAR(EcuM_WakeupSourceType, CANIF_VAR) WakeupSource)
Service ID [Hex]	0x11
Sync/Async	Asynchronous

Reentrancy	Reentrant	
Parameters (In)	WakeupSource	Source device, which initiated the wake up event: CAN controller or CAN transceiver
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: Will be returned, if the check wake up request has been accepted E_NOT_OK: Will be returned, if the check wake up request has not been accepted
Description	This service checks, whether an underlying CAN driver or a CAN transceiver driver already signals a wakeup event. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one Transceiver or Controller should be configured	
Available via	CanIf.h	

Function Name	CanIf_CheckValidation	
Syntax	FUNC(Std_ReturnType, CANIF_CODE) CanIf_CheckValidation (VAR(EcuM_WakeupSourceType, CANIF_VAR) WakeupSource)	
Service ID [Hex]	0x12	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	WakeupSource	Source device which initiated the wake-up event and which has to be validated: CAN controller or CAN transceiver.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: Will be returned, if the check validation request has been accepted. E_NOT_OK: Will be returned, if the check validation request has not been accepted.
Description	This service is performed to validate a previous wakeup event. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one Transceiver or Controller should be configured	
Available via	CanIf.h	

Function Name	CanIf_ClearTrcvWufFlag
Syntax	FUNC(Std_ReturnType, CANIF_CODE) CanIf_ClearTrcvWufFlag (VAR(uint8, CANIF_VAR) TransceiverId

	)	
Service ID [Hex]	0x1e	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different CAN transceivers	
Parameters (In)	TransceiverId	Abstract CanIf TransceiverId, which is assigned to the designated CAN transceiver.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: Request has been accepted E_NOT_OK: Request has not been accepted
Description	Requests the CanIf module to clear the WUF flag of the designated CAN transceiver. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one Transceiver should be configured	
Available via	CanIf.h	

Function Name	CanIf_CheckTrcvWakeFlag	
Syntax	FUNC(Std_ReturnType, CANIF_CODE) CanIf_CheckTrcvWakeFlag (VAR(uint8, CANIF_VAR) TransceiverId)	
Service ID [Hex]	0x1f	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different CAN transceivers	
Parameters (In)	TransceiverId	Abstract CanIf TransceiverId, which is assigned to the designated CAN transceiver.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: Request has been accepted E_NOT_OK: Request has not been accepted
Description	Requests the CanIf module to check the Wake flag of the designated CAN transceiver. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one Transceiver should be configured	
Available via	CanIf.h	

Function Name	CanIf_SetBaudrate

Syntax	FUNC(Std_ReturnType, CANIF_CODE) CanIf_SetBaudrate (VAR(uint8, CANIF_VAR) ControllerId, CONST(uint16, CANIF_CONST) Baudrate)	
Service ID [Hex]	0x27	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different ControllerIds. Non reentrant for the same ControllerId.	
Parameters (In)	ControllerId	Abstract CanIf ControllerId which is assigned to a CAN controller, whose baud rate shall be set.
	BaudRateConfigID	references a baud rate configuration by ID (see CanControllerBaudRateConfigID)
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	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. This function is used by user. But it needs configuration. (It cannot be called directly by user)	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one Controller should be configured	
Available via	CanIf.h	

Function Name	CanIf_SetIcomConfiguration	
Syntax	FUNC(Can_ReturnType, CANIF_CODE) CanIf_SetIcomConfiguration (VAR(uint8, CANIF_VAR) ControllerId, VAR(IcomConfigIdType, CANIF_VAR) ConfigurationId)	
Service ID [Hex]	0x25	
Sync/Async	Asynchronous	
Reentrancy	Reentrant only for different controller Ids	
Parameters (In)	ControllerId	Abstracted CanIf Controller Id which is assigned to a CAN controller.
	ConfigurationId	Requested Configuration
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: Request accepted E_NOT_OK: Request denied
Description	This service shall change the Icom Configuration of a CAN controller to the requested one.	

	This function is used by user. But it needs configuration. (It cannot be called directly by user)
Preconditions	CAN Interface module should be initialized
Configuration Dependency	At least one Controller should be configured
Available via	CanIf.h

Function Name	CanIf_EnableBusMirroring	
Syntax	FUNC(Can_ReturnType, CANIF_CODE) CanIf_EnableBusMirroring (VAR(uint8, CANIF_VAR) ControllerId, VAR(boolean, CANIF_VAR) MirroringActive)	
Service ID [Hex]	0x4c	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	ControllerId	Abstracted CanIf Controller Id 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.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: Mirroring mode was changed. E_NOT_OK: Wrong ControllerId, or mirroring globally disabled (see CanIfBusMirroringSupport).
Description	Enables or disables mirroring for a CAN controller. This function is used by user. But it needs configuration. (It cannot be called directly by user)	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	CANIF_BUS_MIRRORING_SUPPORT should be configured as STD_ON	
Available via	CanIf.h	

Function Name	CanIf_CurrentIcomConfiguration
Syntax	FUNC(Can_ReturnType, CANIF_CODE) CanIf_CurrentIcomConfiguration (VAR(uint8, CANIF_VAR) ControllerId, VAR(IcomConfigIdType, CANIF_VAR) ConfigurationId, VAR(IcomSwitch_ErrorType, CANIF_VAR) Error)
Service ID [Hex]	0x26
Sync/Async	Synchronous
Reentrancy	Reentrant only for different controller Ids

Parameters (In)	ControllerId	Abstract CanIf ControllerId which is assigned to a CAN controller, which informs about the Configuration Id.
	ConfigurationId	Active Configuration Id.
	Error	ICOM_SWITCH_E_OK: No Error ICOM_SWITCH_E_FAILED: Switch to requested Configuration failed. Severe Error.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	
Description	This service shall inform about the change of the Icom Configuration of a CAN controller using the abstract CanIf ControllerId. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	CANIF_PUBLIC_ICOM_SUPPORT should be configured as STD_ON	
Available via	CanIf.h	

Function Name	CanIf_ControllerBusOff	
Syntax	FUNC(void, CANIF_CODE) CanIf_ControllerBusOff (VAR(uint8, CANIF_VAR) ControllerId)	
Service ID [Hex]	0x16	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	ControllerId	Abstract CanIf ControllerId which is assigned to a CAN controller, where a BusOff occured.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	
Description	This service indicates a Controller BusOff event referring to the corresponding CAN Controller with the abstract CanIf ControllerId. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	None	
Available via	CanIf.h	

Function Name	CanIf_ConfirmPnAvailability
Syntax	FUNC(void, CANIF_CODE) CanIf_ConfirmPnAvailability (VAR(uint8, CANIF_VAR) TransceiverId

	)	
Service ID [Hex]	0x1a	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	TransceiverId	Abstract CanIf ControllerId which is assigned to a CAN controller, where a BusOff occured.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	
Description	This service indicates that the transceiver is running in PN communication mode referring to the corresponding CAN transceiver with the abstract CanIf TransceiverId. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	None	
Available via	CanIf_CanTrcv.h	

Function Name	CanIf_ClearTrcvWufFlagIndication	
Syntax	FUNC(void, CANIF_CODE) CanIf_ClearTrcvWufFlagIndication (VAR(uint8, CANIF_VAR) TransceiverId)	
Service ID [Hex]	0x20	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	TransceiverId	Abstract CanIf TransceiverId, which is assigned to a CAN transceiver, for which this function was called.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	
Description	This service indicates that the transceiver has cleared the WufFlag referring to the corresponding CAN transceiver with the abstract CanIf TransceiverId. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one Tranceiver should be configured.	
Available via	CanIf_CanTrcv.h	

Function Name	CanIf_CheckTrcvWakeFlagIndication
Syntax	FUNC(void, CANIF_CODE) CanIf_CheckTrcvWakeFlagIndication (VAR(uint8, CANIF_VAR) TransceiverId

	)	
Service ID [Hex]	0x21	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	TransceiverId	Abstract CanIf TransceiverId, which is assigned to a CAN transceiver, for which this function was called.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	
Description	This service indicates that the check of the transceiver's wake-up flag has been finished by the corresponding CAN transceiver with the abstract CanIf TransceiverId. This indication is used to cope with the asynchronous transceiver communication. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one Transceiver should be configured.	
Available via	CanIf_CanTrcv.h	

Function Name	CanIf_ControllerModeIndication	
Syntax	FUNC(void, CANIF_CODE) CanIf_ControllerModeIndication (VAR(uint8, CANIF_VAR) ControllerId, VAR(Can_ControllerStateType, CANIF_VAR) ControllerMode)	
Service ID [Hex]	0x17	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	ControllerId	Abstract CanIf ControllerId which is assigned to a CAN controller, which state has been transitioned.
	ControllerMode	Mode to which the CAN controller transitioned
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	
Description	This service indicates a controller state transition referring to the corresponding CAN controller with the abstract CanIf ControllerId. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one Controller should be configured.	
Available via	CanIf_Can.h	

Function Name	CanIf_TrcvModeIndication	
Syntax	FUNC(void, CANIF_CODE) CanIf_TrcvModeIndication (VAR(uint8, CANIF_VAR) TransceiverId, VAR(CanTrcv_TrcvModeType, CANIF_VAR) TransceiverMode)	
Service ID [Hex]	0x22	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	TransceiverId	Abstract CanIf TransceiverId, which is assigned to a CAN transceiver, which state has been transitioned.
	TransceiverMode	Mode to which the CAN transceiver transitioned
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	
Description	This service indicates a transceiver state transition referring to the corresponding CAN transceiver with the abstract CanIf TransceiverId. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	At least one Transceiver should be configured.	
Available via	CanIf_CanTrcv.h	

6.3.5 Error Handling

Function Name	CanIf_GetControllerErrorState	
Syntax	FUNC(Std_ReturnType, CANIF_CODE)CanIf_GetControllerErrorState (VAR(uint8, CANIF_VAR) ControllerId, P2VAR(Can_ErrorStateType, AUTOMATIC, CANIF_APPL_DATA) ErrorStatePtr)	
Service ID [Hex]	0x4b	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same ControllerId	
Parameters (In)	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller, which is requested for ErrorState.
Parameters (Inout)	ErrorStatePtr	Pointer to a memory location, where the error state of the CAN controller will be stored.
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: Error state request has been accepted. E_NOT_OK: Error state request has not been accepted.

Description	This service calls the corresponding CAN Driver service for obtaining the error state of the CAN controller. This function is used by user. But it needs configuration. (It cannot be called directly by user)
Preconditions	CAN Interface module should be initialized
Configuration Dependency	None
Available via	CanIf.h

Function Name	CanIf_GetControllerRxErrorCounter	
Syntax	FUNC(Can_ReturnType, CANIF_CODE) CanIf_GetControllerRxErrorCounter (VAR(uint8, CANIF_VAR) ControllerId, P2VAR(uint8, AUTOMATIC, CANIF_VAR) RxErrorCounterPtr)	
Service ID [Hex]	0x4d	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same ControllerId	
Parameters (In)	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller.
Parameters (Inout)	None	
Parameters (Out)	RxErrorCounterPtr	Pointer to a memory location, where the current Rx error counter of the CAN controller will be stored.
Return Value	Std_ReturnType	E_OK: Error state request has been accepted. E_NOT_OK: Error state request has not been accepted.
Description	This service calls the corresponding CAN Driver service for obtaining the Rx error counter of the CAN controller. This function is used by user. But it needs configuration. (It cannot be called directly by user)	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	None	
Available via	CanIf.h	

Function Name	CanIf_GetControllerTxErrorCounter
Syntax	FUNC(Can_ReturnType, CANIF_CODE) CanIf_GetControllerTxErrorCounter (VAR(uint8, CANIF_VAR) ControllerId, P2VAR(uint8, AUTOMATIC, CANIF_VAR) TxErrorCounterPtr)
Service ID [Hex]	0x4e
Sync/Async	Synchronous
Reentrancy	Non Reentrant for the same ControllerId

Parameters (In)	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller.
Parameters (Inout)	None	
Parameters (Out)	TxErrorCounterPtr	Pointer to a memory location, where the current Tx error counter of the CAN controller will be stored.
Return Value	Std_ReturnType	E_OK: Tx error counter available. E_NOT_OK: Wrong ControllerId, or Tx error counter not available.
Description	This service calls the corresponding CAN Driver service for obtaining the Tx error counter of the CAN controller. This function is used by BSW.	
Preconditions	CAN Interface module should be initialized	
Configuration Dependency	None	
Available via	CanIf.h	

7 Generator

7.1 Generator Message

Options	Description
-G,--Generation	Symbolic parameters to be used for fore generation (skip validation).
-H,--Help	Display this help message.
-I,--Input <I>	ECU description file path of the module for which generation tool need to run.
-L,--Log	Symbolic parameters to be used for generation error log.
-M,--Module <M>	Specify module name and version to be generated code for.
-O,--Output <O>	Project-relative path to location where the generated code is to be placed.
-T,--Top_path <T>	Symbolic parameters to be used for set path of module.
-V,--Validate	Symbolic parameters to be used for invoking validation checks.

7.1.1 Error Messages

ERR0600001 The reference path is empty for the parameter <Parameter name> in the container <Container name>, having short name <short name>.

This error message is displayed if the value of following parameters are empty

Parameter name	Container name
CanIfCtrlDrvInitHohConfigRef	CanIfCtrlDrvCfg
CanIfCtrlDrvNameRef	CanIfCtrlDrvCfg
CanIfCtrlCanCtrlRef	CanIfCtrlCfg
CanIfBufferHthRef	CanIfBufferCfg
CanIfHrhCanCtrlIdRef	CanIfHrhCfg
CanIfHrhIdSymRef	CanIfHrhCfg
CanIfHthCanCtrlIdRef	CanIfHrhCfg
CanIfHthIdSymRef	CanIfHrhCfg
CanIfRxPduHrhIdRef	CanIfRxPduCfg
CanIfTTRxHwObjectTriggerIdRef	CanIfRxPduCfg
CanIfTxPduBufferRef	CanIfTxPduCfg
CanIfTxPduRef	CanIfTxPduCfg
CanIfTTTxHwObjectTriggerIdRef	CanIfTxPduCfg
CanIfTTDemEventParameterRefs	CanIfTTGeneral
CanIfTrcvCanTrcvRef	CanIfTrcvCfg

ERR0600002 The parameter <parameter name> in the container <container name> should be configured.

This error message is displayed if the following parameters are not configured.

Parameter name	Container name
AR-RELEASE-VERSION	BSW-IMPLEMENTATION
SW-VERSION	BSW-IMPLEMENTATION
VENDOR-ID	BSW-IMPLEMENTATION

ERR0600003 The value configured for the parameter <parameter> in the container <container> should follow the pattern.

This error message is displayed if (AR-RELEASE-VERSION in BSW-IMPLEMENTATION does not follow the pattern: <4.[0-9]+.[0-9]+>)

ERR0600005 At least one HRH or HTH should be configured within the controller <controller ID> of the CAN Driver <Driver short name>.
This Error message is displayed if (Controller is not configured with HRH or HTH)
Path of parameters related to this error:
+ CanIf/CanIfInitCfg/CanIfInitHohCfg/CanIfHrhCfg/CanIfHrhCanCtrlIdRef
+ CanIf/CanIfInitCfg/CanIfInitHohCfg/CanIfHthCfg/CanIfHthCanCtrlIdRef
+ CanIf/CanIfCtrlDrvCfg

ERR0600006 IDTABLE of CanIfPrivateSoftwareFilterType should be one HRH per CAN Controller

This error message is displayed if CanIfPrivateSoftwareFilterType is set to IDTABLE but number of HRH and Can Controller not be the same.

ERR0600007 Value 'Parameter' of the reference parameter 'Parameter Name' is repeated within the container 'Container Name' in the configuration set <Configset Name>.

This Error message is displayed if Parameters is repeated
+/CanIf/CanIfInitCfg/CanIfRxPduCfg/CanIfRxPduCanId is repeated and CanIfRxPduCanIds is belong to one CanIfCtrlCfg
Path:
CanIf/CanIfInitCfg/CanIfRxPduCfg/CanIfRxPduHrhIdRef -> CanIf/CanIfInitCfg/CanIfInitHohCfg/CanIfHrhCfg/CanIfHrhCanCtrlIdRef

+/CanIf/CanIfInitCfg/CanIfTxPduCfg/CanIfTxPduCanId is repeated and CanIfTxPduCanIds is belong to one CanIfCtrlCfg
Path:
CanIf/CanIfInitCfg/CanIfTxPduCfg/CanIfTxPduBufferRef -> CanIf/CanIfInitCfg/CanIfBufferCfg/CanIfBufferHthRef -> CanIf/CanIfInitCfg/CanIfInitHohCfg/CanIfHthCfg/CanIfHthCanCtrlIdRef

Path of parameters related to this error:
+/CanIf/CanIfTrcvDrvCfg/CanIfTrcvCfg/CanIfTrcvCanTrcvRef
+/CanIf/CanIfInitCfg/CanIfInitHohCfg/CanIfHthCfg/CanIfHthIdSymRef
+ /CanIf/CanIfInitCfg/CanIfInitHohCfg/CanIfHrhCfg/CanIfHrhIdSymRef
+/CanIf/CanIfCtrlDrvCfg/CanIfCtrlCfg/CanIfCtrlId
+/CanIf/CanIfTrcvDrvCfg/CanIfTrcvCfg/CanIfTrcvId
+/CanIf/CanIfCtrlDrvCfg/CanIfCtrlCfg/CanIfCtrlCanCtrlRef

ERR0600008 Rx Pdu Range configured in the container <container name 1> should be within the HRH Range configured in container <container name 2>

This Error message is displayed if {
(CanIfRxPduCanIdRangeLowerCanId < CanIfHRHRangeRxPduLowerCanId ||
CanIfRxPduCanIdRangeLowerCanId > CanIfHRHRangeRxPduUpperCanId) ||
(CanIfRxPduCanIdRangeUpperCanId < CanIfHRHRangeRxPduLowerCanId ||
CanIfRxPduCanIdRangeUpperCanId > CanIfHRHRangeRxPduUpperCanId)
Path of parameters related to this error:
+/CanIf/CanIfInitCfg/CanIfInitHohCfg/CanIfHrhCfg/CanIfHrhRangeCfg/CanIfHrhRangeRxPduLowerCanId
+/CanIf/CanIfInitCfg/CanIfRxPduCfg/CanIfRxPduCanIdRange/CanIfRxPduCanIdRangeLowerCanId
+/CanIf/CanIfInitCfg/CanIfInitHohCfg/CanIfHrhCfg/CanIfHrhRangeCfg/CanIfHrhRangeRxPduUpperCanId
+/CanIf/CanIfInitCfg/CanIfRxPduCfg/CanIfRxPduCanIdRange/CanIfRxPduCanIdRangeUpperCanId

ERR0600009 At least one Tx PDU or Rx PDU should be configured.

This Error message is displayed, if (At least one Tx PDU or Rx PDU is not configured)

ERR0600010 Parameter <parameter> should be <true/1>, since Dynamic Tx PDUs are configured.

This error message displayed when Dynamic Tx PDUs are configured but CanIfPublicSetDynamicTxIdApi disabled.
Path:
+ /CanIf/CanIfPublicCfg/CanIfPublicSetDynamicTxIdApi
+ /CanIf/CanIfInitCfg/CanIfTxPduCfg/CanIfTxPduType

ERR0600011 Lower HRH Range configured in the parameter 'parameter name' should be less than upper HRH Range configured in the parameter 'parameter name' of container 'container name'.

This Error message is displayed, if (Lower Rx Pdu Range configured in the parameter 'CanIfRxPduCanIdRangeLowerCanId' greater than upper Rx pdu Range configured in the parameter 'CanIfRxPduCanIdRangeUpperCanId' of container ' CanIfRxPduCanIdRange') and
if (Lower HRH Range configured in the parameter 'CanIfHRHRangeRxPduLowerCanId' greater than upper HRH Range configured in the parameter 'CanIfHRHRangeRxPduUpperCanId' of container ' CanIfRxPduCanIdRange')
Path of parameters related to this error:
+/CanIf/CanIfInitCfg/CanIfRxPduCfg/CanIfRxPduCanIdRange/CanIfRxPduCanIdRangeLowerCanId
/CanIf/CanIfInitCfg/CanIfRxPduCfg/CanIfRxPduCanIdRange/CanIfRxPduCanIdRangeUpperCanId
+/CanIf/CanIfInitCfg/CanIfInitHohCfg/CanIfHrhCfg/CanIfHrhRangeCfg/CanIfHrhRangeRxPduLowerCanId
/CanIf/CanIfInitCfg/CanIfInitHohCfg/CanIfHrhCfg/CanIfHrhRangeCfg/CanIfHrhRangeRxPduUpperCanId

ERR0600012 Value(s) <list of Id> is (are) not configured for the parameter <Parameter Name> in the container <Container Name>.

This Error message is displayed if value of the parameter 'CanIfCtrlId/ CanIfTrcvId/ CanIfRxPduId/ CanIfTxPduId' does not start from 0 and is not sequential

ERR0600013 Valid values of the parameter 'Parameter Name' of the container 'Container Name' <PDU name> are <0 – 2047/536870911> for <STANDARD/EXTENDED> type.

This Error message is displayed, if((CanIfRxPduCanId < 0 or > 2047, when CanIfRxPduCanIdType == "STANDARD")OR (CanIfRxPduCanId < 0 or > 2415919103, when CanIfRxPduCanIdType == "EXTENDED")OR (CanIfTxPduCanId < 0 or > 2047, when CanIfTxPduCanIdType == "STANDARD")OR (CanIfTxPduCanId is < 0 or > 2415919103, when CanIfTxPduCanIdType == "EXTENDED"))

ERR0600014 Number of HTHs configured should be within the range of <UINT8>/<UINT16>. Number of container 'Container Name' configured = <Container count>. Value of the parameter 'Parameter Name' = <Value>.

This Error message is displayed if (parameter 'CanIfPublicHandleTypeEnum' configured as UINT8 & number of HTHs configured are more than 255 Or if parameter 'CanIfPublicHandleTypeEnum' configured as UINT16 & number of HTHs configured are more than 65535)

ERR0600015 Value of the parameter <CanIfRxPduUserRxIndicationName> should be conifgured when value of the parameter <CanIfRxPduUserRxIndicationUL> is configured as <CDD> in the container <CanIfRxPduCfg>.

This Error message is displayed if (parameter 'CanIfRxPduUserRxIndicationName' not configured when 'CanIfRxPduUserRxIndicationUL' is configured as <CDD> in the container 'CanIfRxPduCfg').

ERR0600016 Value of the parameter 'CanIfRxPduCanId' must be configured since 'CanHandleType' configured as 'FULL' in CAN driver.

This Error message is displayed if (Value of the parameter 'CanIfRxPduCanId' not configured when CanHandleType' configured as 'FULL' in CAN driver).

ERR0600017 HRH Range should be configured as range for CanIf Rx Pdu is configured.
This Error message is displayed if (HRH Range is not configured when range for CanIfRxPduCfg is configured).

ERR0600018 Parameter 'CanIfPublicWakeupCheckValidSupport' and 'CanIfPublicWakeupCheckValidByNM' of container 'CanIfPublicCfg' should be configured <false/0> since, 'CanIfCtrlWakeupSupport/CanIfTrcvWakeupSupport' of 'CanIfCtrlCfg/CanIfTrcvCfg' is equal to <0>

This Error is displayed, if parameter CanIfPublicWakeupCheckValidSupport or CanIfPublicWakeupCheckValidByNM set to True, but CanIfCtrlWakeupSupport value is False.

ERR0600019 Configuration on 'CanIfDispatch*Name' is Empty**

This message is displayed when check condition of macro CANIF_DISPATCH_***_NAME is 'NULL' when CANIF_DISPATCH_***_UL is 'CDD'

ERR0600022 CAN Transceiver Wakeup supported feature should be STD_ON within the CanTrcvWakeUpSupport of the CanTrcv. CanTrcvWakeUpSupport should be CANTRCV_WKEUP_BY_POLLING.

This error is displayed when CANIF_TRCV_WAKEUP_SUPPORT is set to STD_ON && CanTrcvWakeUpSupport value not equal to CANTRCV_WAKEUP_BY_POLLING.

ERR0600024 < RxPduCfg name >CanIfRxPduCanIdType must not be STANDARD_NO_FD_CAN or EXTENDED_NO_FD_CAN since CanIfRxPduDataLength greater than 8.

This error is displayed when CanIfRxPduDataLength configured greater than 8 but CanIfRxPduCanIdType is STANDARD_NO_FD_CAN/EXTENDED_NO_FD_CAN.

ERR0600023 Value of the parameter 'CanIfTxPduUserTriggerTransmitName' should be conifgured when value of the parameter 'CanIfTxPduUserTxConfirmationUL' is configured as <CDD> in the container 'CanIfTxPduCfg'.

This Error message is displayed if parameter 'CanIfTxPduUserTriggerTransmitName' not configured when 'CanIfTxPduUserTxConfirmationUL' is configured as <CDD> in the container 'CanIfTxPduCfg'.

ERR0600026 Value of the parameter 'CanIfTxPduUserTxConfirmationName' should be conifgured when value of the parameter 'CanIfTxPduUserTxConfirmationUL' is configured as <CDD> in the container 'CanIfTxPduCfg'.

This Error message is displayed if parameter 'CanIfTxPduUserTxConfirmationName' not configured when 'CanIfTxPduUserTxConfirmationUL' is configured as <CDD> in the container 'CanIfTxPduCfg'.

ERR0600025 Just configurable only one Container 'CanIfInitCfg' when configured 'VARIANT-PRE-COMPILE'

This Error message is displayed if parameter 'Implementation Config Variant' equals 'VARIANT-PRE-COMPILE' but number of variant configured greater than 1 (number Container 'CanIfInitCfg' > 1).

ERR0600027 Mismatch variant between EcuC and CanIf.

This Error message is displayed if parameter 'Implementation Configure Variant' equals 'VARIANT-POST-BUILD' /'VARIANT-POST-BUILD-SELECTABLE' but Variants in EcuC(EcuC/EcucPostBuildVariants/EcucPostBuildVariantRef) not same with Variants in CanIf(number of Container CanIf/CanIfInitCfg).

ERR0600030: Mismatch configure TxConfirm between CanIf(ON) & CanNm(OFF).

This error message is displayed when CanIf have configure Tx Confirmation to CanNm, but in CanNm is turn off Tx confirmation. If exists CanIfRxPduUserRxIndicationUL = "CAN_NM" in List CanIfRxPduCfg and if ((canNmImmediateTxconfEnabled == true) || (canNmPassiveModeEnabled == true)

ERR0600031: If CanIfPublicPnSupport equals TRUE, <'CanIfDispatchUserCheckTrcvWakeFlagIndicationUL'/'CanIfDispatchUserClearTrcvWufFlagIndicationUL'/'CanIfDispatchUserConfirmPnAvailabilityUL'> shall be configurable.

This Error message is displayed if parameter CanIfPublicPnSupport equals TRUE but **CanIfDispatch...UL** is not configure
if CanIfPublicPnSupport = true
and (CanIfDispatchUserCheckTrcvWakeFlagIndicationUL = null || CanIfDispatchUserClearTrcvWufFlagIndicationUL || CanIfDispatchUserConfirmPnAvailabilityUL)

ERR0600032: Both 'Rx Pdu Can Id' and 'Rx Pdu Id' of Rx Pdu included in Basic must have an ascending order if Private Software Filter Type is 'BINARY'. Please change below orders or Private Software Filter type can be 'LINEAR'. RxPdu : < ~ >, Rx Pdu Hrh Id Ref : < ~ >, Rx Pdu Can Id : < ~ >, Rx Pdu Id : < ~ > should be changed

This Error message is displayed if value of parameter “CanIfRxPduCanId” and “CanIfRxPduId” sequence is ascending or not, when “CanIfPrivateSoftwareFilterType” is “BINARY”. This error can be fixed by changing the Rx Pdu Id, Rx Pdu Can Id to ascending order or changing “CanIfPrivateSoftwareFilterType” to “LINEAR”.

ERR0600033: The reference path of parameter 'CanTrcvWakeupSourceRef/CanWakeupSourceRef' in container 'CanTrcvChannel/CanCtrl' should not be empty, when parameter ' CanIfTrcvWakeupSupport /CanIfCtrlWakeupSupport' in container ‘CanIfTrcvCfg/CanIfCtrlCfg' is set ON

This Error message is displayed if 'CanIfWakeupSupport/CanIfPublicWakeupCheckValidSupport’ is set ON and parameter 'CanIfCtrlWakeupSupport/CanIfTrcvWakeupSupport' in container 'CanIfCtrlCfg/CanIfTrcvCfg' is set ON. But The reference path of parameter 'CanTrcvWakeupSourceRef' in container 'CanTrcvChannel' is not configured or the reference path of parameter 'CanWakeupSourceRef' in container 'CanCtrl' is not configured.

ERR0600035: Missing 'EcuM/CanTrcv/Can' file in the input files

This Error message is displayed if

- Parameter 'CanIfWakeupSupport/CanIfPublicWakeupCheckValidSupport’ in container 'CanIfPublicCfg' is set ON. But missing EcuM file in the input files
- Parameter 'CanIfCtrlWakeupSupport' in container 'CanIfCtrlCfg' is set ON. But missing Can file in the input files
- Parameter 'CanIfTrcvWakeupSupport' in container 'CanIfTrcvCfg' is set ON. But missing CanTrcv file in the input files

ERR0600036: HRH <ShortName> connected to CAN Driver <driver name> is not connected to any Rx PDU.

This Error message is displayed when HRH have no connected by any RxPDU.

ERR0600037: HTH <ShortName> connected to CAN Driver <driver name> is not connected to any Tx PDU.

This Error message is displayed when HTH have no connected by any TxPDU.

7.1.2 Warning Message

WRN0600001 Parameter <parameter Name> of the container <Container name> should not be configured as greater than <0>, when Tx Hardware Object referred through the parameter 'CanIfHthCanHandleTypeRef' to Can Driver is configured as <FULL>.

This warning is displayed, if parameter CanIfHthCanHandleTypeRef referenced through Can driver is configured as 'FULL' but parameter CanIfBufferSize of container CanIfBufferCfg configured as greater than 0 or equal 0 when 'BASIC'.

WRN0600002 Parameter <parameter> in the container <container> should be <false/0>, since no HRH is configured.

This warning message displayed, when CanIfPrivateDataLengthCheck set to true but no HRH configured.

WRN0600004 Parameter < CanIfPublicSetDynamicTxIdApi> should be <false/0>, since Dynamic Tx PDUs are not configured. Parameter < CanIfPublicSetDynamicTxIdApi > is set to <false/0>.

This warning is displayed, if parameter CanIfPublicSetDynamicTxIdApi set to True, but Dynamic Tx PDUs are not configured.

WRN0600005: Parameter <parameter name 1> of PDU <PDU name> should be <false/0>, since parameter <parameter name 2> of <container name> is disabled. parameter <parameter name 2> is set to <false/0>.

This warning is displayed if parameter CanIfTxPduReadNotifyStatus of PDU set to True, since CanIfPublicReadTxPduNotifyStatusApi is disabled.

Or parameter CanIfRxPduReadData of PDU set to true, since CanIfPublicReadRxPduDataApi is disabled.

Or CanIfRxPduReadNotifyStatus of PDU set to True, since CanIfPublicReadRxPduNotifyStatusApi is disabled.

WRN0600007 CanIfRxPduCanId configured in the container <name> should be within the Rx Pdu Range configured in container CanIfRxPduCanIdRange.

This warning is displayed when CanIfRxPduCanId not configured in range.

7.1.3 Information Messages

None

8 SWP Error Code

8.1 SWP Error Code List

None

9 Appendix

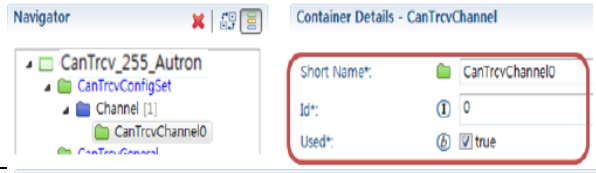
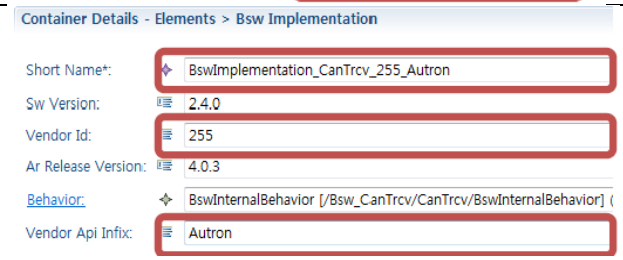
9.1 CanTrcv Module development

Refer to AUTOSAR 4.4.0 CANTransceiverDriver Spec for details on creating CANTRCV module.

When creating a CANTRCV module, file, or API (including internal functions), it must be written in accordance with the recommended Naming Rule.

- VendorID uses 255 (0xFF)
- Naming Rule : CanTrcv_VendorId_VendorSpecifiName

8.1.1 File that need to be created

Ecud_CanTrcv_255_ VendorspecificName.arxm	Module Definition PDF: It is recommended to use the basic PDF provided by AUTOSAR. 	Ecud_CanTrcv_255_Autron.arxml Distribution file
Bswmd_CanTrcv_255_ VendorspecficName.arxml		Bswmd_CanTrcv_255_Autron.arxml Distribution file
CanTrcv_255_ VendorspecificName.h		CanTrcv_255_Autron.h Distribution file
CanTrcv_ VenderID_VendorspecificName.c		CanTrcv_255_Autron.c Distribution file

9.1.2 Required API

1) **When not using CAN Transceiver Wake Up**

CanTrcv_255_VendorspecificName_SetOpMode	Change module mode
CanTrcv_255_VendorspecificName_GetOpMode	Check module mode information
CanTrcv_255_VendorspecificName_Init	Module Initialization

2) **When using CAN Transceiver Wake Up**

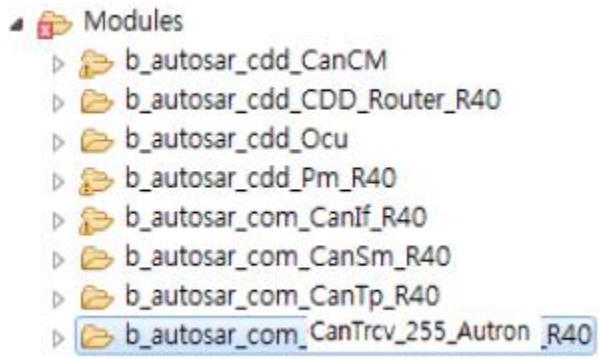
CanTrcv_255_VendorspecificName_SetOpMode	Change module mode
CanTrcv_255_VendorspecificName_GetOpMode	Check module mode information
CanTrcv_255_VendorspecificName_Init	Module Initialization
CanTrcv_255_VendorspecificName_GetBusWuReason	Gets the wakeup reason for the Transceiver and returns it in parameter Reason
CanTrcv_255_VendorspecificName_SetWakeupMode	Enables, disables or clears wakeup events of the Transceiver according to TrcvWakeupMode.
CanTrcv_255_VendorspecificName_CheckWakeup	Service is called by underlying CANIF in case a wake up interrupt is detected
CanTrcv_255_VendorspecificName_ClearTrcvWufFlag	Clears the WUF flag in the transceiver hardware. This API shall exist only if CanTrcvHwPnSupport = TRUE.
CanTrcv_255_VendorspecificName_CheckWakeFlag	Requests to check the status of the wakeup flag from the transceiver hardware.

9.1.3 CANTRCV Integration Method

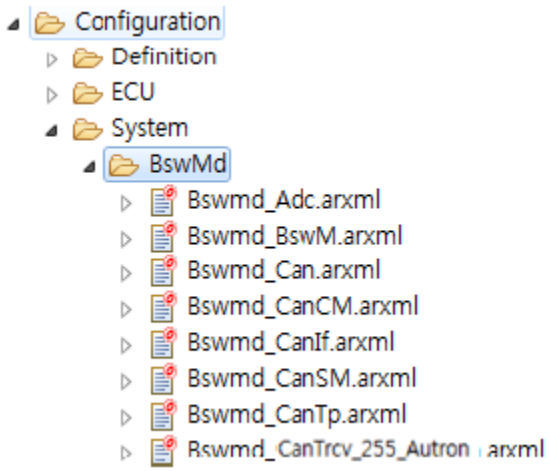
9.1.3.1 Add CANTRCV Module (Source or Library) and Build Configuration

AUTRON CANTRCV Module + New CANTRCV Module

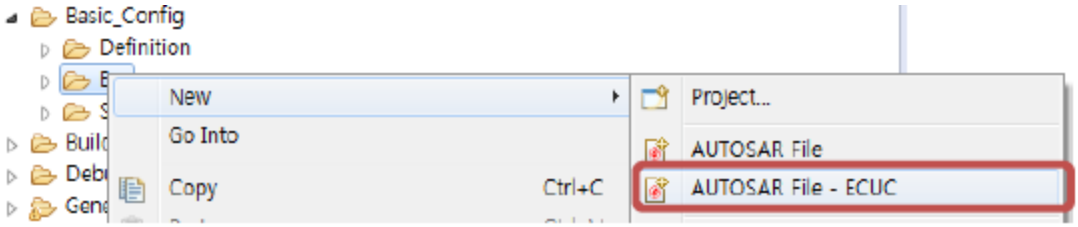
1) Copy the new CANTRCV module (Source or Libraray) to the folder where the Bsw module is located. (The location of the module can be changed.)

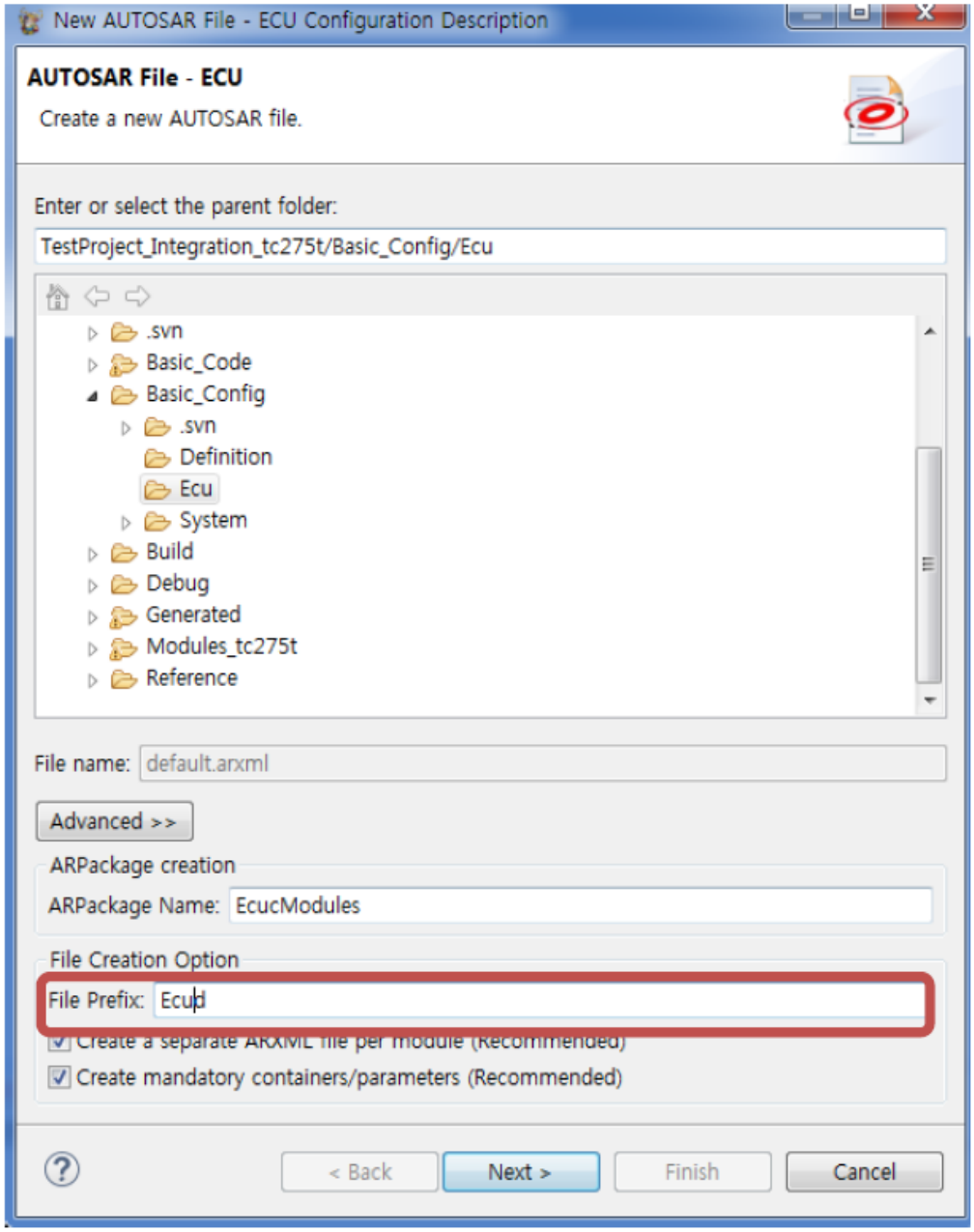


2) Copy the Bswmd file of the new CANTRCV module: ECU> System> BswMd

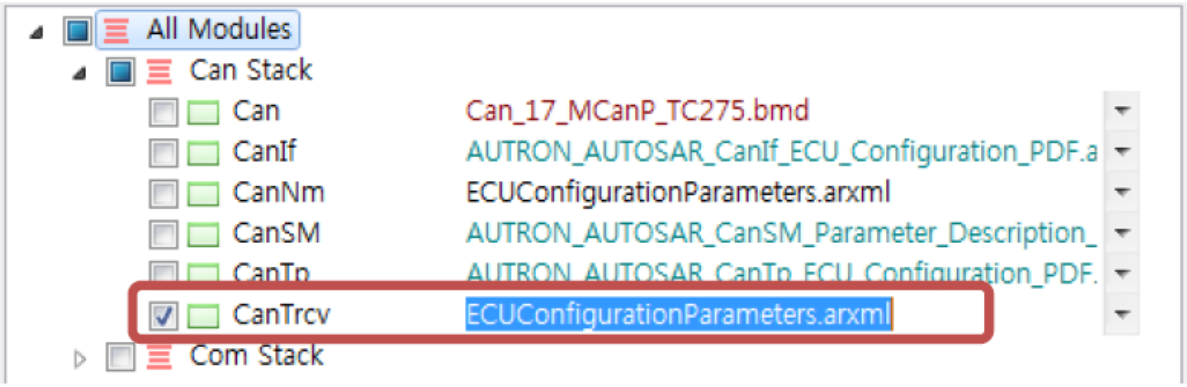


3) Project-AUTOSAR File EcuC Generation> Select CanTrcv in Module Selection> ECUConfigurationParameters.arxml

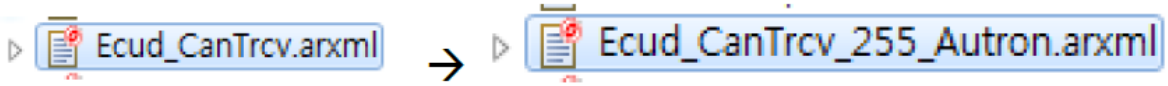




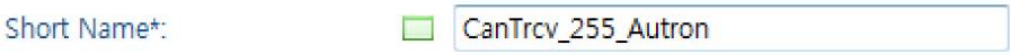
Change File Prefix from Ecucd to Ecud



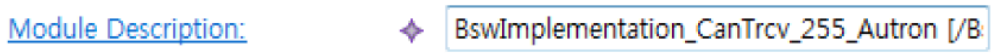
4) Rename> Ecud_CanTrcv_ 255_VendorspecificName



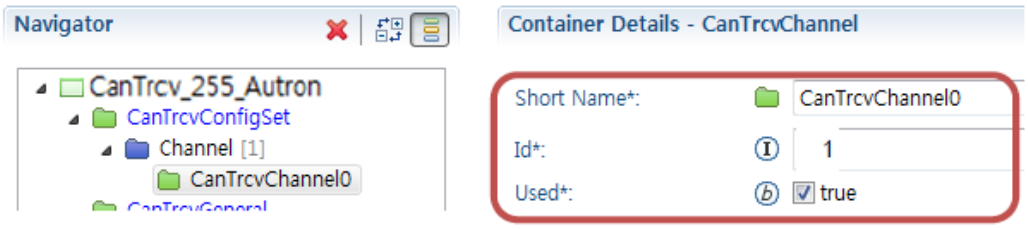
5) Change Module Short Name> CanTrcv_ 255_VendorspecificName



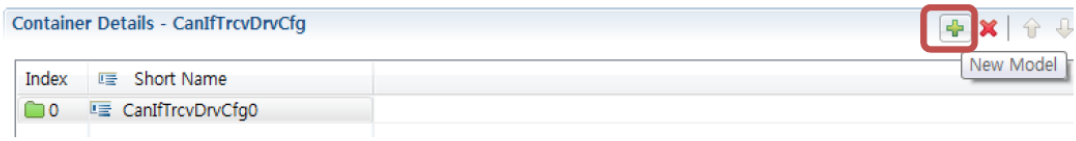
6) Select Module Description

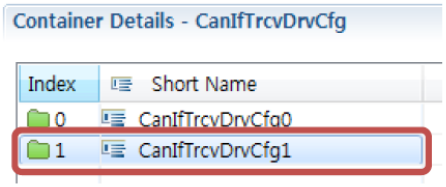


7) CanTrcv> Create a New Channel from Channel
There are three items that must be present in Ecud_CanTrcv_255_Autron.arxml.
- Short Name: The name
- Id: Use the following number CanTrcvChannelId of AUTRON CANTRCV module
- (If CanTrcvChannelId of Ex. AUTRON CANTRCV is 0, Ecud_CanTrcv_255_Autron module is available from No.1)
- Used: True

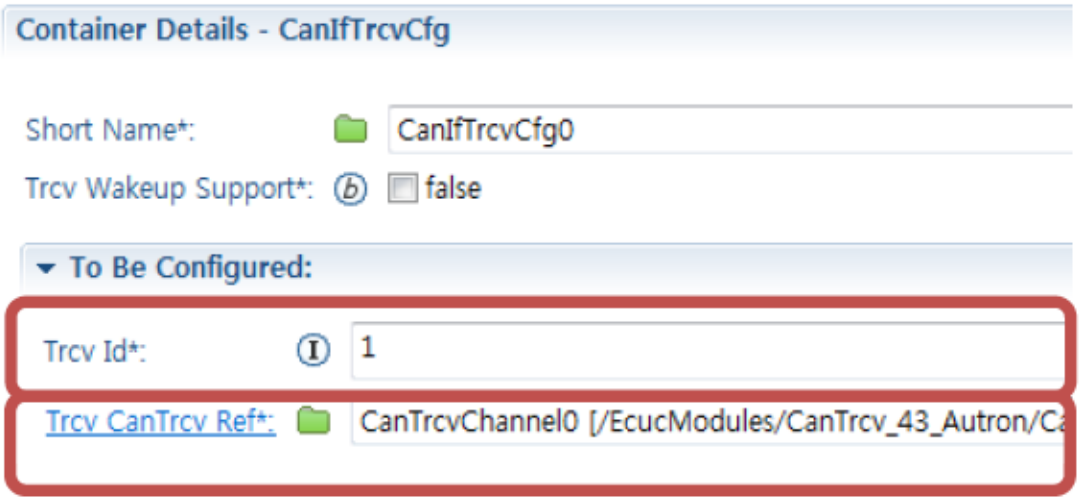


8) In CanIf> Trcv Drv Cfg, click the plus icon to create a new CanIfTrcvDrvCfg

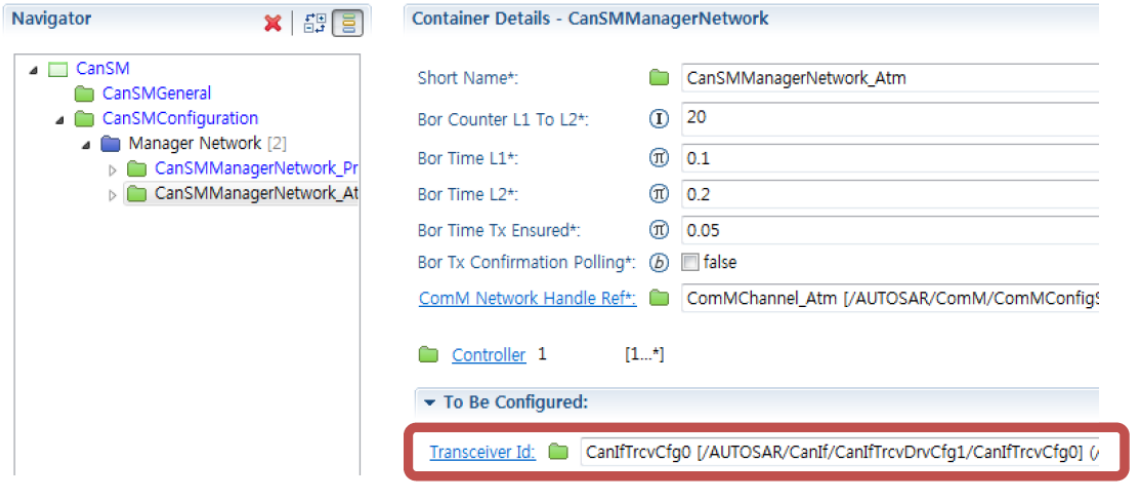




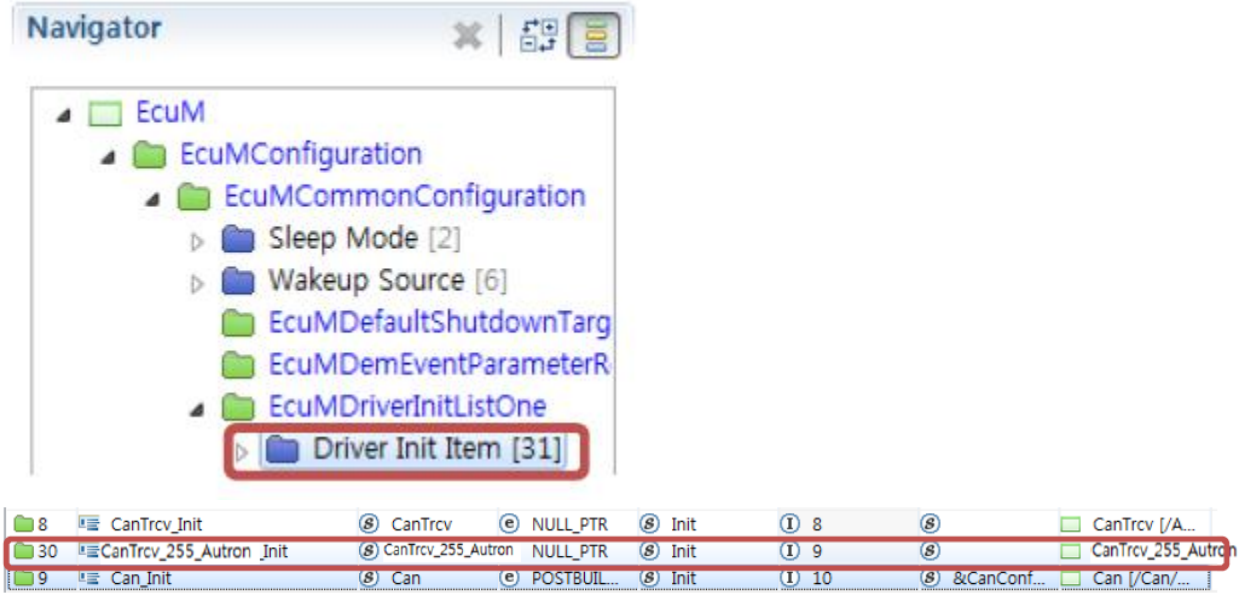
9) CanTrcvChannelId is set to the following number CanTrcvChannelId used by AUTRON CANTRCV (example: 1 Trcv is used for existing AUTRON CANTRCV module)



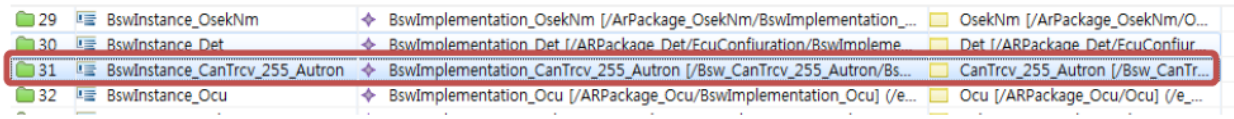
10) CanSM> CanSMConfiguration> Configure a New Trcv



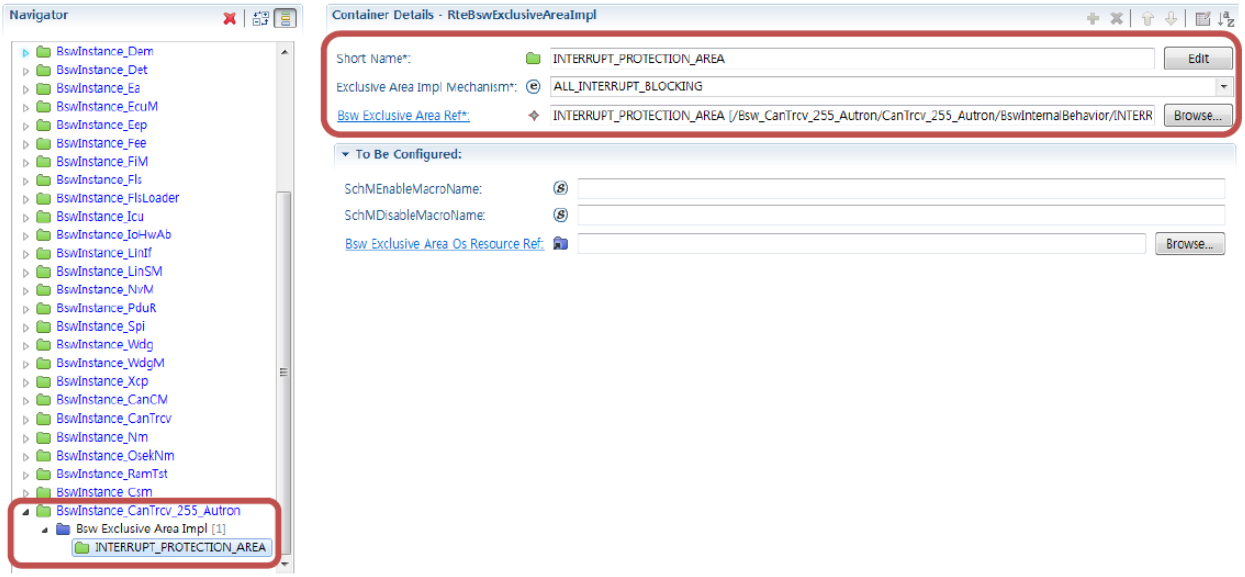
11) EcuM> EcuMConfiguration> EcuMCommonConfiguration> Add Init function of new CanTrcv module to EcuMDriverInitListOne (order is added after existing CanTrcv)



12) Add CanTrcv module to Rte> Bsw Module Instance



13) Add Bsw Exclusive Area to Rte> Bsw Module Instance



14) Modify Generate.py file: Added Ecud_CanTrcv_255_VendorspecificName added to GenerateCanIf, GenerateCanSm, GenerateEcuM, GenerateRte and Bswmd_CanTrcv_255_VendorspecificName if necessary

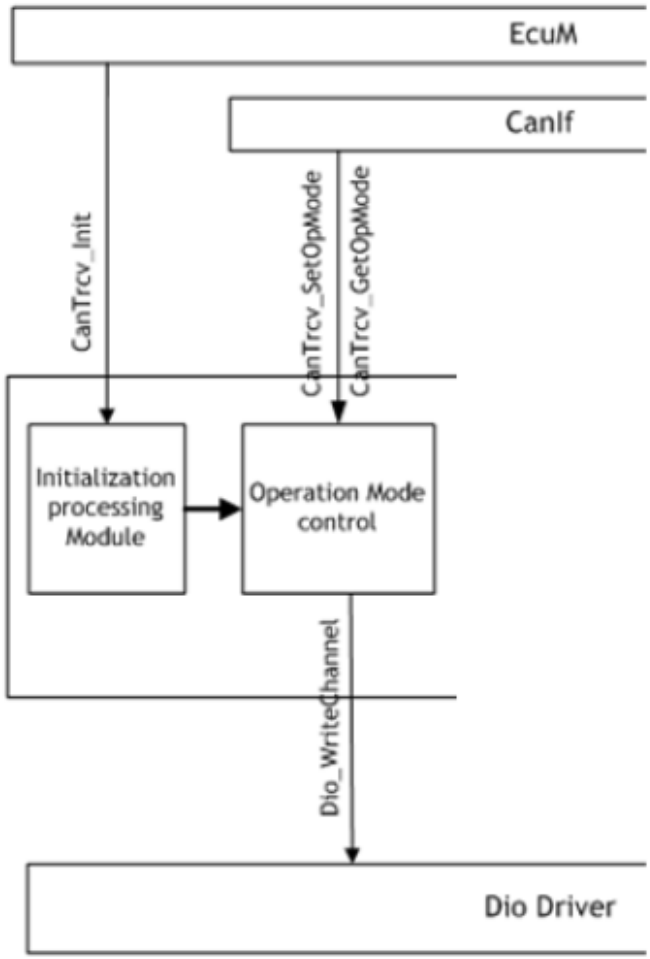
9.1.4 Cautions when setting CANTRCV module

1) CanTrcvChannelId of CANTRCV and CanIfTrcvId of CanIf must be the same.
   (If not, error occurs in CanIf.exe)
2) CanTrcvChannelId of CANTRCV should be set after AUTORN CANTRCV CanTrcvChannelId first (when using AUTRON CANTRCV module) and then CanTrcvChannelId of EXTENAL CANTRCV.
3) AUTRON CANTRCV module does not support Transceiver Wake Up function

9.1.5 Explanation of CANTRCV Module Operation

See the AUTOSAR CANTRCV specification for more information.

1) CANTRCV_TRCVMODE_NORMAL request through CanTrcv_255_VendorspecificName_SetOpMode API call when communication full-communication request
2) CANTRCV_TRCVMODE_STANDBY request by calling CanTrcv_255_VendorspecificName_SetOpMode API when communication No-Communication request. Command gives STANDBY)
3) After mode change is completed, changed mode related Indication should be called by using CanIf_TrcvModeIndication API.



9.1.6 Cautions when selecting CANTRCV H / W

When determining communication wake up in the CANCM module, judge it by looking at Level (Low) of CAN RX. If Wake Up operation does not maintain Level (Low) when selecting CANTRCV H / W, the function cannot be used.

9.2 Imported types

Module	Header File	Imported Type
Can_GeneralTypes	Can_GeneralTypes.h	CanTrcv_TrcvModeType
	Can_GeneralTypes.h	CanTrcv_TrcvWakeupModeType
	Can_GeneralTypes.h	CanTrcv_TrcvWakeupReasonType
	Can_GeneralTypes.h	Can_ControllerStateType
	Can_GeneralTypes.h	Can_ErrorStateType
	Can_GeneralTypes.h	Can_HwHandleType
	Can_GeneralTypes.h	Can_HwType
	Can_GeneralTypes.h	Can_IdType
	Can_GeneralTypes.h	Can_PduType
ComStack_Types	ComStackTypes.h	IcomConfigIdType
	ComStackTypes.h	IcomSwitch_ErrorType
	ComStackTypes.h	PduIdType
	ComStackTypes.h	PduInfoType
EcuM	EcuM.h	EcuM_WakeupSourceType
Std_Types	StandardTypes.h	Std_ReturnType
	StandardTypes.h	Std_VersionInfoType