Document Title	Specification of CAN XL Transceiver Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	1015

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R24-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2024-11-27	R24-11	AUTOSAR Release Management	• Editorial changes • Fix Service IDs
2023-11-23	R23-11	AUTOSAR Release Management	• Editorial changes
2022-11-24	R22-11	AUTOSAR Release Management	• Initial release

Disclaimer

Contents

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Basic Software module CAN XL Transceiver Driver. The CAN XL Transceiver Driver is an extension of the CAN Transceiver Driver so this document shall only provide information and specifications which differ from the CAN Transceiver Driver. Some general information is given for a better understanding.

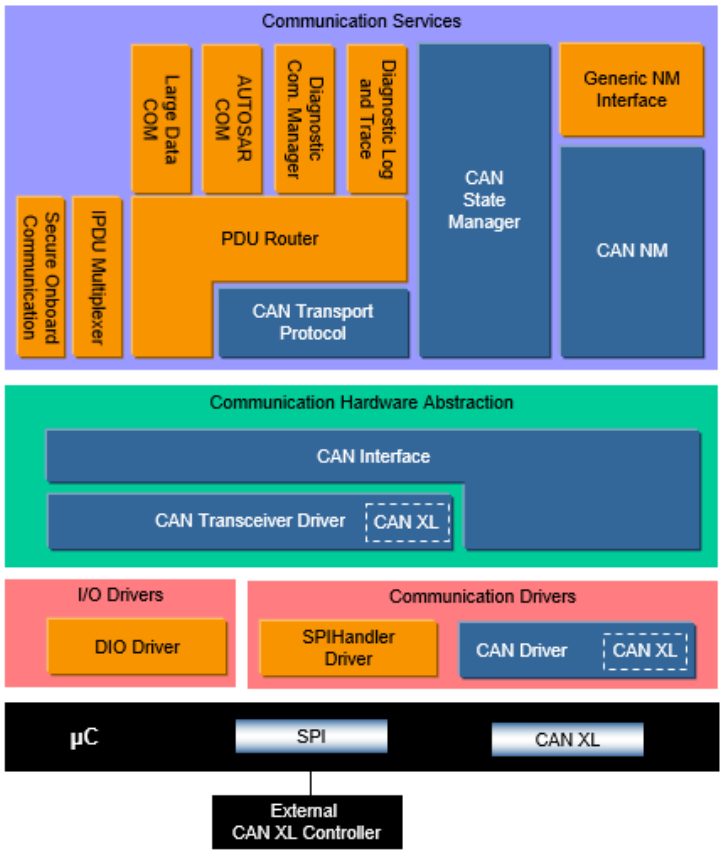

Figure 1.1: Autosar CanXL Layered Architecture

2 Acronyms and Abbreviations

The CAN XL Transceiver Driver does not define any local acronyms or abbreviations that are not included in the [1, AUTOSAR glossary].

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[3] Specification of CAN Transceiver Driver
AUTOSAR_CP_SWS_CANTransceiverDriver

[4] Specification for CAN XL Driver
AUTOSAR_CP_SWS_CANXLDriver

[5] Specification of CAN Driver
AUTOSAR_CP_SWS_CANDriver

[6] Specification of CAN Interface
AUTOSAR_CP_SWS_CANInterface

[7] Specification of Ethernet Interface
AUTOSAR_CP_SWS_EthernetInterface

[8] CiA 610-1 version 1.0.0 (DSP) - CAN XL specifications and test plans - Part 1: Data link layer and physical coding sub-layer requirements
http://www.can-cia.org

[9] CiA 611-1 version 1.0.0 (DSP) - CAN XL higher layer functions - Part 1: Definition of service data unit types
http://www.can-cia.org

[10] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[11] Requirements on CAN
AUTOSAR_CP_RS_CAN

[12] Specification of Ethernet Transceiver Driver
AUTOSAR_CP_SWS_EthernetTransceiverDriver

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [2, SWS BSW General], which is also valid for CAN XL Transceiver Driver.

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

4 Constraints and assumptions

The constraints and assumptions of the CAN XL Transceiver Driver are the same as for the CAN Transceiver Driver module.

5 Dependencies to other modules

The CAN XL Transceiver Driver module extends the CAN Transceiver Driver [3] and has interfaces towards the [4, CAN XL Driver], [5, CAN Driver], the [6, CAN Interface] and the [7, Ethernet Interface].

5.1 File Structure

This section explains the file structure of the CAN XL Transceiver Driver module.

5.1.1 Code File Structure

For details, refer to the section 5.1.6 “Code file structure” in [2, SWS BSW General].

6 Requirements Tracing

The following tables reference the requirements specified in [10] as well as [11] and link to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[CP_SWS_CanXLTrcv_00001]
[SRS_BSW_00310]	API naming convention	[CP_SWS_CanXLTrcv_10001] [CP_SWS_CanXLTrcv_10007] [CP_SWS_CanXLTrcv_10008] [CP_SWS_CanXLTrcv_10009] [CP_SWS_CanXLTrcv_10010] [CP_SWS_CanXLTrcv_10011]
[SRS_BSW_00327]	Error values naming convention	[CP_SWS_CanXLTrcv_10002]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/ disabling of detection and reporting of development errors.	[CP_SWS_CanXLTrcv_10002]
[SRS_BSW_00357]	For success/failure of an API call a standard return type shall be defined	[CP_SWS_CanXLTrcv_10001]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[CP_SWS_CanXLTrcv_10001] [CP_SWS_CanXLTrcv_10007] [CP_SWS_CanXLTrcv_10008] [CP_SWS_CanXLTrcv_10009] [CP_SWS_CanXLTrcv_10010] [CP_SWS_CanXLTrcv_10011]
[SRS_BSW_00385]	List possible error notifications	[CP_SWS_CanXLTrcv_10002]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[CP_SWS_CanXLTrcv_10002]
[SRS_BSW_00406]	API handling in uninitialized state	[CP_SWS_CanXLTrcv_10001] [CP_SWS_CanXLTrcv_10007] [CP_SWS_CanXLTrcv_10008] [CP_SWS_CanXLTrcv_10009] [CP_SWS_CanXLTrcv_10010] [CP_SWS_CanXLTrcv_10011]
[SRS_Can_01097]	CAN Bus Transceiver driver API shall be synchronous	[CP_SWS_CanXLTrcv_10001]
[SRS_Can_02002]	The CAN bus transceiver driver shall support the configuration for more than one bus	[CP_SWS_CanXLTrcv_00001] [CP_SWS_CanXLTrcv_00002] [CP_SWS_CanXLTrcv_00033] [CP_SWS_CanXLTrcv_00034] [CP_SWS_CanXLTrcv_00036] [CP_SWS_CanXLTrcv_00037] [CP_SWS_CanXLTrcv_00050] [CP_SWS_CanXLTrcv_00051] [CP_SWS_CanXLTrcv_00052] [CP_SWS_CanXLTrcv_10001] [CP_SWS_CanXLTrcv_10007] [CP_SWS_CanXLTrcv_10008] [CP_SWS_CanXLTrcv_10009] [CP_SWS_CanXLTrcv_10010] [CP_SWS_CanXLTrcv_10011]
[SRS_Eth_00040]	The Ethernet Transceiver Driver shall provide access to the link state.	[CP_SWS_CanXLTrcv_10008]

△

Requirement	Description	Satisfied by
[SRS_Eth_00108]	The Ethernet Transceiver Driver shall be able to wake-up an Ethernet network.	[CP_SWS_CanXLTrcv_10007]

Table 6.1: Requirements Tracing

7 Functional specification

The following section only describes additional CAN XL Transceiver specific 'Functional specifications'. The Specification of CAN Transceiver Driver [3] is the base of this `CanXLTransceiverDriver` 'extension'.

For a description of the specific functional behavior of CAN XL refer to the Specification of the `CanXLDriver`

7.1 Initialization

[CP_SWS_CanXLTrcv_00001]

Upstream requirements: SRS_Can_02002, SRS_BSW_00101

⌈The `CanTrcv_Init()` shall be extended by all functionality necessary to initialize the `CanXLTransceiverDriver`.⌋

7.2 Communication state handling

The operation mode of each CAN XL Transceiver is controlled only via the CAN stack over `CanTrcv_SetOpMode()` as described in the Specification of CAN Transceiver Driver [3]. Any transceiver mode request over the Ethernet stack has no influence on the actual operation mode of a CAN XL Transceiver. The corresponding APIs are only required so that the `CanXLTransceiverDriver` is compatible with the Ethernet stack. In order to inform the Ethernet stack whether a communication via the CAN bus is possible or not, the link state known in the Ethernet stack is reused. The current link state consists of the CAN XL Transceiver operation mode (`CANTRCV_TRCVMODE_NORMAL`, `CANTRCV_TRCVMODE_STANDBY` or `CANTRCV_TRCVMODE_SLEEP`), the status (`CAN_ERRORSTATE_ACTIVE`, `CAN_ERRORSTATE_PASSIVE` or `CAN_ERRORSTATE_BUSOFF`) of the corresponding physical CAN bus and the requested link state (`ETHTRCV_LINK_STATE_DOWN` or `ETHTRCV_LINK_STATE_ACTIVE`) from the Ethernet stack.

The Ethernet transceiver mode is stored by the `CAN XL Transceiver Driver`, and returned on request. See Section 8.3.3 and Section 8.3.4 for the behavior of `CanXLTrcv_SetTransceiverMode()` and `CanXLTrcv_GetTransceiverMode()`.

[CP_SWS_CanXLTrcv_00050]

Upstream requirements: SRS_Can_02002

⌈The Ethernet transceiver mode shall initially be set to `ETH_MODE_DOWN`.⌋

The requested Ethernet link state is stored by the CAN XL Transceiver Driver, and updated via CanXLTrcv_TransceiverLinkStateRequest as described in Section 8.3.2.

[CP_SWS_CanXLTrcv_00051]

Upstream requirements: SRS_Can_02002

⌈The requested Ethernet link state shall initially be set to ETHTRCV_LINK_STATE_DOWN.⌋

The status of the physical CAN bus to which a CAN XL Transceiver is connected is reported as CAN error state from the CanXLDriver to the CanXLTransceiverDriver via CanXLTrcv_ReportErrorState each time it is changed.

[CP_SWS_CanXLTrcv_00002]

Upstream requirements: SRS_Can_02002

⌈The link state reported by CanXLTrcv_GetLinkState shall be ETHTRCV_LINK_STATE_ACTIVE if the CAN XL Transceiver operation mode is CANTRCV_TRCVMODE_NORMAL, the stored CAN error state is CAN_ERRORSTATE_ACTIVE or CAN_ERRORSTATE_PASSIVE and the requested link state is ETHTRCV_LINK_STATE_ACTIVE.⌋

[CP_SWS_CanXLTrcv_00033]

Upstream requirements: SRS_Can_02002

⌈The link state reported by CanXLTrcv_GetLinkState shall be ETHTRCV_LINK_STATE_DOWN if the requested link state is ETHTRCV_LINK_STATE_DOWN (independent of the current CAN XL Transceiver operation mode and the stored CAN error state).⌋

[CP_SWS_CanXLTrcv_00034]

Upstream requirements: SRS_Can_02002

⌈The link state reported by CanXLTrcv_GetLinkState shall be ETHTRCV_LINK_STATE_DOWN if the stored CAN error state is CAN_ERRORSTATE_BUSOFF (independent of the current CAN XL Transceiver operation mode and the requested link state).⌋

[CP_SWS_CanXLTrcv_00052]

Upstream requirements: SRS_Can_02002

⌈The stored CAN error state shall be initialized as CAN_ERRORSTATE_BUSOFF.⌋

[CP_SWS_CanXLTrcv_00036]

Upstream requirements: SRS_Can_02002

⌈The link state reported by `CanXLTrcv_GetLinkState` shall be `ETHTRCV_LINK_STATE_DOWN` if the CAN XL Transceiver operation mode is `CANTRCV_TRCVMODE_STANDBY` or `CANTRCV_TRCVMODE_SLEEP` (independent of the stored CAN error state and the requested link state).⌋

7.3 Wake up handling

The wake up handling for each CAN XL Transceiver is only performed via the CAN stack as described in the Specification of CAN Transceiver Driver [3]. The Ethernet stack is neither part of the wake up handling nor is it informed about a occurred wake up event. The power saving state of a CAN XL Transceiver is indicated to the Ethernet stack as `ETHTRCV_LINK_STATE_DOWN` (refer to [CP_SWS_CanXLTrcv_00036]). After a successful wake up sequence the Ethernet stack can be informed over the link state (refer to [CP_SWS_CanXLTrcv_00002]) that a communication via the CAN bus is possible again.

[CP_SWS_CanXLTrcv_00037]

Upstream requirements: SRS_Can_02002

⌈The `CanXLTrcv_CheckWakeup` shall not actively participate in the wake up handling of the CAN XL transceiver.⌋

7.4 Error Classification

Section 7.2 "Error Handling" of the document "General Specification of Basic Software Modules" [19] describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules. Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.4.1 Development Errors

[CP_SWS_CanXLTrcv_10002] Definiton of development errors in module CanXL-Trcv

Upstream requirements: SRS_BSW_00327, SRS_BSW_00350, SRS_BSW_00385, SRS_BSW_-00386

⌈

Type of error	*Related error code*	*Error value*
API called with wrong parameter for the CAN transceiver	CANXLTRCV_E_INVALID_TRANSCEIVER	0x01
API called with null pointer parameter	CANXLTRCV_E_PARAM_POINTER	0x02
API service used without initialization	CANXLTRCV_E_UNINIT	0x11
Invalid error state	CANXLTRCV_E_INVALID_ERROR_STATE	0x30
Invalid link state	CANXLTRCV_E_INVALID_LINK_STATE	0x31

⌋

7.4.2 Runtime Errors

There are no addtitional runtime errors.

7.4.3 Production Errors

There are no addtitional production errors.

7.4.4 Extended Production Errors

There are no addtitional extended production errors.

8 API specification

Please note, that the CAN XL Transceiver Driver uses the MSN CanTrcv for parts that are shared with the [3, CAN Transceiver Driver] and the MSN CanXL-Trcv for extensions defined in this document. Deviating from SRS_BSW_00101 and SRS_BSW_00407, the CAN XL Transceiver Driver does not provide separate Init and GetVersionInfo APIs with the MSN CanXL. Following SWS_MemMap_00022, memory sections associated with APIs defined in this document will use the MSN CanXL, and also symbolic name values referring to containers defined in this document will use the MSN CanXL to follow TPS_ECUC_02108.

8.1 Imported types

In this chapter all types included from the following files are listed.

[CP_SWS_CanXLTrcv_10006] Definition of imported datatypes of module Can XLTrcv ⌈

Module	*Header File*	*Imported Type*
Can	Can_GeneralTypes.h	Can_ErrorStateType
Eth	Eth_GeneralTypes.h	Eth_ModeType
EthTrcv	Eth_GeneralTypes.h	EthTrcv_LinkStateType
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type definitions

There are no additional type definitions.

8.3 Function definitions

8.3.1 CanXLTrcv_ReportErrorState

[CP_SWS_CanXLTrcv_10001] Definition of API function CanXLTrcv_ReportErrorState

Upstream requirements: SRS_Can_02002, SRS_BSW_00310, SRS_BSW_00369, SRS_BSW_00357, SRS_BSW_00406, SRS_Can_01097

⌈

Service Name	CanXLTrcv_ReportErrorState	
Syntax	`Std_ReturnType CanXLTrcv_ReportErrorState ( uint8 Transceiver, Can_ErrorStateType ErrorState )`	
Service ID [hex]	0x30	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	Transceiver	CAN transceiver to which API call has to be applied
	ErrorState	New error state of the corresponding CAN controller
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Report was successful `E_NOT_OK`: Call was rejected
Description	Reports each change of the CAN error state.	
Available via	CanXLTrcv.h	

⌋

Note: The service `CanXLTrcv_ReportErrorState()` is implemented in `CanXLTransceiverDriver` and called by `CanXLDriver` after each change of the CAN error state at the corresponding CAN XL controller.

[CP_SWS_CanXLTrcv_00032] ⌈The CAN error state shall be stored for each CAN XL Transceiver in the `CanXLTransceiverDriver` and shall be updated at each call of the `CanXLTrcv_ReportErrorState()`.⌋

[CP_SWS_CanXLTrcv_00038] ⌈If development error detection is enabled: the function shall check that the service `CanTrcv_Init()` was previously called. If the check fails, the function shall raise the development error `CANXLTRCV_E_UNINIT`.⌋

[CP_SWS_CanXLTrcv_00039] ⌈If development error detection is enabled: the function shall check the parameter `Transceiver` for being valid. If the check fails, the function shall raise the development error `CANXLTRCV_E_INVALID_TRANSCEIVER`.⌋

[CP_SWS_CanXLTrcv_00040] ⌈If development error detection is enabled: the function shall check the parameter `ErrorState` for being valid. If the check fails, the function shall raise the development error `CANXLTRCV_E_INVALID_ERROR_STATE`.⌋

8.3.2 CanXLTrcv_TransceiverLinkStateRequest

[CP_SWS_CanXLTrcv_10011] Definition of API function CanXLTrcv_TransceiverLinkStateRequest

Upstream requirements: SRS_Can_02002, SRS_BSW_00310, SRS_BSW_00369, SRS_BSW_00406

⌈

Service Name	CanXLTrcv_TransceiverLinkStateRequest	
Syntax	`Std_ReturnType CanXLTrcv_TransceiverLinkStateRequest ( uint8 TrcvIdx, EthTrcv_LinkStateType LinkState )`	
Service ID [hex]	0x31	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different TrcvIdx. Non reentrant for the same TrcvIdx.	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Transceiver Driver
	LinkState	The link state of a physical connection.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the given link state for the given transceiver	
Available via	CanXLTrcv.h	

⌋

Note: This API is derived from Ethernet Transceiver Driver ([SWS_EthTrcv_91025]). For better understanding of the API's original intention you may check [7, Ethernet Interface] and [12, Ethernet Transceiver Driver].

[CP_SWS_CanXLTrcv_00041] ⌈The requested link state shall be stored for each CAN XL Transceiver in the `CanXLTransceiverDriver` and shall be updated at each call of `CanXLTrcv_TransceiverLinkStateRequest()`.⌋

[CP_SWS_CanXLTrcv_00047] ⌈If development error detection is enabled: the function shall check that the service CanTrcv_Init was previously called. If the check fails, the function shall raise the development error `CANXLTRCV_E_UNINIT`.⌋

[CP_SWS_CanXLTrcv_00048] ⌈If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error CANXLTRCV_E_INVALID_TRANSCEIVER.⌋

[CP_SWS_CanXLTrcv_00049] ⌈If development error detection is enabled: the function shall check the parameter LinkState for being valid. If the check fails, the function shall raise the development error CANXLTRCV_E_INVALID_LINK_STATE.⌋

See also Section 7.2.

8.3.3 CanXLTrcv_SetTransceiverMode

[CP_SWS_CanXLTrcv_10010] Definition of API function CanXLTrcv_SetTransceiverMode

Upstream requirements: SRS_Can_02002, SRS_BSW_00310, SRS_BSW_00369, SRS_BSW_00406

⌈

Service Name	CanXLTrcv_SetTransceiverMode	
Syntax	`Std_ReturnType CanXLTrcv_SetTransceiverMode ( uint8 TrcvIdx, Eth_ModeType TrcvMode )`	
Service ID [hex]	0x32	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Transceiver Driver
	TrcvMode	ETH_MODE_DOWN: disable the transceiver ETH_MODE_ACTIVE: enable the transceiver ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST: enable the transceiver and request to trigger a wake-up on the network, if the used PHY support such a feature. E.g. used for PHYs compliant to OA TC10
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied
Description	Enables / disables the indexed transceiver	
Available via	CanXLTrcv.h	

⌋

Note: This API is derived from Ethernet Transceiver Driver ([SWS_EthTrcv_00042]). For better understanding of the API's original intention you may check [7, Ethernet Interface] and [12, Ethernet Transceiver Driver].

[CP_SWS_CanXLTrcv_00005] ⌈The set transceiver mode shall be stored for each CAN XL Transceiver in the `CanXLTransceiverDriver` and shall be updated at each call of `CanXLTrcv_SetTransceiverMode()`. The CanXLTrcv shall directly call `EthIf_TrcvModeIndication()`.⌋

[CP_SWS_CanXLTrcv_00006] ⌈A new requested CanXLTrcv mode shall overwrite the last requested CanXLTrcv mode. If ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST was requested, ETH_MODE_ACTIVE shall be stored.⌋

[CP_SWS_CanXLTrcv_00011] ⌈If development error detection is enabled: the function shall check that the service CanTrcv_Init was previously called. If the check fails, the function shall raise the development error `CANXLTRCV_E_UNINIT`.⌋

[CP_SWS_CanXLTrcv_00012] ⌈If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error `CANXLTRCV_E_INVALID_TRANSCEIVER`.⌋

See also Section 7.2.

8.3.4 CanXLTrcv_GetTransceiverMode

[CP_SWS_CanXLTrcv_10009] Definition of API function CanXLTrcv_GetTransceiverMode

Upstream requirements: SRS_Can_02002, SRS_BSW_00310, SRS_BSW_00369, SRS_BSW_00406

⌈

Service Name	CanXLTrcv_GetTransceiverMode	
Syntax	`Std_ReturnType CanXLTrcv_GetTransceiverMode ( uint8 TrcvIdx, Eth_ModeType* TrcvModePtr )`	
Service ID [hex]	0x33	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Transceiver Driver
Parameters (inout)	None	
Parameters (out)	TrcvModePtr	ETH_MODE_DOWN: the transceiver is disabled ETH_MODE_ACTIVE: the transceiver is enable
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: transceiver could not be initialized
Description	Obtains the state of the indexed transceiver	
Available via	CanXLTrcv.h	

⌋

Note: This API is derived from Ethernet Transceiver Driver ([SWS_EthTrcv_00048]). For better understanding of the API's original intention you may check [7, Ethernet Interface] and [12, Ethernet Transceiver Driver].

[CP_SWS_CanXLTrcv_00015] ⌈The function shall return the stored requested transceiver mode in `CanXLTransceiverDriver` for the corresponding transceiver.⌋

[CP_SWS_CanXLTrcv_00016] ⌈If development error detection is enabled: the function shall check that the service CanTrcv_Init was previously called. If the check fails, the function shall raise the development error `CANXLTRCV_E_UNINIT`.⌋

[CP_SWS_CanXLTrcv_00017] ⌈If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error `CANXLTRCV_E_INVALID_TRANSCEIVER`.⌋

[CP_SWS_CanXLTrcv_00018] ⌈If development error detection is enabled: the function shall check the parameter TrcvModePtr for being valid. If the check fails, the function shall raise the development error `CANXLTRCV_E_PARAM_POINTER`.⌋

See also Section 7.2.

8.3.5 CanXLTrcv_GetLinkState

[CP_SWS_CanXLTrcv_10008] Definition of API function CanXLTrcv_GetLinkState

Upstream requirements: SRS_Eth_00040, SRS_Can_02002, SRS_BSW_00310, SRS_BSW_00369, SRS_BSW_00406

⌈

Service Name	CanXLTrcv_GetLinkState	
Syntax	`Std_ReturnType CanXLTrcv_GetLinkState ( uint8 TrcvIdx, EthTrcv_LinkStateType* LinkStatePtr )`	
Service ID [hex]	0x34	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Transceiver Driver
Parameters (inout)	None	
Parameters (out)	LinkStatePtr	ETHTRCV_LINK_STATE_DOWN: transceiver is disconnected ETHTRCV_LINK_STATE_ACTIVE: transceiver is connected
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: transceiver could not be initialized

▽

△

Description	Obtains the link state of the indexed transceiver
Available via	CanXLTrcv.h

⌋

Note: This API is derived from Ethernet Transceiver Driver ([SWS_EthTrcv_00061]). For better understanding of the API's original intention you may check [7, Ethernet Interface] and [12, Ethernet Transceiver Driver].

[CP_SWS_CanXLTrcv_00020] ⌈If development error detection is enabled: the function shall check that the service CanTrcv_Init() was previously called. If the check fails, the function shall raise the development error `CANXLTRCV_E_UNINIT`.⌋

[CP_SWS_CanXLTrcv_00021] ⌈If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error `CANXLTRCV_E_INVALID_TRANSCEIVER`.⌋

[CP_SWS_CanXLTrcv_00022] ⌈If development error detection is enabled: the function shall check the parameter LinkStatePtr for being valid. If the check fails, the function shall raise the development error `CANXLTRCV_E_PARAM_POINTER`.⌋

See also Section 7.2.

8.3.6 CanXLTrcv_CheckWakeup

[CP_SWS_CanXLTrcv_10007] Definition of API function CanXLTrcv_CheckWakeup

Upstream requirements: SRS_Eth_00108, SRS_Can_02002, SRS_BSW_00310, SRS_BSW_00369, SRS_BSW_00406

⌈

Service Name	CanXLTrcv_CheckWakeup	
Syntax	`Std_ReturnType CanXLTrcv_CheckWakeup ( uint8 TrcvIdx )`	
Service ID [hex]	0x35	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Transceiver Driver
Parameters (inout)	None	

▽

△

Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The function has been successfully executed E_NOT_OK: The function could not be successfully executed
Description	Service is called by EthIf in case a wake-up interrupt is detected.	
Available via	CanXLTrcv.h	

⌋

Note: This API is derived from Ethernet Transceiver Driver ([SWS_EthTrcv_00134]). For better understanding of the API's original intention you may check [7, Ethernet Interface] and [12, Ethernet Transceiver Driver].

[CP_SWS_CanXLTrcv_00035] ⌈This function shall have no functional behaviour and return E_OK.⌋

[CP_SWS_CanXLTrcv_00027] ⌈If development error detection is enabled: The function CanXLTrcv_CheckWakeup() shall check that the service CanTrcv_Init was previously called. If the check fails, the function shall raise the development error CANXLTRCV_E_UNINIT.⌋

[CP_SWS_CanXLTrcv_00028] ⌈If development error detection is enabled: The function CanXLTrcv_CheckWakeup() shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error CANXLTRCV_E_INVALID_TRANSCEIVER.⌋

8.4 Callback notifications

8.5 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

8.6 Expected interfaces

In this chapter all interfaces required from other modules are listed.

8.6.1 Mandatory interfaces

The CAN XL Transceiver Driver does not specify any mandatory interfaces.

8.6.2 Optional interfaces

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

[CP_SWS_CanXLTrcv_10004] Definition of optional interfaces requested by module CanXLTrcv ⌈

API Function	*Header File*	*Description*
EthIf_TrcvModeIndication	EthIf.h	Called asynchronously when a mode change has been read out. If the function is triggered by previous call of EthTrcv_SetTransceiverMode it can directly be called within the trigger function.

⌋

8.6.3 Configurable interfaces

The CAN XL Transceiver Driver does not specify any configurable interfaces.

9 Sequence diagrams

9.1 CanXL BusOff handling for Ethernet

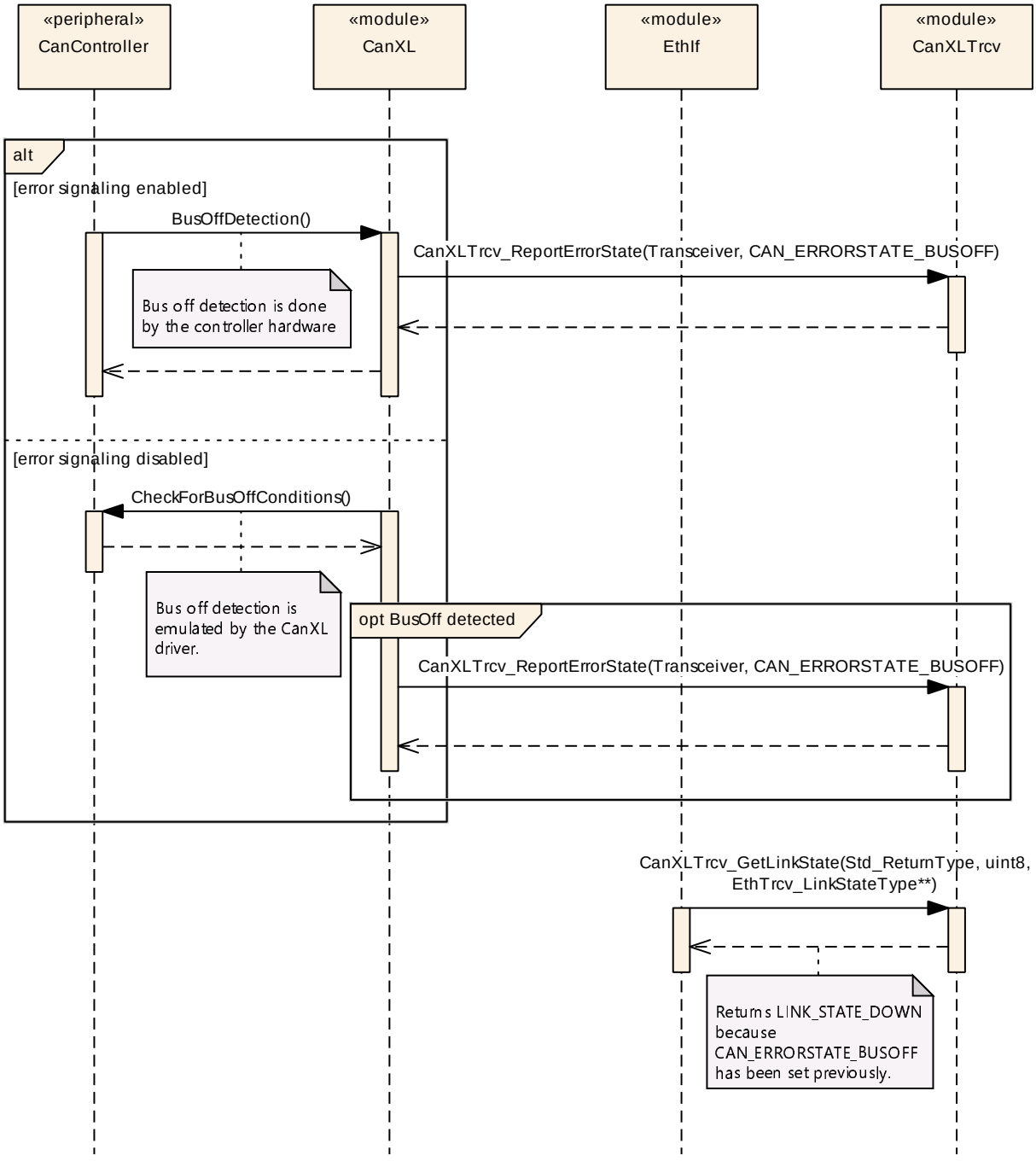

Figure 9.1: CanXL BusOff handling for Ethernet

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module CAN XL Transceiver Driver.

Chapter 10.3 specifies published information of the module CAN XL Tranceiver Driver.

10.1 How to read this chapter

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

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 7 and Chapter 8.

10.2.1 CanXLTrcvChannel

[ECUC_CanTrcv_00195] Definition of EcucParamConfContainerDef CanXLTrcv Channel ⌈

Container Name	CanXLTrcvChannel		
Parent Container	CanTrcvChannel		
Description	This container is specified in the SWS CAN XL Transceiver Driver and represents a CAN XL transceiver channel. If this container is present, the CAN transceiver will provide the extended CanXLTrcv API.		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanXLTrcvEthEcucPartitionRef	0..1	[ECUC_CanTrcv_00196]

No Included Containers

⌋

[ECUC_CanTrcv_00196] Definition of EcucReferenceDef CanXLTrcvEthEcucPartitionRef ⌈

Parameter Name	CanXLTrcvEthEcucPartitionRef		
Parent Container	CanXLTrcvChannel		
Description	Maps the Ethernet Interface access to the CAN XL transceiver channel to zero or one ECUC partitions.		
Multiplicity	0..1		
Type	Reference to EcucPartition		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.3 Published Information

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

A Not applicable requirements

[CP_SWS_CanXLTrcv_NA_00999] ⌈These requirements are not applicable to this specification.⌋

B Change History

Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These constraints and specification items do not appear as hyperlinks in the document.

B.1 Change History of this document according to AUTOSAR Release R22-11

B.1.1 Added Specification Items in R22-11

Number	Heading
[CP_SWS_CanXLTrcv_00001]	
[CP_SWS_CanXLTrcv_00002]	
[CP_SWS_CanXLTrcv_00005]	
[CP_SWS_CanXLTrcv_00006]	
[CP_SWS_CanXLTrcv_00011]	
[CP_SWS_CanXLTrcv_00012]	
[CP_SWS_CanXLTrcv_00015]	
[CP_SWS_CanXLTrcv_00016]	
[CP_SWS_CanXLTrcv_00017]	
[CP_SWS_CanXLTrcv_00018]	
[CP_SWS_CanXLTrcv_00020]	
[CP_SWS_CanXLTrcv_00021]	
[CP_SWS_CanXLTrcv_00022]	
[CP_SWS_CanXLTrcv_00027]	
[CP_SWS_CanXLTrcv_00028]	
[CP_SWS_CanXLTrcv_00032]	
[CP_SWS_CanXLTrcv_00033]	
[CP_SWS_CanXLTrcv_00034]	
[CP_SWS_CanXLTrcv_00035]	
[CP_SWS_CanXLTrcv_00036]	
[CP_SWS_CanXLTrcv_00037]	
[CP_SWS_CanXLTrcv_00038]	
[CP_SWS_CanXLTrcv_00039]	
[CP_SWS_CanXLTrcv_00040]	
[CP_SWS_CanXLTrcv_00041]	
[CP_SWS_CanXLTrcv_00047]	
[CP_SWS_CanXLTrcv_00048]	

▽

△

Number	Heading
[CP_SWS_CanXLTrcv_00049]	
[CP_SWS_CanXLTrcv_00050]	
[CP_SWS_CanXLTrcv_00051]	
[CP_SWS_CanXLTrcv_00052]	
[CP_SWS_CanXLTrcv_10001]	Definition of API function CanXLTrcv_ReportErrorState
[CP_SWS_CanXLTrcv_10002]	Definiton of development errors in module CanXLTrcv
[CP_SWS_CanXLTrcv_10004]	Definition of optional interfaces in module CanXLTrcv
[CP_SWS_CanXLTrcv_10006]	Definition of imported datatypes of module CanXLTrcv
[CP_SWS_CanXLTrcv_10007]	Definition of API function CanXLTrcv_CheckWakeup
[CP_SWS_CanXLTrcv_10008]	Definition of API function CanXLTrcv_GetLinkState
[CP_SWS_CanXLTrcv_10009]	Definition of API function CanXLTrcv_GetTransceiverMode
[CP_SWS_CanXLTrcv_10010]	Definition of API function CanXLTrcv_SetTransceiverMode
[CP_SWS_CanXLTrcv_10011]	Definition of API function CanXLTrcv_TransceiverLinkState Request
[CP_SWS_CanXLTrcv_NA_00999]	

Table B.1: Added Specification Items in R22-11

B.1.2 Changed Specification Items in R22-11

none

B.1.3 Deleted Specification Items in R22-11

none

B.2 Change History of this document according to AUTOSAR Release R23-11

B.2.1 Added Specification Items in R23-11

none

B.2.2 Changed Specification Items in R23-11

none

B.2.3 Deleted Specification Items in R23-11

none

B.3 Change History of this document according to AUTOSAR Release R24-11

B.3.1 Added Specification Items in R24-11

none

B.3.2 Changed Specification Items in R24-11

Number	Heading
[CP_SWS_CanXLTrcv_10001]	Definition of API function CanXLTrcv_ReportErrorState
[CP_SWS_CanXLTrcv_10007]	Definition of API function CanXLTrcv_CheckWakeup
[CP_SWS_CanXLTrcv_10008]	Definition of API function CanXLTrcv_GetLinkState
[CP_SWS_CanXLTrcv_10009]	Definition of API function CanXLTrcv_GetTransceiverMode
[CP_SWS_CanXLTrcv_10010]	Definition of API function CanXLTrcv_SetTransceiverMode
[CP_SWS_CanXLTrcv_10011]	Definition of API function CanXLTrcv_TransceiverLinkState Request

Table B.2: Changed Specification Items in R24-11

B.3.3 Deleted Specification Items in R24-11

none