

Document Title	Specification of TTCAN Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	432
Document Classification	Standard

Document Version	1.2.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History			
Date	Version	Changed by	Change Description
12.12.2011	1.2.0	AUTOSAR Administration	• Provided min/max values of configuration parameters • New traceability matrix
03.11.2010	1.1.0	AUTOSAR Administration	Updated artifacts of configuration section
02.12.2009	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

This specification and the material contained in it, as released by AUTOSAR is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only.

For any other purpose, no part of the specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Advice for users

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

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

Table of Content

1	Introduction and functional overview	5
2	Acronyms and abbreviations	7
3	Related documentation.....	8
3.1	Input documents.....	8
3.2	Related standards and norms	8
4	Constraints and assumptions	9
5	Dependencies to other modules.....	10
5.1	TTCAN Interface	10
6	Requirements traceability	11
7	Functional specification	13
7.1	TTCAN Controller State Machine.....	13
7.1.1	TTCAN Controller specific State Description.....	13
7.1.2	TTCAN Controller specific State Transitions.....	13
7.2	L-PDU Transmission	14
7.2.1	Priority Inversion	14
7.3	L-PDU Reception	15
7.4	Synchronization.....	15
7.4.1	Event Synchronisation.....	15
7.5	Time-Triggered Operation	16
7.6	Application Watchdog.....	16
7.7	TTCAN error handling	16
7.8	Error Classification	17
8	API specification	18
8.1	Imported types.....	18
8.2	Type definitions	18
8.2.1	Can_TTTimeType	18
8.2.2	Can_TTMasterSlaveModeType	18
8.2.3	Can_TTSyncModeEnumType	19
8.2.4	Can_TTMasterStateType	19
8.2.5	Can_TTErrorLevelEnumType	19
8.2.6	Can_TTErrorLevelType.....	19
8.2.7	Can_TTTimeSourceType	20
8.3	Function definitions	20
8.3.1	Can_TTGetControllerTime	20
8.3.2	Can_TTGetMasterState	21
8.3.3	Can_TTGetNTUActual	21
8.3.4	Can_TTGetErrorLevel.....	22
8.3.5	Can_TTSetNextIsGap.....	23
8.3.6	Can_TTSetEndOfGap.....	23
8.3.7	Can_TTSetTimeCommand	24
8.3.8	Can_TTGlobalTimePreset	25

8.3.9	Can_TTSetExtClockSyncCommand	26
8.3.10	Can_TTSetNTUAdjust	26
8.4	Optional Function definitions	27
8.4.1	Can_TTGetSyncQuality	27
8.4.2	Can_TTSetTimeMark.....	28
8.4.3	Can_TTCancelTimeMark	28
8.4.4	Can_TTAckTimeMark	29
8.4.5	Can_TTEnableTimeMarkIRQ.....	29
8.4.6	Can_TTDisableTimeMarkIRQ	30
8.4.7	Can_TTGetTimeMarkIRQStatus	30
8.4.8	Can_TTReceive	31
8.5	Scheduled Functions.....	32
8.5.1	Can_TTMainFunction_IRQ	32
8.6	Expected Interfaces.....	32
8.6.1	Mandatory interfaces.....	32
9	Sequence diagrams	34
9.1	Interaction between Ttcan and TtcanIf module	34
9.2	Wakeup sequence.....	34
10	Configuration specification.....	35
10.1	How to read this chapter	35
10.1.1	Configuration and configuration parameters	35
10.1.2	Variant.....	35
10.1.3	Containers.....	35
10.1.4	Specification template for configuration parameters	36
10.2	Containers and configuration parameters	36
10.2.1	CanTTController.....	38
10.2.2	CanTTHardwareObjectTrigger	43
10.3	Published information.....	46
11	Not applicable requirements	47

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module TTCAN Driver (called “Ttcan module” in this document).

The base for this document is ISO 11898-4 [12]. It is assumed that the reader is familiar with this specification. This document will not describe TTCAN functionality again.

The Ttcan module is part of the lowest layer, performs the hardware access and offers a hardware independent API to the upper layer.
The only upper layer that has access to the Ttcan module is the TtcanIf module (see also BSW12092).

The Ttcan module is an extension of the Can module so this document shall only provide information and specifications which differ from the CAN stack. Some general information is given for a better understanding.

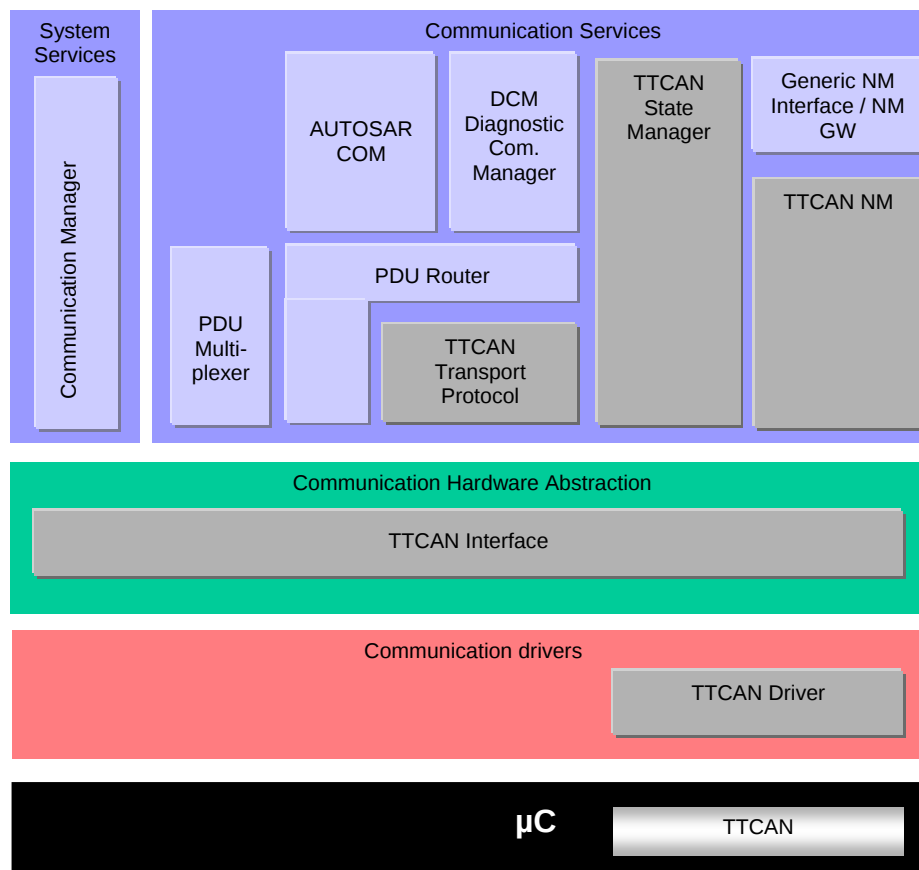


Figure 1 AUTOSAR TTCAN Layer Model

The Ttcan module provides services for initiating transmissions and calls the callback functions of the TtcanIf module for notifying events, independently from the hardware.

Furthermore, it provides services to control the behavior and state of the TTCAN controllers that are belonging to the same TTCAN Hardware Unit.

Several TTCAN controllers can be controlled by a single Ttcan module as long as they belong to the same TTCAN Hardware Unit.

Messages, which are configured for [exclusive time windows](#), will be transmitted periodically with every [Tx_Trigger](#) configured for this message ([continuous transmission](#)).

Messages, which are configured for [arbitrating time windows](#), will be transmitted only once per transmit request ([single shot](#)).

2 Acronyms and abbreviations

Abbreviation / Acronym:	Description:
Arbitrating time window	See ISO 11898-4 [12]
Basic cycle	See ISO 11898-4 [12]
BSW	Basic Software
CANIF	CAN Interface
Continuous transmission	Contrary to ' single shot ' a message will be transmitted cyclically even without a new transmit request.
Current time master	See ISO 11898-4 [12]
DLC	Data Length Code (part of L-PDU that describes the SDU length)
Cycle time	See ISO 11898-4 [12]
Exclusive time window	See ISO 11898-4 [12]
Global time	See ISO 11898-4 [12]
Hardware Receive Handle (HRH)	The Hardware Receive Handle (HRH) is defined and provided by the TTCAN driver. Typically each HRH represents exactly one hardware object. The HRH can be used to optimize software filtering.
Inner Priority Inversion	Transmission of a high-priority L-PDU is prevented by the presence of a pending low-priority L-PDU in the same transmit hardware object.
ISR	Interrupt Service Routine
L-PDU	Protocol Data Unit for the data link layer (DLL)
Local time	See ISO 11898-4 [12]
Matrix cycle	See ISO 11898-4 [12]
MCAL	Microcontroller Abstraction Layer
NTU	See ISO 11898-4 [12]
Reference message	See ISO 11898-4 [12]
Single shot	A message will be transmitted only once contrary to ' continuous transmission '.
System Matrix	See ISO 11898-4 [12]
Time gap	See ISO 11898-4 [12]
Time master	See ISO 11898-4 [12]
Time window	See ISO 11898-4 [12]
Transmission column	See ISO 11898-4 [12]
Transmit trigger event	See ISO 11898-4 [12]
TTCAN controller	A TTCAN controller serves exactly one physical channel.
TtcanDrv	CAN Driver module with enabled TTCAN functionality
TtcanIf	CAN Interface module with enabled TTCAN functionality
Tx_Trigger	See ISO 11898-4 [12]

3 Related documentation

All documents of the referenced CAN Driver document [5] are also valid for this document.

3.1 Input documents

- [1] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf
- [2] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf
- [3] Requirements on CAN
AUTOSAR_SRS_CAN.pdf
- [4] Specification of CAN Interface
AUTOSAR_SWS_CANInterface.pdf
- [5] Specification of CAN Driver
AUTOSAR_SWS_CANDriver.pdf
- [6] Specification of TTCAN Interface
AUTOSAR_SWS_TTCANInterface.pdf
- [7] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf
- [8] Specification of ECU State Manager
AUTOSAR_SWS_ECUSTateManager.pdf
- [9] Specification of Watchdog Driver
AUTOSAR_SWS_WatchdogDriver.pdf
- [10] Requirements on TTCAN
AUTOSAR_SRS_TTCAN.pdf
- [11] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

3.2 Related standards and norms

- [12] ISO11898-4 Road vehicles – Controller Area Network (CAN)
Part 4: Time-triggered communication

4 Constraints and assumptions

The constraints and assumptions of the Ttcan module are the same as for the CAN Driver module [5].

5 Dependencies to other modules

This chapter contains brief descriptions of configuration information and services, which are additionally required by the TTCAN Driver module from other modules.

The dependencies described in the referenced CAN Driver module [5] also apply for the TTCAN Driver module.

5.1 TTCAN Interface

The TTCAN Driver needs additional callback functions provided by the TTCAN Interface (refer to chapter 8.6).

6 Requirements traceability

Requirement	Satisfied by
-	TTCAN112
-	TTCAN043
-	TTCAN025
-	TTCAN059
-	TTCAN077
-	TTCAN014
-	TTCAN013
-	TTCAN116
-	TTCAN113
-	TTCAN028
-	TTCAN071
-	TTCAN070
-	TTCAN088
-	TTCAN021
-	TTCAN051
-	TTCAN029
-	TTCAN024
-	TTCAN110
-	TTCAN084
-	TTCAN040
-	TTCAN057
-	TTCAN067
-	TTCAN048
-	TTCAN034
-	TTCAN047
-	TTCAN012
-	TTCAN016
-	TTCAN039
-	TTCAN073
-	TTCAN038
-	TTCAN064
-	TTCAN115
-	TTCAN086
-	TTCAN080
-	TTCAN018
-	TTCAN046
-	TTCAN031
-	TTCAN065

-	TTCAN052
-	TTCAN032
-	TTCAN058
-	TTCAN074
-	TTCAN108
-	TTCAN049
-	TTCAN022
-	TTCAN033
-	TTCAN020
-	TTCAN017
-	TTCAN085
-	TTCAN044
-	TTCAN117
-	TTCAN068
-	TTCAN090
-	TTCAN125
-	TTCAN062
-	TTCAN154
-	TTCAN111
-	TTCAN078
-	TTCAN061
-	TTCAN036
-	TTCAN026
-	TTCAN155
-	TTCAN076
-	TTCAN041
-	TTCAN037
BSW00337	TTCAN010
BSW00387	TTCAN082
BSW441003	TTCAN156
BSW441005	TTCAN107, TTCAN102, TTCAN101, TTCAN104, TTCAN103, TTCAN106, TTCAN105, TTCAN006, TTCAN004, TTCAN005, TTCAN099, TTCAN092, TTCAN091, TTCAN094, TTCAN093, TTCAN096, TTCAN095, TTCAN098, TTCAN097
BSW441006	TTCAN007, TTCAN094, TTCAN095
BSW441007	TTCAN009, TTCAN126, TTCAN124
BSW441008	TTCAN120, TTCAN126, TTCAN082
BSW441009	TTCAN123, TTCAN121, TTCAN122

7 Functional specification

The following section only describes additional TTCAN specific 'Functional specifications'. The Specification of CAN Driver [5] is the base of this TtcanDrv 'extension'.

For a description of the specific functional behaviour of TTCAN refer to the Specification of the TTCAN Interface [6] and the TTCAN ISO Specification [12].

7.1 TTCAN Controller State Machine

An additional state SYNCHRONIZING has to be incorporated between the CAN controller states STOPPED and STARTED.

7.1.1 TTCAN Controller specific State Description

This chapter corresponds to the chapter "Can Controller State Machine" of the CAN Driver SWS.

TTCAN controller state SYNCHRONIZING:

The controller has left the state STOPPED and is ready for normal operation.

However, in order to participate on the bus, the controller needs to be synchronized to the global bus timing. As long as the controller is not synchronized to the bus, the controller stays in the state SYNCHRONIZING and error frames and acknowledges must not be sent. As soon as the controller is synchronized to the bus, the state of the controller changes from SYNCHRONIZING to STARTED.

For description of the procedure for a controller to become synchronized to the bus refer to [12].

TTCAN controller states IN_GAP and IN_SCHEDULE:

During normal operation the controller may switch between IN_SCHEDULE (normal time-triggered operation) and IN_GAP (as soon as a gap at the end of the current basic cycle is signaled until next reference message is sent on the bus to indicate the end of the gap). These state changes do not affect the Ttcan module.

7.1.2 TTCAN Controller specific State Transitions

State transition caused by function Can_SetControllerMode(CAN_T_START):

[TTCAN155] [Replaces CAN262: The function Can_SetControllerMode(CAN_T_START) shall wait for a limited time until the TTCAN controller has changed to the state SYNCHRONIZING (Compare to CAN371)] ()

Rational for TTCAN155: The controller will switch to the state SYNCHRONISING and will try to become synchronized to the bus. The procedure of synchronizing the

controller to the bus might be significantly longer than CanTimeoutTime. Therefore, only the change to the state SYNCHRONIZING shall be observed by the function Can_SetControllerMode (compare to CAN371) and the function Can_Mainfunction_Timeout (compare to CAN372).

State Transition caused by Severe Error (triggered by state change of TTCAN controller)

[TTCAN120] [

- STARTED → STOPPED
- triggered by hardware if the TTCAN controller reaches error level S3 (see TTCAN ISO Specification [12])
- The CanIf module is notified with the function CanIf_TTSevereError after STOPPED state is reached.] (BSW441008)

[TTCAN121] [After severe error detection, the TTCAN controller shall transition to the state STOPPED and the Ttcan Driver module shall ensure that the CAN controller doesn't participate on the network anymore.] (BSW441009)

[TTCAN122] [After severe error detection, the TtcanDrv shall cancel still pending messages without raising a cancellation notification.] (BSW441009)

[TTCAN123] [The TtcanDrv shall disable or suppress automatic severe error recovery.] (BSW441009)

7.2 L-PDU Transmission

Due to the time-triggered schedule, the L-PDU transmission is scheduled according to the Matrix cycle configured during initialization, i.e. a call of the function Can_Write() does not directly trigger an immediate transmission but rather stores the L-PDU in the corresponding HW object, which is scheduled for transmission in a specific [time window](#).

[TTCAN156] [It shall be possible to map all transmit message objects to specific [time windows](#) (see TTCAN ISO Specification [12]) by configuration (see [TTCANIF145_Conf](#), [TTCANIF146_Conf](#), [TTCANIF147_Conf](#), [TTCANIF148_Conf](#)).] (BSW441003)

7.2.1 Priority Inversion

[TTCAN154] [Multiplexed transmission and transmit cancellation described in the Specification of CAN Driver [5] shall only be used in arbitrating time windows.] ()

Note: In TTCAN communication priority inversion can only happen in arbitration time windows, because the L-PDU with its corresponding CAN ID, which has to be available in a HW object is fixed for exclusive time windows.

7.3 L-PDU Reception

The verification of the message reception is controlled by the HW using the configured trigger for reception CAN_TT_RX_TRIGGER (see [TTCAN145 Conf](#)).

A detailed description of reception triggering and the verification of message reception can be found in [12].

Dies gehört in das Interface:

Configuration hint: To suppress regular notifications of consecutive received messages, which maybe needed not that frequently as they arrive, the notifications can be switched-off. In this case the polling via “Read received data” and API `CanIf_ReadRxPduData()`, can be used to get the data from `CanIf`, when it is needed.

7.4 Synchronization

Since TTCAN supports time-triggered communication, the `TtcanDrv` needs to support maintaining the timing parameters and the master-controlled synchronization mechanisms.

[TTCAN004] [The `TtcanDrv` shall provide information from the TTCAN controller about the timing parameters (see [TTCAN090](#)), the synchronization state and the master state (see [TTCAN091](#)).] (BSW441005)

[TTCAN005] [The `TtcanDrv` shall provide means to influence the timing parameters of a TTCAN controller (see [TTCAN096](#), [TTCAN097](#), [TTCAN098](#), [TTCAN099](#)) during runtime, if the TTCAN controller acts as the timing master.] (BSW441005)

[TTCAN006] [The `TtcanDrv` shall provide the functionality of a timer, which is based on the time marks of the communication system, provided by the TTCAN controller.] (BSW441005)

7.4.1 Event Synchronisation

[TTCAN007] [The `TtcanDrv` shall support event-synchronized communication (see [TTCAN094](#), [TTCAN095](#)) (refer to ISO 11898-4 [12]).] (BSW441006)

7.5 Time-Triggered Operation

The events listed below are related to the time-triggered operation of a TTCAN system.

The following events shall be indicated to the application via the TtcanIf:

[TTCAN009] [

Event	Description	TtcanIf Function*
Application Watchdog	The application has not served the application watchdog in time.	TtcanIf_ApplWatchdogError
Change of error level	The error level of the TTCAN Controller changes between the states S0 – S3	TtcanIf_TimingError
Tx overflow	More Tx triggers than expected	TtcanIf_TimingError
Tx underflow	Less Tx triggers than expected	TtcanIf_TimingError
Global time error	Synchronization failed	TtcanIf_TimingError
Watch trigger	Watch trigger occurs	TtcanIf_TimingError
Initialization watch trigger	Init_watch_trigger is reached	TtcanIfTimingError
Gap	“Next is Gap” bit is set	TtcanIf_Gap
Start of Cycle	Start of a basic cycle (including the cycle count value).	TtcanIf_StartOfCycle
Time discontinuity	“Disc Bit” is set	TtcanIf_TimeDisc
Master state change	Change of the master state between potential and current time master	TtcanIf_MasterStateChange

* to be called in interrupt context (refer to chapter 8.6.1)] (BSW441007)

7.6 Application Watchdog

Note: The TTCAN Application Watchdog shall be served by using a Watchdog Driver instance (see [9] Watchdog Driver SWS). The Watchdog Driver instance shall serve the TTCAN Application Watchdog regularly before the timeout is reached.

Note: The timeout is the maximum time period between two consecutive calls to serve the TTCAN Application Watchdog.

Note: The Application Watchdog timeout limit shall be configured by CanTTControllerApplWatchdogLimit (see [TTCAN139 Conf](#)).

7.7 TTCAN error handling

This chapter corresponds to the chapter “Error handling” of the CAN Driver SWS.

[TTCAN124] [Either the function Can_TTMainFunction_IRQ() or an interrupt shall call the function CanIf_TTTimingError() with the corresponding event type, when error levels S1 or S2 (see TTCAN ISO Specification [12]) are reached.] (BSW441007)

[TTCAN126] [Either the function `Can_TTMainFunction_IRQ( )` or an interrupt shall call the function `CanIf_TTSevereError( )` with the corresponding event type, when error level S3 (see TTCAN ISO Specification [12]) is reached.] (BSW441007, BSW441008)

7.8 Error Classification

[TTCAN010] [The following errors and exceptions are specific to TTCAN

<i>Type or error</i>	<i>Relevance</i>	<i>Related error code</i>	<i>Value [hex]</i>
TTCAN Controller is not a potential time master	Development	CAN_TT_E_NOT_MASTER	0x08
TTCAN Controller is not a current time master	Development	CAN_TT_E_NOT_CURRENT_MASTER	0x09
TTCAN Controller transmits two consecutive reference messages which both have the "Disc_bit" set	Development	CAN_TT_E_CONSEQUITIVE_DISC	0x0a
Adjustment of global time fails, because external synchronization has been disabled during configuration	Development	CAN_TT_E_SYNC_DISABLED	0x0b

] (BSW00337)

8 API specification

Since the Ttcan module is an extension of the CAN Driver module, only specifications which differ from the CAN stack and which are TTCAN specific shall be provided within this chapter.

8.1 Imported types

Additional TTCAN specific imported types

[TTCAN125] [

Module	Imported Type
Can	Can_IdType
CanIf	CanIf_TTMasterStateType
	CanIf_TTSevereErrorEnumType
	CanIf_TTTimingErrorIRQType
Std_Types	Std_ReturnType

] ()

8.2 Type definitions

Additional TTCAN specific type definitions

8.2.1 Can_TTTimeType

[TTCAN084] [

Name:	Can_TTTimeType
Type:	uint16
Description:	16 bit value representing time values of TTCAN, e.g. cycle, local or global time

] ()

8.2.2 Can_TTMasterSlaveModeType

[TTCAN115] [

Name:	Can_TTMasterSlaveModeType	
Type:	Enumeration	
Range:	CAN_TT_BACKUP_MASTER	Master-Slave Mode: Backup master
	CAN_TT_CURRENT_MASTER	Master-Slave Mode: Current master
	CAN_TT_MASTER_OFF	Master-Slave Mode: Master off
	CAN_TT_SLAVE	Master-Slave Mode: Slave
Description:	Master-Slave Mode	

] ()

8.2.3 Can_TTSyncModeEnumType

[TTCAN116] [

Name:	Can_TTSyncModeEnumType	
Type:	Enumeration	
Range:	CAN_TT_IN_GAP	Sync mode: In_Gap
	CAN_TT_IN_SCHEDULE	Sync mode: In_Schedule
	CAN_TT_SYNC_OFF	Sync mode: Sync_Off
	CAN_TT_SYNCHRONIZING	Sync mode: Synchronizing
Description:	Sync mode	

] ()

8.2.4 Can_TTMasterStateType

[TTCAN085] [

Name:	Can_TTMasterStateType		
Type:	Structure		
Element:	Can_TTMasterSlaveModeType	masterSlaveMode	--
	uint8	refTriggerOffset	current value of ref trigger offset
	Can_TTSyncModeEnumType	syncMode	--
Description:	Master state type including sync mode, master-slave mode and current ref trigger offset		

] ()

8.2.5 Can_TTErrorLevelEnumType

[TTCAN117] [

Name:	Can_TTErrorLevelEnumType		
Type:	Enumeration		
Range:	CAN_TT_ERROR_S0	Error level S0: No Error	
	CAN_TT_ERROR_S1	Error level S1: Warning	
	CAN_TT_ERROR_S2	Error level S2: Error	
	CAN_TT_ERROR_S3	Error level S3: Fatal Error	
Description:	Error level (S0-S3)		

] ()

8.2.6 Can_TTErrorLevelType

[TTCAN086] [

Name:	Can_TTErrorLevelType		
Type:	Structure		
Element:	Can_TTErrorLevelEnumType	errorLevel	Error Level (S0-S3)
	uint8	maxMessageStatusCount	Max value of message status count (0-7)
	uint8	minMessageStatusCount	Min value of message status count (0-7)
Description:	TTCAN error level including min and max values of message status count		

] ()

8.2.7 Can_TTTimeSourceType

[TTCAN088] [

Name:	Can_TTTimeSourceType	
Type:	Enumeration	
Range:	CAN_TT_CYCLE_TIME	Time source: Cycle Time
	CAN_TT_GLOBAL_TIME	Time source: Global Time
	CAN_TT_LOCAL_TIME	Time source: Local Time
	CAN_TT_UNDEFINED	Time source: Undefined
Description:	Time source	

] ()

8.3 Function definitions

Additional TTCAN specific function definitions

8.3.1 Can_TTGetControllerTime

[TTCAN090] [

Service name:	Can_TTGetControllerTime	
Syntax:	<pre>void Can_TTGetControllerTime(uint8 Controller, Can_TTTimeType* Can_TTGlobalTime, Can_TTTimeType* Can_TTLocalTime, Can_TTTimeType* Can_TTCycleTime, uint8* Can_TTCycleCount)</pre>	
Service ID[hex]:	0x33	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Controller	Controller from which the time information shall be retrieved
Parameters (inout):	None	
Parameters (out):	Can_TTGlobalTime	Address to store return value: Global time
	Can_TTLocalTime	Address to store return value: Local time
	Can_TTCycleTime	Address to store return value: Cycle time
	Can_TTCycleCount	Address to store return value: Cycle count value
Return value:	None	
Description:	Gets the current values for the global, local and cycle time and the cycle count of the controller	

Note: A Std_ReturnType is needed for all Functions of chapter 8:

Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
----------------	---

] ()

[TTCAN012] [If development error detection for the Ttcan module is enabled: The function Can_TTGetControllerTime() shall raise the error CAN_E_UNINIT if the driver is not yet initialized.] ()

[TTCAN013] [If development error detection for the Ttcan module is enabled: The function Can_TTGetControllerTime() shall raise the error CAN_E_PARAM_CONTROLLER if the parameter Controller is out of range.] ()

[TTCAN014] [If development error detection for the Ttcan module is enabled: The function Can_TTGetControllerTime() shall raise the error CAN_E_PARAM_POINTER and shall return CAN_NOT_OK if the parameter Can_TTGlobalTime or the parameter Can_TTLocalTime or the parameter Can_TTCycleTime or the parameter Can_TTCycleCount is a NULL pointer.] ()

8.3.2 Can_TTGetMasterState

[TTCAN091] [

Service name:	Can_TTGetMasterState	
Syntax:	void Can_TTGetMasterState(uint8 Controller, Can_TTMasterStateType* Can_TTMasterState)	
Service ID[hex]:	0x34	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Controller	Controller from which the master state shall be retrieved
Parameters (inout):	None	
Parameters (out):	Can_TTMasterState	Address to store return value: Master state
Return value:	None	
Description:	Gets the master state. The master state includes the sync mode (sync_off, synchronizing, in_gap, in_schedule) the master-slave mode (master_off, slave, backup_master, current_master) and the current value for ref trigger offset.	

] (BSW441005)

[TTCAN016] [If development error detection for the Ttcan module is enabled: The function Can_TTGetMasterState() shall raise the error CAN_E_UNINIT if the driver is not yet initialized.] ()

[TTCAN017] [If development error detection for the Ttcan module is enabled: The function Can_TTGetMasterState() shall raise the error CAN_E_PARAM_CONTROLLER if the parameter Controller is out of range.] ()

[TTCAN018] [If development error detection for the Ttcan module is enabled: The function Can_TTGetMasterState() shall raise the error CAN_E_PARAM_POINTER and shall return CAN_NOT_OK if the parameter Can_TTMasterState is a NULL pointer.] ()

8.3.3 Can_TTGetNTUActual

[TTCAN092] [

Service name:	Can_TTGetNTUActual	
Syntax:	<pre>void Can_TTGetNTUActual(uint8 Controller, Can_TTTURType* Can_TTTURAct)</pre>	
Service ID[hex]:	0x35	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Controller	Controller from which the NTU vale shall be retrieved
Parameters (inout):	None	
Parameters (out):	Can_TTTURAct	Address to store return value: Actual value of NTU. Value is given in microseconds.
Return value:	None	
Description:	Gets the actual value of NTU (network time unit). Together with the local oscillator period, the actual value of NTU can be derived from the actual value of TUR.	

] (BSW441005)

[TTCAN020] [If development error detection for the Ttcan module is enabled: The function Can_TTGetNTUActual() shall raise the error CAN_E_UNINIT if the driver is not yet initialized.] ()

[TTCAN021] [If development error detection for the Ttcan module is enabled: The function Can_TTGetNTUActual() shall raise the error CAN_E_PARAM_CONTROLLER if the parameter Controller is out of range.] ()

[TTCAN022] [If development error detection for the Ttcan module is enabled: The function Can_TTGetNTUActual() shall raise the error CAN_E_PARAM_POINTER and shall return CAN_NOT_OK if the parameter Can_TTNTUAct is a NULL pointer.] ()

8.3.4 Can_TTGetErrorLevel

[TTCAN093] [

Service name:	Can_TTGetErrorLevel	
Syntax:	<pre>void Can_TTGetErrorLevel(uint8 Controller, Can_TTErrorLevelType* Can_TTErrorLevel)</pre>	
Service ID[hex]:	0x36	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Controller	Controller from which the error level shall be retrieved
Parameters (inout):	None	
Parameters (out):	Can_TTErrorLevel	Address to store return value: Error level
Return value:	None	
Description:	Gets the error level. This includes the severity of the error level (S0-S3) and the minimum and maximum value of the message status count.	

] (BSW441005)

[TTCAN024] [If development error detection for the Ttcan module is enabled: The function `Can_TTGetErrorLevel()` shall raise the error `CAN_E_UNINIT` if the driver is not yet initialized.] ()

[TTCAN025] [If development error detection for the Ttcan module is enabled: The function `Can_TTGetErrorLevel()` shall raise the error `CAN_E_PARAM_CONTROLLER` if the parameter `Controller` is out of range.] ()

[TTCAN026] [If development error detection for the Ttcan module is enabled: The function `Can_TTGetErrorLevel()` shall raise the error `CAN_E_PARAM_POINTER` and shall return `CAN_NOT_OK` if the parameter `Can_TTErrorLevel` is a NULL pointer.] ()

8.3.5 Can_TTSetNextIsGap

[TTCAN094] [

Service name:	Can_TTSetNextIsGap
Syntax:	void Can_TTSetNextIsGap(uint8 Controller)
Service ID[hex]:	0x37
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	Controller Controller for which the "next is gap" indication shall be set.
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Sets the "Next_is_Gap" bit.

] (BSW441005, BSW441006)

[TTCAN028] [If development error detection for the Ttcan module is enabled: The function `Can_TTSetNextIsGap()` shall raise the error `CAN_E_UNINIT` if the driver is not yet initialized.] ()

[TTCAN029] [If development error detection for the Ttcan module is enabled: The function `Can_TTSetNextIsGap()` shall raise the error `CAN_E_PARAM_CONTROLLER` if the parameter `Controller` is out of range.] ()

8.3.6 Can_TTSetEndOfGap

[TTCAN095] [

Service name:	Can_TTSetEndOfGap
Syntax:	void Can_TTSetEndOfGap(uint8 Controller)
Service ID[hex]:	0x38
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant

Parameters (in):	Controller	Controller for which the "set end of gap" indication shall be set
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Signals the end of a gap.	

] (BSW441005, BSW441006)

[TTCAN031] [The function Can_TTSetEndOfGap () shall only take effect if the TTCAN Controller is a potential [time master](#).] ()

[TTCAN032] [If development error detection for the Ttcan module is enabled: The function Can_TTSetEndOfGap () shall raise the error CAN_TT_E_NOT_MASTER if the TTCAN Controller is not a potential [time master](#).] ()

[TTCAN033] [If development error detection for the Ttcan module is enabled: The function Can_TTSetEndOfGap () shall raise the error CAN_E_UNINIT if the driver is not yet initialized.] ()

[TTCAN034] [If development error detection for the Ttcan module is enabled: The function Can_TTSetEndOfGap () shall raise the error CAN_E_PARAM_CONTROLLER if the parameter Controller is out of range.] ()

8.3.7 Can_TTSetTimeCommand

[TTCAN096] [

Service name:	Can_TTSetTimeCommand	
Syntax:	void Can_TTSetTimeCommand(uint8 Controller)	
Service ID[hex]:	0x39	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Controller	Controller for which the global time shall be adjusted
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Adjusts the global time at the beginning of the next basic cycle by the amount of "global time preset"	

] (BSW441005)

[TTCAN036] [If development error detection for the Ttcan module is enabled: The function Can_TTSetTimeCommand () shall raise the error CAN_TT_E_CONSEQUITIVE_DISC if two consecutive reference messages are transmitted wich both have the "Disc_bit" set.] ()

[TTCAN037] [If development error detection for the Ttcan module is enabled: The function Can_TTSetTimeCommand () shall raise the error

CAN_TT_E_SYNC_DISABLED if the adjustment of the global time fails, because the external synchronization has been disabled during configuration.] ()

[TTCAN038] [The function Can_TTSetTimeCommand() shall only take effect if the TTCAN Controller is the [current time master](#).] ()

[TTCAN039] [If development error detection for the Ttcan module is enabled: The function Can_TTSetTimeCommand() shall raise the error CAN_TT_E_NOT_CURRENT_MASTER if the TTCAN Controller is not the [current time master](#).] ()

[TTCAN040] [If development error detection for the Ttcan module is enabled: The function Can_TTSetTimeCommand() shall raise the error CAN_E_UNINIT if the driver is not yet initialized.] ()

[TTCAN041] [If development error detection for the Ttcan module is enabled: The function Can_TTSetTimeCommand() shall raise the error CAN_E_PARAM_CONTROLLER if the parameter Controller is out of range.] ()

8.3.8 Can_TTGlobalTimePreset

[TTCAN097] [

Service name:	Can_TTGlobalTimePreset	
Syntax:	<pre>void Can_TTGlobalTimePreset(uint8 Controller, Can_TTTimeType Can_TTGlobalTimePreset)</pre>	
Service ID[hex]:	0x3a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Controller	Controller for which the "global time preset" shall be set
	Can_TTGlobalTimePreset	New value for "global time preset"
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Sets the value of "global time preset".	

] (BSW441005)

[TTCAN043] [If development error detection for the Ttcan module is enabled: The function Can_TTGlobalTimePreset() shall raise the error CAN_E_UNINIT if the driver is not yet initialized.] ()

[TTCAN044] [If development error detection for the Ttcan module is enabled: The function Can_TTGlobalTimePreset() shall raise the error CAN_E_PARAM_CONTROLLER if the parameter Controller is out of range.] ()

8.3.9 Can_TTSetExtClockSyncCommand

[TTCAN098] [

Service name:	Can_TTSetExtClockSyncCommand
Syntax:	void Can_TTSetExtClockSyncCommand(uint8 Controller)
Service ID[hex]:	0x3b
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	Controller Controller for which the NTU shall be adjusted.
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Adjusts the NTU (network time unit) according to the value given by "NTU adjust". Together with the local oscillator period, "TUR adjust" can be derived from "NTU adjust".

] (BSW441005)

[TTCAN046] [The function Can_TTSetExtClockSyncCommand() shall only take effect if the TTCAN Controller is the [current time master](#).] ()

[TTCAN047] [If development error detection for the Ttcan module is enabled: The function Can_TTSetExtClockSyncCommand() shall raise the error CAN_TT_E_NOT_CURRENT_MASTER if the TTCAN Controller is not the [current time master](#).] ()

[TTCAN048] [If development error detection for the Ttcan module is enabled: The function Can_TTSetExtClockSyncCommand() shall raise the error CAN_E_UNINIT if the driver is not yet initialized.] ()

[TTCAN049] [If development error detection for the Ttcan module is enabled: The function Can_TTSetExtClockSyncCommand() shall raise the error CAN_E_PARAM_CONTROLLER if the parameter Controller is out of range.] ()

8.3.10 Can_TTSetNTUAdjust

[TTCAN099] [

Service name:	Can_TTSetNTUAdjust
Syntax:	void Can_TTSetNTUAdjust(uint8 Controller, Can_TTTURType Can_TTTURAdjust)
Service ID[hex]:	0x3c
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	Controller Controller for which the "NTU adjust" shall be set Can_TTTURAdjust New value for "NTU adjust" Value is given in microseconds.
Parameters	None

(inout):	
Parameters (out):	None
Return value:	None
Description:	Sets the value of "NTU adjust". Together with the local oscillator period, "TUR adjust" can be derived from "NTU adjust".

] (BSW441005)

[TTCAN051] [If development error detection for the Ttcan module is enabled: The function `Can_TTSetNTUAdjust()` shall raise the error `CAN_E_UNINIT` if the driver is not yet initialized.] ()

[TTCAN052] [If development error detection for the Ttcan module is enabled: The function `Can_TTSetNTUAdjust()` shall raise the error `CAN_E_PARAM_CONTROLLER` if the parameter `Controller` is out of range.] ()

8.4 Optional Function definitions

Additional optional TTCAN specific function definitions

8.4.1 Can_TTGetSyncQuality

[TTCAN101] [

Service name:	Can_TTGetSyncQuality	
Syntax:	<pre>void Can_TTGetSyncQuality(uint8 Controller, boolean* Can_TTClockSpeed, boolean* Can_TTGlobalTimePhase)</pre>	
Service ID[hex]:	0x47	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Controller	Controller from which the sync quality shall be retrieved
Parameters (inout):	None	
Parameters (out):	Can_TTClockSpeed	Address to store return value: True if the synchronization deviation is smaller than the "Synchronization deviation limit"
	Can_TTGlobalTimePhase	Address to store return value: True if the global time is in phase with the time master.
Return value:	None	
Description:	Gets the synchronization quality.	

] (BSW441005)

[TTCAN057] [If development error detection for the Ttcan module is enabled: The function `Can_TTGetSyncQuality()` shall raise the error `CAN_E_UNINIT` if the driver is not yet initialized.] ()

[TTCAN058] [If development error detection for the Ttcan module is enabled: The function `Can_TTGetSyncQuality()` shall raise the error `CAN_E_PARAM_CONTROLLER` if the parameter `Controller` is out of range.] ()

[TTCAN059] [If development error detection for the Ttcan module is enabled: The function Can_TTGetSyncQuality() shall raise the error CAN_E_PARAM_POINTER and shall return CAN_NOT_OK if the parameter Can_TTclockSpeed or the parameter Can_TTGlobalTimePhase is a NULL pointer.] ()

8.4.2 Can_TTSetTimeMark

[TTCAN102] [

Service name:	Can_TTSetTimeMark	
Syntax:	<pre>void Can_TTSetTimeMark(uint8 Controller, Can_TTTimeType Can_TTTimeMark, Can_TTTimeSourceType Can_TTTimeSource)</pre>	
Service ID[hex]:	0x48	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Controller	Controller for which the time mark shall be set
	Can_TTTimeMark	Gives the value of the time mark to be set.
	Can_TTTimeSource	Defines the time source for the time mark to be set.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Sets a new value for the time mark for the given time source.	

] (BSW441005)

[TTCAN061] [If development error detection for the Ttcan module is enabled: The function Can_TTSetTimeMark() shall raise the error CAN_E_UNINIT if the driver is not yet initialized.] ()

[TTCAN062] [If development error detection for the Ttcan module is enabled: The function Can_TTSetTimeMark() shall raise the error CAN_E_PARAM_CONTROLLER if the parameter Controller is out of range.] ()

8.4.3 Can_TTCancelTimeMark

[TTCAN103] [

Service name:	Can_TTCancelTimeMark	
Syntax:	<pre>void Can_TTCancelTimeMark(uint8 Controller)</pre>	
Service ID[hex]:	0x49	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Controller	Controller for which the time mark shall be cancelled.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	

Description:	Cancels the time mark.
---------------------	------------------------

] (BSW441005)

[TTCAN064] [If development error detection for the Ttcan module is enabled: The function `Can_TTCancelTimeMark()` shall raise the error `CAN_E_UNINIT` if the driver is not yet initialized.] ()

[TTCAN065] [If development error detection for the Ttcan module is enabled: The function `Can_TTCancelTimeMark()` shall raise the error `CAN_E_PARAM_CONTROLLER` if the parameter `Controller` is out of range.] ()

8.4.4 Can_TTAckTimeMark

[TTCAN104] [

Service name:	Can_TTAckTimeMark
Syntax:	<code>void Can_TTAckTimeMark(uint8 Controller)</code>
Service ID[hex]:	0x4a
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	Controller Controller for which the time mark shall be acknowledged.
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Acknowledges the time mark interrupt by resetting the flag in the interrupt vector register.

] (BSW441005)

[TTCAN067] [If development error detection for the Ttcan module is enabled: The function `Can_TTAckTimeMark()` shall raise the error `CAN_E_UNINIT` if the driver is not yet initialized.] ()

[TTCAN068] [If development error detection for the Ttcan module is enabled: The function `Can_TTAckTimeMark()` shall raise the error `CAN_E_PARAM_CONTROLLER` if the parameter `Controller` is out of range.] ()

8.4.5 Can_TTEnableTimeMarkIRQ

[TTCAN105] [

Service name:	Can_TTEnableTimeMarkIRQ
Syntax:	<code>void Can_TTEnableTimeMarkIRQ(uint8 Controller)</code>
Service ID[hex]:	0x4b
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	Controller Controller for which the time mark interrupt shall be enabled.
Parameters (inout):	None

Parameters (out):	None
Return value:	None
Description:	Enables the time mark interrupt.

] (BSW441005)

[TTCAN070] [If development error detection for the Ttcan module is enabled: The function Can_TTEnableTimeMarkIRQ() shall raise the error CAN_E_UNINIT if the driver is not yet initialized.] ()

[TTCAN071] [If development error detection for the Ttcan module is enabled: The function Can_TTEnableTimeMarkIRQ() shall raise the error CAN_E_PARAM_CONTROLLER if the parameter Controller is out of range.] ()

8.4.6 Can_TTDisableTimeMarkIRQ

[TTCAN106] [

Service name:	Can_TTDisableTimeMarkIRQ
Syntax:	void Can_TTDisableTimeMarkIRQ(uint8 Controller)
Service ID[hex]:	0x4c
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	Controller Controller for which the time mark interrupt shall be disabled.
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Disables the time mark interrupt.

] (BSW441005)

[TTCAN073] [If development error detection for the Ttcan module is enabled: The function Can_TTDisableTimeMarkIRQ() shall raise the error CAN_E_UNINIT if the driver is not yet initialized.] ()

[TTCAN074] [If development error detection for the Ttcan module is enabled: The function Can_TTDisableTimeMarkIRQ() shall raise the error CAN_E_PARAM_CONTROLLER if the parameter Controller is out of range.] ()

8.4.7 Can_TTGetTimeMarkIRQStatus

[TTCAN107] [

Service name:	Can_TTGetTimeMarkIRQStatus
Syntax:	void Can_TTGetTimeMarkIRQStatus(uint8 Controller, boolean* Can_TTIRQStatus)
Service ID[hex]:	0x4d
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	Controller Controller from which the status of the time mark IRQ shall be

		retrieved.
Parameters (inout):	None	
Parameters (out):	Can_TTIRQStatus	Address to store return value: True if the timer for the time mark is pending.
Return value:	None	
Description:	Gets the IRQ status of the time mark.	

] (BSW441005)

[TTCAN076] [If development error detection for the Ttcan module is enabled: The function Can_TTGetTimeMarkIRQStatus() shall raise the error CAN_E_UNINIT if the driver is not yet initialized.] ()

[TTCAN077] [If development error detection for the Ttcan module is enabled: The function Can_TTGetTimeMarkIRQStatus() shall raise the error CAN_E_PARAM_CONTROLLER if the parameter Controller is out of range.] ()

[TTCAN078] [If development error detection for the Ttcan module is enabled: The function Can_TTGetTimeMarkIRQStatus() shall raise the error CAN_E_PARAM_POINTER and shall return CAN_NOT_OK if the parameter Can_TTIRQStatus is a NULL pointer.] ()

8.4.8 Can_TTReceive

[TTCAN108] [

Service name:	Can_TTReceive	
Syntax:	<pre>void Can_TTReceive(uint8 Controller, uint8 Hrh, Can_IdType* CanId, uint8* CanDlc, uint8* CanSduPtr)</pre>	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Controller	Controller for which data shall be read out
	Hrh	Hardware receive handle of the hardware object, to read the received data from
Parameters (inout):	None	
Parameters (out):	CanId	Address to store return value: Can ID of the received frame
	CanDlc	Address to store return value: Length of the received frame
	CanSduPtr	Address to store return value: SDU of received frame
Return value:	None	
Description:	Reads received data from the controller by returning the pointer of the CanID, the DLC and the Data of the message in the requested HRH.	

] ()

[TTCAN110] [If development error detection for the Ttcan module is enabled: The function Can_TTReceive() shall raise the error CAN_E_UNINIT if the driver is not yet initialized.] ()

[TTCAN111] [If development error detection for the Ttcan module is enabled: The function Can_TTReceive() shall raise the error CAN_E_PARAM_CONTROLLER if the parameter Controller is out of range.] ()

[TTCAN112] [If development error detection for the Ttcan module is enabled: The function Can_TTReceive() shall raise the error CAN_E_PARAM_POINTER and shall return CAN_NOT_OK if one of the parameter CanId, CanDlc or CanSduPtr is a NULL pointer.] ()

8.5 Scheduled Functions

Additional TTCAN specific scheduled function definitions

8.5.1 Can_TTMainFunction_IRQ

[TTCAN113] [

Service name:	Can_TTMainFunction_IRQ
Syntax:	void Can_TTMainFunction_IRQ(void)
Service ID[hex]:	0x50
Timing:	FIXED_CYCLIC
Description:	Polls the interrupt flags specific to TTCAN

Note: The generic items from CAN Driver SWS regarding the main functions apply for Can_TTMainFunction_IRQ(), too.] ()

[TTCAN080] [If development error detection for the Ttcan module is enabled: The function Can_TTMainFunction_IRQ() shall raise the error CAN_E_UNINIT if the driver is not yet initialized.] ()

8.6 Expected Interfaces

8.6.1 Mandatory interfaces

Additional TTCAN specific mandatory interfaces

[TTCAN082] [

API function	Description
CanIf_TTApplWatchdogError	Reports an application watchdog error.
CanIf_TTGap	Reports the occurrence of a gap.
CanIf_TTMasterStateChange	Reports change of the master state between potential and current master.
CanIf_TTSevereError	Reports one of the following errors:

	- failed to serve appl. watchdog - config error - watch trigger reached
CanIf_TTStartOfCycle	Reports the start of a basic cycle.
CanIf_TTTimeDisc	Reports a time discontinuity.
CanIf_TTTimingError	Reports one of the following errors: - Change of error level - Tx overflow / underflow - Synchronization failed - Init watch trigger

] (BSW00387, BSW441008)

Hint: These additional mandatory interfaces for TTCAN shall serve the interrupts that may occur during time triggered operation as described in [12].

9 Sequence diagrams

9.1 Interaction between Ttcan and TtcanIf module

For sequence diagrams see the TTCAN Interface specification [6] and CAN Interface specification [4]. There are described the complete sequences for Transmission, Reception and Error Handling.

9.2 Wakeup sequence

For Wakeup sequence diagrams refer to specification of ECU State Manager [8].

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 Ttcan module.

Chapter 10.3 specifies published information of the Ttcan module.

10.1 How to read this chapter

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture [1]
- AUTOSAR ECU Configuration Specification [7]
This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Variant

Variants describe sets of configuration parameters. E.g., variant 1: only pre-compile time configuration parameters; variant 2: mix of pre-compile- and post build time-configuration parameters. In one variant a parameter can only be of one configuration class.

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.

- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

10.1.4 Specification template for configuration parameters

The following tables consist of three sections:

- the general section
- the configuration parameter section
- the section of included/referenced containers

Pre-compile time - specifies whether the configuration parameter shall be of configuration class *Pre-compile time* or not

Label	Description
x	The configuration parameter shall be of configuration class <i>Pre-compile time</i> .
--	The configuration parameter shall never be of configuration class <i>Pre-compile time</i> .

Link time - specifies whether the configuration parameter shall be of configuration class *Link time* or not

Label	Description
x	The configuration parameter shall be of configuration class <i>Link time</i> .
--	The configuration parameter shall never be of configuration class <i>Link time</i> .

Post Build - specifies whether the configuration parameter shall be of configuration class *Post Build* or not

Label	Description
x	The configuration parameter shall be of configuration class <i>Post Build</i> and no specific implementation is required.
L	<i>Loadable</i> - the configuration parameter shall be of configuration class <i>Post Build</i> and only one configuration parameter set resides in the ECU.
M	<i>Multiple</i> - the configuration parameter shall be of configuration class <i>Post Build</i> and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class <i>Post Build</i> .

10.2 Containers and configuration parameters

Additional TTCAN specific configuration parameters

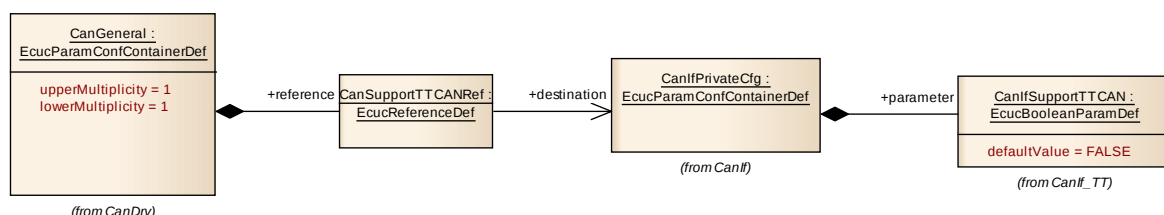


Figure 10-1: CAN Driver Time Triggered General Configuration

The reference CanSupportTTCANRef is described in Specification of CAN Driver [5], SWS Item Id CAN430_Conf.

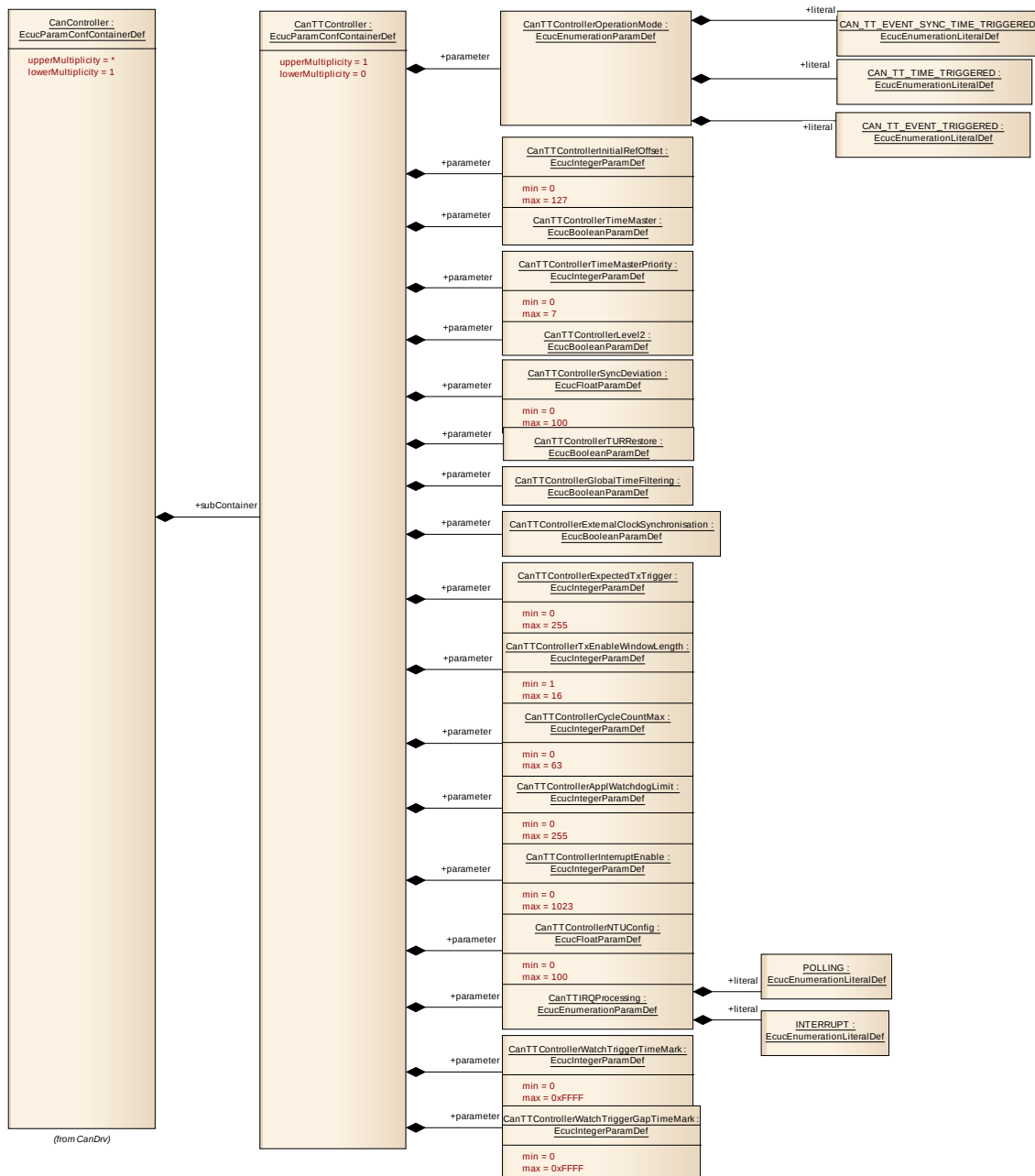


Figure 10-2: CAN Driver Time Triggered Controller Configuration

10.2.1 CanTTController

SWS Item	TTCAN001_Conf :
Container Name	CanTTController
Description	This container is only included and valid if TTCAN SWS is used and TTCAN is enabled. This container contains the configuration parameters of the TTCAN controller(s) (which are needed in addition to the configuration parameters of the CAN controller(s)). CanTTController is only included, if the controller supports TTCAN.
Configuration Parameters	

SWS Item	TTCAN139_Conf :		
Name	CanTTControllerApplWatchdogLimit		
Description	Defines the maximum time period (unit is 256 times NTU) after which the application has to serve the watchdog.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	TTCAN138_Conf :		
Name	CanTTControllerCycleCountMax		
Description	Defines the value for cycle_count_max. Allowed values: 0x00: 1 basic cycle 0x01: 2 basic cycles 0x03: 4 basic cycles 0x07: 8 basic cycles 0x0F: 16 basic cycles 0x1F: 32 basic cycles 0x3F: 64 basic cycles		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 63		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	TTCAN136_Conf :		
Name	CanTTControllerExpectedTxTrigger		
Description	Number of expected_tx_trigger.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	TTCAN135_Conf :		
Name	CanTTControllerExternalClockSynchronisation		
Description	Enables/disables the external clock synchronization. TRUE: External clock synchronization enabled. FALSE: External clock synchronization disabled. This parameter shall only be configurable if parameter CanTTControllerLevel2 equals TRUE.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	dependency: CanTTControllerLevel2 (TTCAN131_Conf)		

SWS Item	TTCAN134_Conf :		
Name	CanTTControllerGlobalTimeFiltering		
Description	Enables/disables the global time filtering. TRUE: Global time filtering enabled. FALSE: Global time filtering disabled. This parameter shall only be configurable if parameter CanTTControllerLevel2 equals TRUE.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	dependency: CanTTControllerLevel2 (TTCAN131_Conf)		

SWS Item	TTCAN128_Conf :		
Name	CanTTControllerInitialRefOffset		
Description	Defines the initial value for ref trigger offset.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 127		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	TTCAN140_Conf :		
Name	CanTTControllerInterruptEnable		
Description	Enables/disables the respective interrupts. Bit Position set to 1: Enable respective interrupt. Bit Position set to 0: Disable respective interrupt. Bit Position / Interrupt Source: 10: Application Watchdog. 9: Watch Trigger reached. 8: Initialization Watchdog. 7: Change of Error Level. 6: Tx Overflow. 5: Tx Underflow. 4: Global Time Error. 3: Gap. 2: Start of Cycle. 1: Time Discontinuity. 0: Master State Change. Bit position "1: Time Discontinuity" and "4: Global Time Error" shall only be configurable if parameter CanTTControllerLevel2 equals TRUE.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 1023		
Default value	--		

ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	dependency: CanTTControllerLevel2 (TTCAN131_Conf)		

SWS Item	TTCAN131_Conf :		
Name	CanTTControllerLevel2		
Description	Defines whether Level 2 or Level 1 is used. TRUE: Level 2. FALSE: Level 1. If this parameter is set to FALSE then all parameters with dependency to CanTTControllerLevel2 need not be configured.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	TTCAN141_Conf :		
Name	CanTTControllerNTUConfig		
Description	Defines the config value for NTU (network time unit). Value given in microseconds. The value configured shall be greater than 0. Together with the local oscillator period, the TUR (time unit ratio) can be derived from the NTU. This parameter shall only be configurable if parameter CanTTControllerLevel2 equals TRUE.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 100		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	dependency: CanTTControllerLevel2 (TTCAN131_Conf)		

SWS Item	TTCAN127_Conf :		
Name	CanTTControllerOperationMode		
Description	Defines the operation mode.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CAN_TT_EVENT_SYNC_TIME_TRIGGERED	Event-synchronized time triggered operation	
	CAN_TT_EVENT_TRIGGERED	Event triggered operation (normal can operation without time schedule)	
	CAN_TT_TIME_TRIGGERED	Time triggered operation	
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-

		POST-BUILD
Scope / Dependency		

SWS Item	TTCAN132_Conf :		
Name	CanTTControllerSyncDeviation		
Description	Defines the maximum synchronization deviation: Given as a percentage value of the NTU (network time unit). The value configured shall be greater than 0. This parameter shall only be configurable if parameter CanTTControllerLevel2 equals TRUE.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 100		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	dependency: CanTTControllerLevel2 (TTCAN131_Conf)		

SWS Item	TTCAN133_Conf :		
Name	CanTTControllerTURRestore		
Description	Enables/disables the TUR restore. Note that the value configured for TUR can be derived from the value configured for NTU and the local oscillator preperiod. TRUE: TUR restore enabled. FALSE: TUR restore disabled. This parameter shall only be configurable if parameter CanTTControllerLevel2 equals TRUE.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	dependency: CanTTControllerLevel2 (TTCAN131_Conf)		

SWS Item	TTCAN129_Conf :		
Name	CanTTControllerTimeMaster		
Description	Defines whether the controller acts as a potential time master. TRUE: Potential time master. FALSE: Time slave.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	TTCAN130_Conf :		
Name	CanTTControllerTimeMasterPriority		
Description	Defines the time master priority.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	

	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	TTCAN137_Conf :		
Name	CanTTControllerTxEnableWindowLength		
Description	Length of the tx enable window given in CAN bit times. Definition parameter "CanTTControllerTxEnableWindowlength" is used such that: Length of enable window = CanTTControllerTxEnableWindowLength + 1		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 16		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	TTCAN158_Conf :		
Name	CanTTControllerWatchTriggerGapTimeMark		
Description	watch trigger time mark after a gap		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	TTCAN157_Conf :		
Name	CanTTControllerWatchTriggerTimeMark		
Description	watch trigger time mark		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	TTCAN142_Conf :		
Name	CanTTIRQProcessing		
Description	Enables / disables API Can_MainFunction_BusOff() for handling busoff events in polling mode.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	INTERRUPT	Interrupt Mode of operation.	
	POLLING	Polling Mode of operation.	
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

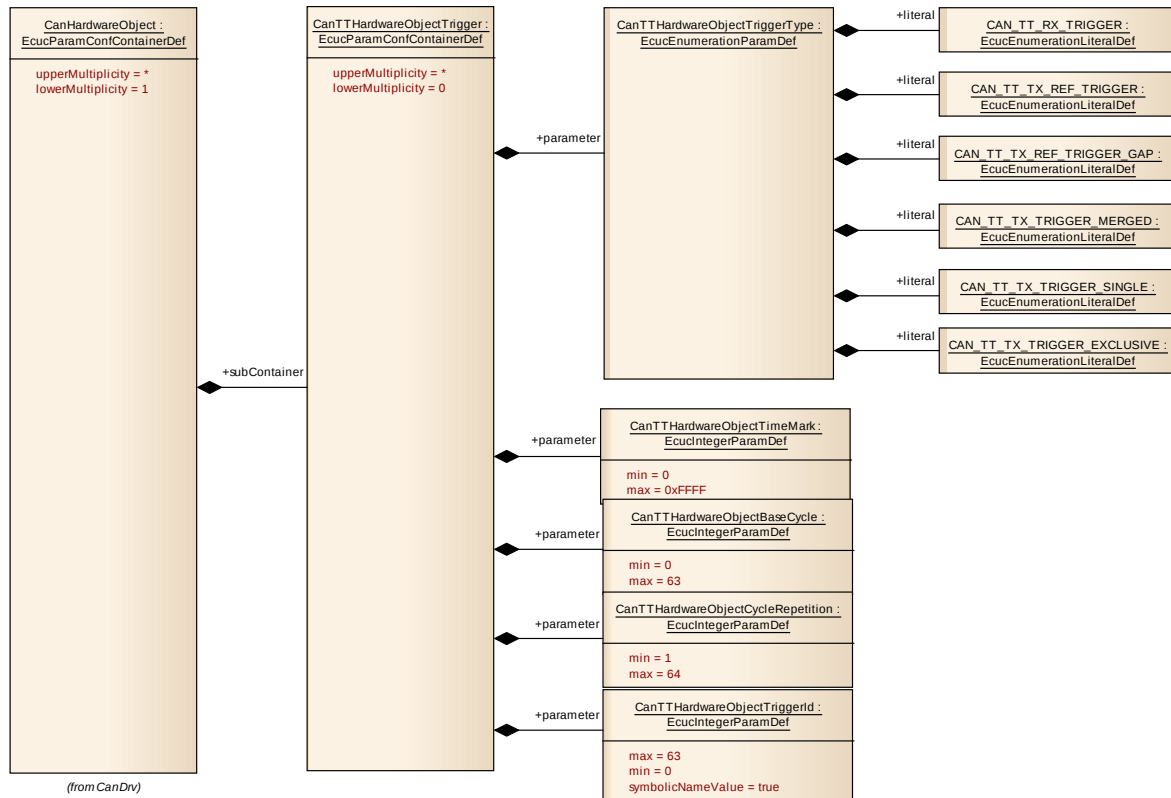


Figure 10-3: CAN Driver Time Triggered Hardware Object Configuration

10.2.2 CanTTHardwareObjectTrigger

SWS Item	TTCAN002_Conf :
Container Name	CanTTHardwareObjectTrigger
Description	This container is only included and valid if TTCAN SWS is used and TTCAN is enabled. This container contains the configuration (parameters) of TTCAN triggers for Hardware Objects, which are additional to the configuration (parameters) of CAN Hardware Objects. CanTTHardwareObjectTrigger is only included, if the controller supports TTCAN.
Configuration Parameters	

SWS Item	TTCAN147_Conf :
Name	CanTTHardwareObjectBaseCycle
Description	Defines the cycle_offset. CanTTHardwareObjectBaseCycle must be not greater than cycle_count_max.
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 63

Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	TTCAN148_Conf :		
Name	CanTTHardwareObjectCycleRepetition		
Description	Defines the repeat_factor. CanTTHardwareObjectCycleRepetition shall be a power of two (2), greater than cycle_offset but not greater than cycle_count_max + 1.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 64		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	TTCAN146_Conf :		
Name	CanTTHardwareObjectTimeMark		
Description	Defines the point in time, when the trigger will be activated. Value is given in cycle time.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	TTCAN155_Conf :		
Name	CanTTHardwareObjectTriggerId		
Description	Sequential number which allows separation of different TTCAN triggers configured for one and the same hardware object.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 63		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	TTCAN145_Conf :		
Name	CanTTHardwareObjectTriggerType		
Description	Defines the type of the trigger associated with the hardware object. This parameter depends on plain CAN parameter CAN_OBJECT_TYPE. If CAN_OBJECT_TYPE equals RECEIVE than this parameter is fixed to CAN_TT_RX_TRIGGER. If CAN_OBJECT_TYPE equals TRANSMIT than one of the following literals is configurable: CAN_TT_TX_REF_TRIGGER, CAN_TT_TX_REF_TRIGGER_GAP, CAN_TT_TX_TRIGGER_MERGED,		

	CAN_TT_TX_TRIGGER_SINGLE, CAN_TT_TX_TRIGGER_EXCLUSIVE.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CAN_TT_RX_TRIGGER	Trigger for verifying the successful reception of messages.	
	CAN_TT_TX_REF_TRIGGER	Trigger for transmitting the reference message.	
	CAN_TT_TX_REF_TRIGGER_GAP	Trigger for transmitting the reference message in case no event occurs after a gap.	
	CAN_TT_TX_TRIGGER_EXCLUSIVE	Trigger for transmitting a message in an exclusive time window. Note, that messages in an exclusive window are transmitted continuously, i.e. regardless whether the same message has been transmitted before, the message, which is currently available, will be transmitted every time the tx trigger occurs.	
	CAN_TT_TX_TRIGGER_MERGED	Trigger for transmitting a message inside a merged arbitration window (the last tx trigger in a merged arbitration window is of type CAN_TT_TX_TRIGGER_SINGLE). Note, that messages in an arbitration window are transmitted only, if new data is available. When the transmission was not successful, it will be repeated at the next tx trigger for this message. When the transmission was successful, this message will not be transmitted again at the next tx triggers until a new message for this tx trigger is provided.	
	CAN_TT_TX_TRIGGER_SINGLE	Trigger for transmitting a message in a single (non-merged) arbitration window (or the last tx trigger in a merged arbitration window). Note, that messages in an arbitration window are transmitted only, if new data is available. When the transmission was not successful, it will be repeated at the next tx trigger for this message. When the transmission was successful, this message will not be transmitted again at the next tx triggers until a new message for this tx trigger is provided.	
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	dependency: CAN_OBJECT_TYPE		
No Included Containers			

10.3 Published information

[TTCAN725] [The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules [2] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [11].] ()

Additional module-specific published parameters are listed below if applicable.

11 Not applicable requirements

[TTCAN726] [These requirements are not applicable to this specification.] ()