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

Document Status	Final
Part of AUTOSAR Release	4.2.2

Document Change History

Release	Changed by	Description
4.2.2	AUTOSAR Release Management	• Fixed error section
4.2.1	AUTOSAR Release Management	• Updated disclaimer • Editorial changes
4.1.3	AUTOSAR Release Management	• Updated description of TTCAN EcuC containers • Editorial changes
4.1.2	AUTOSAR Release Management	• Removed timing in [SWS_TtCan_00113] • Editorial changes
4.1.1	AUTOSAR Administration	• Updated scope of parameters • Formal update for traceability analysis • Aligned to General Documents
4.0.3	AUTOSAR Administration	• Provided min/max values of configuration parameters • New tracebility matrix
4.0.1	AUTOSAR Administration	Updated artifacts of configuration section
3.1.4	AUTOSAR Administration	Initial Release

Disclaimer

Advice for users

AUTOSAR specifications 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 specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, 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 Contents

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 [1]. 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 SRS_SPAL_12092).

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.

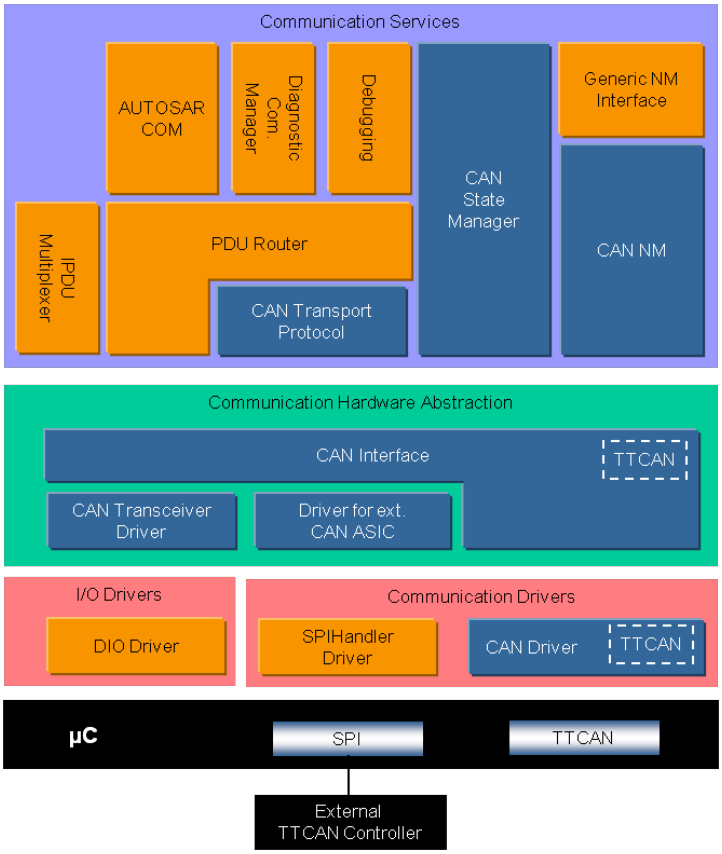

Figure 1.1: AUTOSAR TTCAN Layer Model (see [2])

The Ttcan module provides services for initiating transmissions and calls the call-back 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

The glossary below includes acronyms and abbreviations relevant to the TTCAN Driver module that are not included in the [3, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
Arbitrating Time Window	See ISO 11898-4 [1]
Basic Cycle	See ISO 11898-4 [1]
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 [1]
DLC	Data Length Code (part of `L-PDU` that describes the SDU length)
Cycle Time	See ISO 11898-4 [1]
Exclusive Time Window	See ISO 11898-4 [1]
Global Time	See ISO 11898-4 [1]
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 [1]
Matrix Cycle	See ISO 11898-4 [1]
MCAL	Microcontroller Abstraction Layer
NTU	See ISO 11898-4 [1]
Reference Message	See ISO 11898-4 [1]
Single Shot	A message will be transmitted only once contrary to `Continuous Transmission`.
System Matrix	See ISO 11898-4 [1]
Time Gap	See ISO 11898-4 [1]
Time Master	See ISO 11898-4 [1]
Time Window	See ISO 11898-4 [1]
Transmission Column	See ISO 11898-4 [1]
Transmit Trigger Event	See ISO 11898-4 [1]
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 [1]

3 Related documentation

All documents of the referenced CAN Driver document [4] are also valid for this document.

3.1 Input documents & related standards and norms

Bibliography

[1] ISO 11898-4:2004 - Road vehicles - Controller area network (CAN) - Part 4: Time-triggered communication

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture

[3] Glossary
AUTOSAR_TR_Glossary

[4] Specification of CAN Driver
AUTOSAR_SWS_CANDriver

[5] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral

[6] Specification of CAN Transceiver Driver
AUTOSAR_SWS_CANTransceiverDriver

[7] Specification of TTCAN Interface
AUTOSAR_SWS_TTCANInterface

[8] Specification of Watchdog Driver
AUTOSAR_SWS_WatchdogDriver

[9] Specification of CAN Interface
AUTOSAR_SWS_CANInterface

[10] Specification of ECU State Manager
AUTOSAR_SWS_ECUStateManager

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [5, SWS BSW General], which is also valid for TTCAN Driver.

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

4 Constraints and assumptions

The constraints and assumptions of the Ttcan module are the same as for the CAN Driver module [6].

5 Dependencies to other modules

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

The dependencies described in the referenced CAN Driver module [6] 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 Table 8.5.1).

6 Requirements Tracing

Requirement	Description	Satisfied by
[SRS_BSW_00337]	Classification of development errors	[SWS_TtCan_00010]
[SRS_BSW_00387]	No description	[SWS_TtCan_00082]
[SRS_TtCan_41003]	The Hardware Object Handles shall be mappable to all types of time windows defined in ISO 11898 by configuration.	[SWS_TtCan_00156]
[SRS_TtCan_41005]	The CAN Driver with TTCAN functionality shall provide means for influencing timing parameters and providing information from the TTCAN controller for synchronization purposes.	[SWS_TtCan_00004] [SWS_TtCan_00005] [SWS_TtCan_00006] [SWS_TtCan_00091] [SWS_TtCan_00092] [SWS_TtCan_00093] [SWS_TtCan_00094] [SWS_TtCan_00095] [SWS_TtCan_00096] [SWS_TtCan_00097] [SWS_TtCan_00098] [SWS_TtCan_00099] [SWS_TtCan_00101] [SWS_TtCan_00102] [SWS_TtCan_00103] [SWS_TtCan_00104] [SWS_TtCan_00105] [SWS_TtCan_00106] [SWS_TtCan_00107]
[SRS_TtCan_41006]	The CAN Driver with TTCAN functionality shall support the event synchronized time-triggered communication.	[SWS_TtCan_00007] [SWS_TtCan_00094] [SWS_TtCan_00095]
[SRS_TtCan_41007]	The CAN Driver with TTCAN functionality shall indicate occurred events according to chapter 10.2.2 "Interrupt_Status_Vector" of ISO 11898-4:2004.	[SWS_TtCan_00009] [SWS_TtCan_00124] [SWS_TtCan_00126]
[SRS_TtCan_41008]	The CAN Driver with enabled TTCAN functionality shall provide a notification for severe error (S3).	[SWS_TtCan_00082] [SWS_TtCan_00120] [SWS_TtCan_00126]
[SRS_TtCan_41009]	The CAN Driver with TTCAN functionality shall not recover from severe error (S3) automatically.	[SWS_TtCan_00121] [SWS_TtCan_00122] [SWS_TtCan_00123]

7 Functional specification

The following section only describes additional TTCAN specific 'Functional specifications'. The Specification of CAN Driver [4] 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 [7] and the TTCAN ISO Specification [1].

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 [4].

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 [1, ISO 11898-4].

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):

[SWS_TtCan_00155] ⌈ Replaces *SWS_Can_00262*: The function Can_SetControllerMode(CAN shall wait for a limited time until the TTCAN Controller has changed to the state SYNCHRONIZING (Compare to *SWS_Can_00371*). ⌋*()*

Rational for [SWS_TtCan_00155]: 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 *SWS_Can_00371*) and the function Can_Mainfunction_Timeout (compare to SWS_Can_00372).

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

[SWS_TtCan_00120] ⌈

- STARTED -> STOPPED
- Triggered by hardware if the TTCAN Controller reaches error level S3 (see TTCAN ISO Specification [1])
- The CanIf module is notified with the function CanIf_TTSevereError after STOPPED state is reached.

⌋*(SRS_TtCan_41008)*

[SWS_TtCan_00121] ⌈ 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. ⌋*(SRS_TtCan_41009)*

[SWS_TtCan_00122] ⌈ After severe error detection, TtcanDrv shall cancel still pending messages without raising a cancellation notification. ⌋*(SRS_TtCan_41009)*

[SWS_TtCan_00123] ⌈ TtcanDrv shall disable or suppress automatic severe error recovery. ⌋*(SRS_TtCan_41009)*

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.

[SWS_TtCan_00156] ⌈ It shall be possible to map all transmit message objects to specific Time Windows (see TTCAN ISO Specification [1]) by configuration (see *TTCANIF145_Conf*, *TTCANIF146_Conf*, *TTCANIF147_Conf*, *TTCANIF148_Conf*). ⌋*(SRS_TtCan_41003)*

7.2.1 Priority Inversion

[SWS_TtCan_00154] ⌈ Multiplexed transmission and transmit cancellation described in the Specification of CAN Driver [4] 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 *ECUC_Can_00145*).

A detailed description of reception triggering and the verification of message reception can be found in [1, ISO 11898-4].

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, TtcanDrv needs to support maintaining the timing parameters and the master-controlled synchronization mechanisms.

[SWS_TtCan_00004] ⌈ TtcanDrv shall provide information from the TTCAN Controller about the timing parameters (see [SWS_TtCan_00090]), the synchronization state and the master state (see [SWS_TtCan_00091]). ⌋*(SRS_TtCan_41005)*

[SWS_TtCan_00005] ⌈ TtcanDrv shall provide means to influence the timing parameters of a TTCAN Controller (see [SWS_TtCan_00096], [SWS_TtCan_00097], [SWS_TtCan_00098], [SWS_TtCan_00099]) during runtime, if the TTCAN Controller acts as Time Master. ⌋*(SRS_TtCan_41005)*

[SWS_TtCan_00006] ⌈ TtcanDrv shall provide the functionality of a timer, which is based on the time marks of the communication system, provided by the TTCAN Controller. ⌋*(SRS_TtCan_41005)*

7.4.1 Event Synchronization

[SWS_TtCan_00007] ⌈ TtcanDrv shall support event-synchronized communication (see [SWS_TtCan_00094], [SWS_TtCan_00095]) (refer to [1, ISO 11898-4]). ⌋*(SRS_TtCan_41006)*

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 TtcanIf:

[SWS_TtCan_00009] ⌈

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	TtcanIf_TimingError
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 section 8.6) ⌋*(SRS_TtCan_41007)*

7.6 Application Watchdog

Note: The TTCAN Application Watchdog shall be served by using a Watchdog Driver instance (see [8, Wachtdog 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 *ECUC_Can_00139*).

7.7 TTCAN error handling

This chapter corresponds to the chapter "'Error handling"' of the CAN Driver SWS [4].

[SWS_TtCan_00124] ⌈ 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 [1]) are reached. ⌋*(SRS_TtCan_41007)*

[SWS_TtCan_00126] ⌈ 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 [1]) is reached. ⌋(*SRS_TtCan_41007*, *SRS_TtCan_41008*)

7.8 Error Classification

7.8.1 Development Errors

[SWS_TtCan_00010] ⌈ The following errors and exceptions are specific to TTCAN

Type of error	Relevance	Related error code	Value [hex]
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_CONSEQUTIVE_DISC`	`0x0a`
Adjustment of global time fails, because external synchronization has been disabled during configuration	Development	`CAN_TT_E_SYNC_DISABLED`	`0x0b`

⌋(*SRS_BSW_00337*)

7.8.2 Runtime Errors

There are no runtime errors.

7.8.3 Transient Faults

There are no transient faults.

7.8.4 Production Errors

There are no production errors.

7.8.5 Extended Production Errors

There are no extended production errors.

8 API specification

Since the `Ttcan module` is an extension of the CAN Driver module [4], 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

[SWS_TtCan_00125] ⌈

Module	*Imported Type*
CanIf	CanIf_TTMasterStateType CanIf_TTSevereErrorEnumType CanIf_TTTimingErrorIRQType
Can_GeneralTypes	Can_IdType
Std_Types	Std_ReturnType

Table 8.1: Can_ImportedTypes

⌋()

8.2 Type definitions

Additional TTCAN specific type definitions

8.2.1 Can_TTTimeType

[SWS_TtCan_00084] ⌈

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

Table 8.2: Can_TTTimeType

⌋()

8.2.2 Can_TTMasterSlaveModeType

[SWS_TtCan_00115] ⌈

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

Table 8.3: Can_TTMasterSlaveModeType

⌋()

8.2.3 Can_TTSyncModeEnumType

[SWS_TtCan_00116] ⌈

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

Table 8.4: Can_TTSyncModeEnumType

⌋()

8.2.4 Can_TTMasterStateType

[SWS_TtCan_00085] ⌈

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		

Table 8.5: Can_TTMasterStateType

⌋()

8.2.5 Can_TTErrorLevelEnumType

[SWS_TtCan_00117] ⌈

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

Table 8.6: Can_TTErrorLevelEnumType

⌋()

8.2.6 Can_TTErrorLevelType

[SWS_TtCan_00086] ⌈

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

Table 8.7: Can_TTErrorLevelType

⌋()

8.2.7 Can_TTTimeSourceType

[SWS_TtCan_00088] ⌈

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

Table 8.8: Can_TTTimeSourceType

⌋()

8.3 Function definitions

Additional TTCAN specific function definitions

8.3.1 Can_TTGetControllerTime

[SWS_TtCan_00090] ⌈

Service name:	Can_TTGetControllerTime	
Syntax:	`void Can_TTGetControllerTime( uint8 Controller, Can_TTTimeType* Can_TTGlobalTime, Can_TTTimeType* Can_TTLocalTime, Can_TTTimeType* Can_TTCycleTime, uint8* Can_TTCycleCount )`	
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 Can_TTLocalTime Can_TTCycleTime Can_TTCycleCount	Address to store return value: Global time Address to store return value: Local time Address to store return value: Cycle time 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	

Table 8.9: Can_TTGetControllerTime

⌋()

[SWS_TtCan_00012] ⌈ 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. ⌋()

[SWS_TtCan_00013] ⌈ If development error detection for the `Ttcan module` is enabled: The function `Can_TTGetControllerTime()` shall raise the error `CAN_E_PARAM_CONTROL` if the parameter Controller is out of range. ⌋()

[SWS_TtCan_00014] ⌈ 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_TTCycelCount` is a `NULL` pointer. ⌋()

8.3.2 Can_TTGetMasterState

[SWS_TtCan_00091] ⌈

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.	

Table 8.10: Can_TTGetMasterState

⌋(*SRS_TtCan_41005*)

[SWS_TtCan_00016] ⌈ 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. ⌋()

[SWS_TtCan_00017] ⌈ 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. ⌋()

[SWS_TtCan_00018] ⌈ 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

[SWS_TtCan_00092] ⌈

Service name:	Can_TTGetNTUActual
Syntax:	`void Can_TTGetNTUActual( uint8 Controller, Can_TTTURType* Can_TTTURAct )`
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.	

Table 8.11: Can_TTGetNTUActual

⌋(*SRS_TtCan_41005*)

[SWS_TtCan_00020] ⌈ 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. ⌋*()*

[SWS_TtCan_00021] ⌈ 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. ⌋*()*

[SWS_TtCan_00022] ⌈ 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

[SWS_TtCan_00093] ⌈

Service name:	Can_TTGetErrorLevel	

Syntax:	`void Can_TTGetErrorLevel(`
	`uint8 Controller,`
	`Can_TTErrorLevelType* Can_TTErrorLevel`
	`)`	

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.	

Table 8.12: Can_TTGetErrorLevel

⌋(*SRS_TtCan_41005*)

[SWS_TtCan_00024] ⌈ 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. ⌋*()*

[SWS_TtCan_00025] ⌈ 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. ⌋*()*

[SWS_TtCan_00026] ⌈ 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

[SWS_TtCan_00094] ⌈

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.	

Table 8.13: Can_TTSetNextIsGap

⌋*(SRS_TtCan_41005, SRS_TtCan_41006)*

[SWS_TtCan_00028] ⌈ 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. ⌋*()*

[SWS_TtCan_00029] ⌈ 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

[SWS_TtCan_00095] ⌈

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.	

Table 8.14: Can_TTSetEndOfGap

⌋(*SRS_TtCan_41005*, *SRS_TtCan_41006*)

[SWS_TtCan_00031] ⌈ The function `Can_TTSetEndOfGap()` shall only take effect if the `TTCAN Controller` is a potentional `Time Master`. ⌋()

[SWS_TtCan_00032] ⌈ 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 potentional `Time Master`. ⌋()

[SWS_TtCan_00033] ⌈ 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. ⌋()

[SWS_TtCan_00034] ⌈ 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

[SWS_TtCan_00096] ⌈

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"	

Table 8.15: Can_TTSetTimeCommand

⌋(*SRS_TtCan_41005*)

[SWS_TtCan_00036] ⌈ If development error detection for the `Ttcan module` is enabled: The function `Can_TTSetTimeCommand()` shall raise the error `CAN_TT_E_CONSEQUTIVE_D` if two consecutive reference messages are transmitted wich both have the "'Disc_bit"' set. ⌋()

[SWS_TtCan_00037] ⌈ 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. ⌋()

[SWS_TtCan_00038] ⌈ The function `Can_TTSetTimeCommand()` shall only take effect if the `TTCAN Controller` is the current `Time Master`. ⌋()

[SWS_TtCan_00039] ⌈ If development error detection for the `Ttcan module` is enabled: The function `Can_TTSetTimeCommand()` shall raise the error `CAN_TT_E_NOT_CURRENT_M` if the `TTCAN Controller` is not the current `Time Master`. ⌋()

[SWS_TtCan_00040] ⌈ 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. ⌋()

[SWS_TtCan_00041] ⌈ 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

[SWS_TtCan_00097] ⌈

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

Table 8.16: Can_TTGlobalTimePreset

⌋(*SRS_TtCan_41005*)

[SWS_TtCan_00043] ⌈ 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. ⌋()

[SWS_TtCan_00044] ⌈ If development error detection for the Ttcan module is enabled: The function Can_TTGlobalTimePreset() shall raise the error CAN_E_PARAM_CONTROLL if the parameter Controller is out of range. ⌋()

8.3.9 Can_TTSetExtClockSyncCommand

[SWS_TtCan_00098] ⌈

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".	

Table 8.17: Can_TTSetExtClockSyncCommand

⌋(*SRS_TtCan_41005*)

[SWS_TtCan_00046] ⌈ The function Can_TTSetExtClockSyncCommand() shall only take effect if the TTCAN Controller is the current Time Master. ⌋()

[SWS_TtCan_00047] ⌈ If development error detection for the Ttcan module is enabled: The function Can_TTSetExtClockSyncCommand() shall raise the error CAN_TT_E_NOT_C if the TTCAN Controller is not the current Time Master. ⌋()

[SWS_TtCan_00048] ⌈ 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. ⌋()

[SWS_TtCan_00049] ⌈ If development error detection for the Ttcan module is enabled: The function Can_TTSetExtClockSyncCommand() shall raise the error CAN_E_PARAM_CO if the parameter Controller is out of range. ⌋()

8.3.10 Can_TTSetNTUAdjust

[SWS_TtCan_00099] ⌈

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 Can_TTTURAdjust	Controller for which the "NTU adjust" shall be set New value for "NTU adjust" Value is given in microseconds.
Parameters (inout):	None	
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".	

Table 8.18: Can_TTSetNTUAdjust

⌋(*SRS_TtCan_41005*)

[SWS_TtCan_00051] ⌈ 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. ⌋()

[SWS_TtCan_00052] ⌈ 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

[SWS_TtCan_00101] ⌈

Service name:	Can_TTGetSyncQuality
Syntax:	`void Can_TTGetSyncQuality( uint8 Controller, boolean* Can_TTClockSpeed, boolean* Can_TTGlobalTimePhase )`
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.	

Table 8.19: Can_TTGetSyncQuality

⌋(*SRS_TtCan_41005*)

[SWS_TtCan_00057] ⌈ 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. ⌋*()*

[SWS_TtCan_00058] ⌈ 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. ⌋*()*

[SWS_TtCan_00059] ⌈ 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

[SWS_TtCan_00102] ⌈

Service name:	Can_TTSetTimeMark	
Syntax:	`void Can_TTSetTimeMark( uint8 Controller, Can_TTTimeType Can_TTTimeMark, Can_TTTimeSourceType Can_TTTimeSource )`	
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.	

Table 8.20: Can_TTSetTimeMark

⌋(*SRS_TtCan_41005*)

[SWS_TtCan_00061] ⌈ 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. ⌋*()*

[SWS_TtCan_00062] ⌈ 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

[SWS_TtCan_00103] ⌈

Service name:	Can_TTCancelTimeMark	
Syntax:	`void Can_TTCancelTimeMark( uint8 Controller )`	
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.	

Table 8.21: Can_TTCancelTimeMark

⌋(*SRS_TtCan_41005*)

[SWS_TtCan_00064] ⌈ 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. ⌋*()*

[SWS_TtCan_00065] ⌈ 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

[SWS_TtCan_00104] ⌈

Service name:	Can_TTAckTimeMark

Syntax:	`void Can_TTAckTimeMark( uint8 Controller )`	
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.	

Table 8.22: Can_TTAckTimeMark

⌋(*SRS_TtCan_41005*)

[SWS_TtCan_00067] ⌈ 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. ⌋()

[SWS_TtCan_00068] ⌈ 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

[SWS_TtCan_00105] ⌈

Service name:	Can_TTEnableTimeMarkIRQ	
Syntax:	`void Can_TTEnableTimeMarkIRQ( uint8 Controller )`	
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.	

Table 8.23: Can_TTEnableTimeMarkIRQ

⌋(*SRS_TtCan_41005*)

[SWS_TtCan_00070] ⌈ 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. ⌋*()*

[SWS_TtCan_00071] ⌈ If development error detection for the `Ttcan module` is enabled: The function `Can_TTEnableTimeMarkIRQ()` shall raise the error `CAN_E_PARAM_CONTROL` if the parameter Controller is out of range. ⌋*()*

8.4.6 Can_TTDisableTimeMarkIRQ

[SWS_TtCan_00106] ⌈

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.	

Table 8.24: Can_TTDisableTimeMarkIRQ

⌋*(SRS_TtCan_41005)*

[SWS_TtCan_00073] ⌈ 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. ⌋*()*

[SWS_TtCan_00074] ⌈ If development error detection for the `Ttcan module` is enabled: The function `Can_TTDisableTimeMarkIRQ()` shall raise the error `CAN_E_PARAM_CONTRO` if the parameter Controller is out of range. ⌋*()*

8.4.7 Can_TTGetTimeMarkIRQStatus

[SWS_TtCan_00107] ⌈

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.	

Table 8.25: Can_TTGetTimeMarkIRQStatus

⌋(*SRS_TtCan_41005*)

[SWS_TtCan_00076] ⌈ 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. ⌋()

[SWS_TtCan_00077] ⌈ If development error detection for the `Ttcan module` is enabled: The function `Can_TTGetTimeMarkIRQStatus()` shall raise the error `CAN_E_PARAM_CONT` if the parameter Controller is out of range. ⌋()

[SWS_TtCan_00078] ⌈ If development error detection for the `Ttcan module` is enabled: The function `Can_TTGetTimeMarkIRQStatus()` shall raise the error `CAN_E_PARAM_POIN` and shall return `CAN_NOT_OK` if the parameter `Can_TT IRQStatus` is a `NULL` pointer. ⌋()

8.4.8 Can_TTReceive

[SWS_TtCan_00108] ⌈

Service name:	Can_TTReceive	
Syntax:	`void Can_TTReceive( uint8 Controller, uint8 Hrh, Can_IdType* CanId, uint8* CanDlc, uint8* CanSduPtr )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Controller Hrh	Controller for which data shall be read out Hardware receive handle of the hardware object, to read the received data from
Parameters (inout):	None	
Parameters (out):	CanId CanDlc CanSduPtr	Address to store return value: Can ID of the received frame Address to store return value: Length of the received frame 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.

Table 8.26: Can_TTReceive

⌋()

[SWS_TtCan_00110] ⌈ 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. ⌋()

[SWS_TtCan_00111] ⌈ 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. ⌋()

[SWS_TtCan_00112] ⌈ 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

[SWS_TtCan_00113] ⌈

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

Table 8.27: Can_TTMainFunction_IRQ

⌋()

Note: The generic items from CAN Driver SWS [4] regarding the main functions apply for Can_TTMainFunction_IRQ(), too.

[SWS_TtCan_00080] ⌈ 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

[SWS_TtCan_00082] ⌈

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

Table 8.28: Ttcan Mandatory Interfaces

⌋*(SRS_BSW_00387, SRS_TtCan_41008)*

Hint: These additional mandatory interfaces for TTCAN shall serve the interrupts that may occur during time triggered operation as described in [1, ISO 11898-4].

9 Sequence diagrams

9.1 Interaction between Ttcan and TtcanIf module

For sequence diagrams see the TTCAN Interface specification [7] and CAN Interface specification [9]. 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 [10].

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification chapter 10 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave chapter 10 in the specification to guarantee comprehension.

section 10.1 specifies the structure (containers) and the parameters of the `Ttcan module`.

Figure 10.2 specifies published information of the `Ttcan module`.

10.1 How to read this chapter

For details refer to the [5, chapter 10.1 "Introduction to configuration specification" in SWS_BSWGeneral]

10.2 Containers and configuration parameters

Additional TTCAN specific configuration parameters

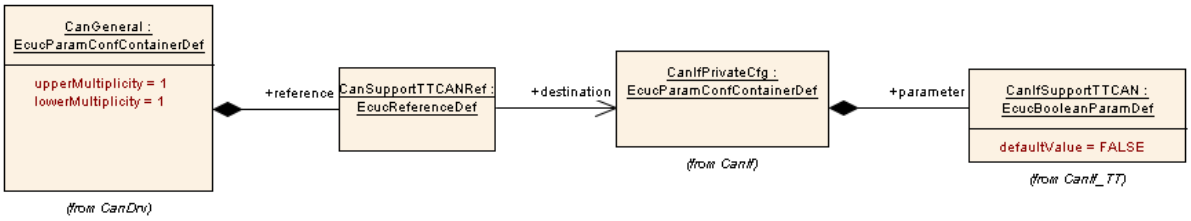

Figure 10.1: CAN Driver Time Triggered General Configuration

The reference `CanSupportTTCANRef` is described in Specification of CAN Driver [4], SWS Item Id *ECUC_Can_00430*.

CanTTController

SWS Item	[ECUC_Can_00001]
Container Name	CanTTController
Description	CanTTController is specified in the SWS TTCAN and contains the configuration parameters of the TTCAN controller(s) (which are needed in addition to the configuration parameters of the CAN controller(s)). This container is only included and valid if TTCAN is supported by the controller, enabled (see CanSupportTTCANRef, ECUC_Can_00430), and used.
Configuration Parameters	

Name	CanTTControllerApplWatchdogLimit [ECUC_Can_00139]		
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			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	CanTTControllerCycleCountMax [ECUC_Can_00138]		
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			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	CanTTControllerExpectedTxTrigger [ECUC_Can_00136]		
Description	Number of expected_tx_trigger.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	CanTTControllerExternalClockSynchronisation [ECUC_Can_00135]		
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			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: CanTTControllerLevel2 (ECUC_Can_00131)		

Name	CanTTControllerGlobalTimeFiltering [ECUC_Can_00134]		
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			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: CanTTControllerLevel2 (ECUC_Can_00131)		

Name	CanTTControllerInitialRefOffset [ECUC_Can_00128]		
Description	Defines the initial value for ref trigger offset.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 127		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	CanTTControllerInterruptEnable [ECUC_Can_00140]		
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 Watch Trigger reached. 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			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: CanTTControllerLevel2 (ECUC_Can_00131)		

Name	CanTTControllerLevel2 [ECUC_Can_00131]		
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			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	CanTTControllerNTUConfig [ECUC_Can_00141]		
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			

Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: CanTTControllerLevel2 (ECUC_Can_00131)		

Name	CanTTControllerOperationMode [ECUC_Can_00127]	
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
Post-Build Variant Value	true	
Value Configuration Class	**Pre-compile time**	X VARIANT-PRE-COMPILE
	Link time	–
	Post-build time	X VARIANT-POST-BUILD
Scope / Dependency	scope: ECU	

Name	CanTTControllerSyncDeviation [ECUC_Can_00132]		
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			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: CanTTControllerLevel2 (ECUC_Can_00131)		

Name	CanTTControllerTURRestore [ECUC_Can_00133]		
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 preriod. 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			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: CanTTControllerLevel2 (ECUC_Can_00131)		

Name	CanTTControllerTimeMaster [ECUC_Can_00129]		
Description	Defines whether the controller acts as a potential time master. TRUE: Potential time master. FALSE: Time slave.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	CanTTControllerTimeMasterPriority [ECUC_Can_00130]		
Description	Defines the time master priority.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	CanTTControllerTxEnableWindowLength [ECUC_Can_00137]		
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			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	CanTTControllerWatchTriggerGapTimeMark [ECUC_Can_00158]		
Description	watch trigger time mark after a gap		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	CanTTControllerWatchTriggerTimeMark [ECUC_Can_00157]		
Description	watch trigger time mark		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	CanTTIRQProcessing [ECUC_Can_00142]		
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.	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

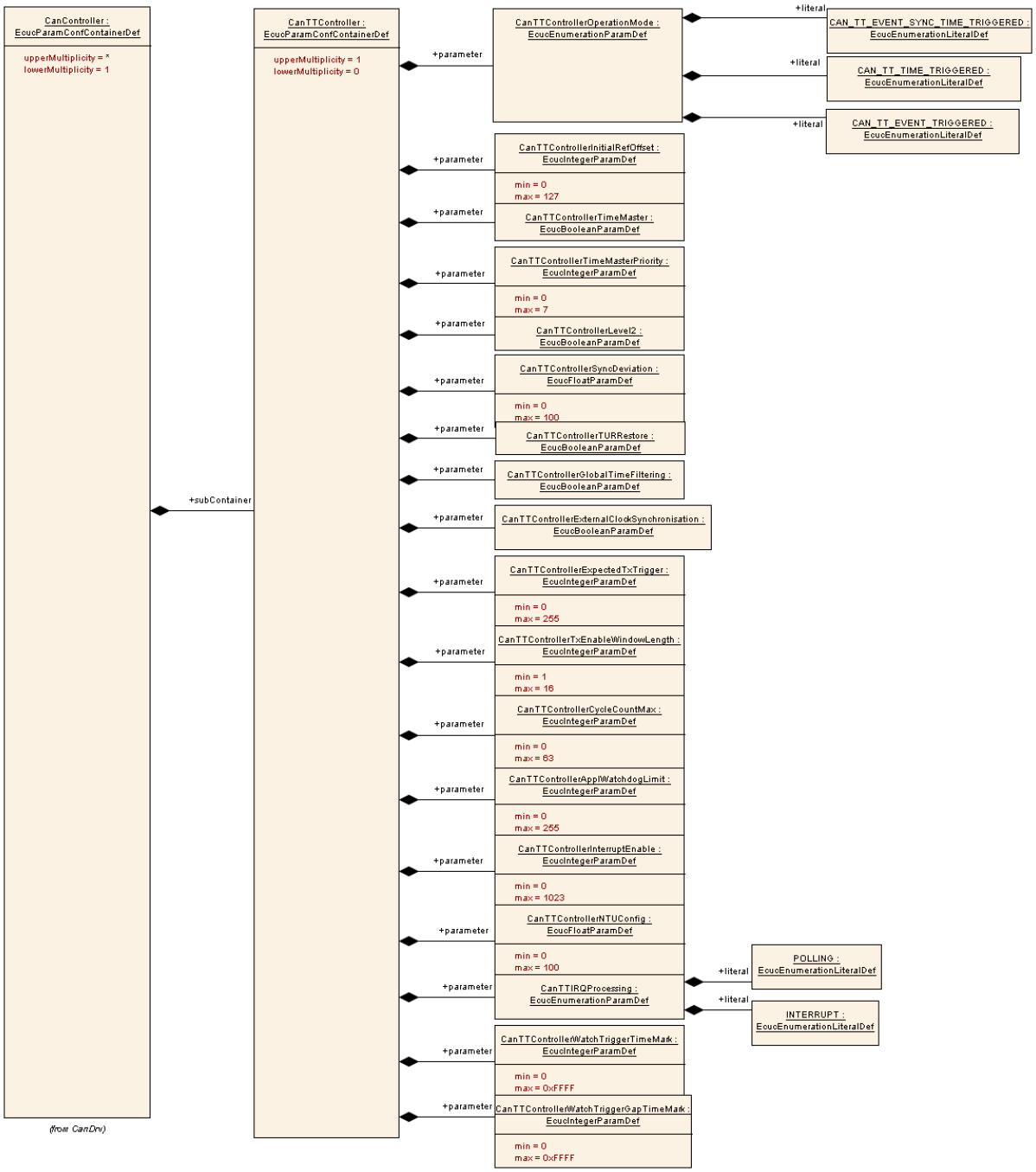

Figure 10.2: CAN Driver Time Triggered Controller Configuration

CanTTHardwareObjectTrigger

SWS Item	[ECUC_Can_00002]
Container Name	CanTTHardwareObjectTrigger

Description	CanTTHardwareObjectTrigger is specified in the SWS TTCAN and contains the configuration (parameters) of TTCAN triggers for Hardware Objects, which are additional to the configuration (parameters) of CAN Hardware Objects. This container is only included and valid if TTCAN is supported by the controller and, enabled (see CanSupportTTCANRef, ECUC_Can_00430), and used.
Configuration Parameters	

Name	CanTTHardwareObjectBaseCycle [ECUC_Can_00147]		
Description	Defines the cycle_offset. CanTTHardwareObjectBaseCycle must be not greater than cycle_count_max.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 63		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	CanTTHardwareObjectCycleRepetition [ECUC_Can_00148]		
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			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Name	CanTTHardwareObjectTimeMark [ECUC_Can_00146]		
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			

Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Name	CanTTHardwareObjectTriggerId [ECUC_Can_00155]		
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			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTTHardwareObjectTriggerType [ECUC_Can_00145]	
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.	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: CAN_OBJECT_TYPE		

No Included Containers

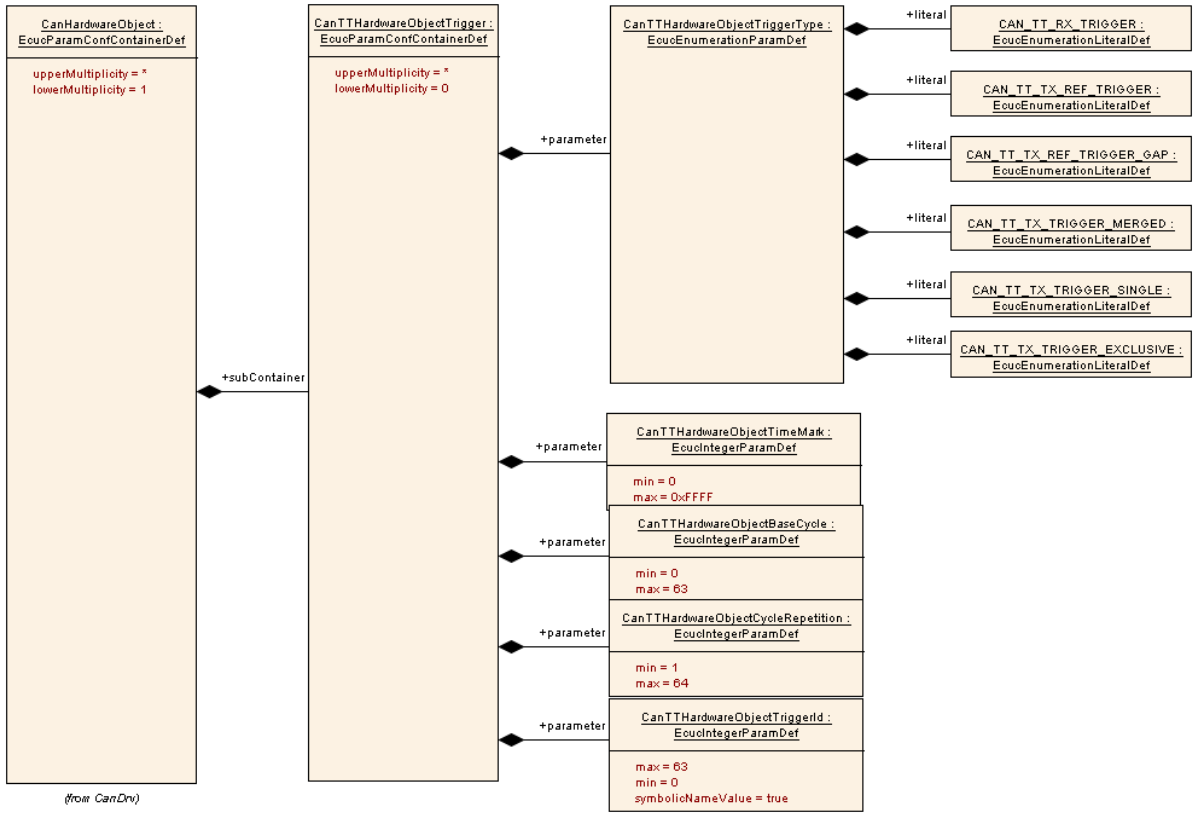

Figure 10.3: CAN Driver Time Triggered Hardware Object Configuration

10.3 Published information

For details refer to the chapter 10.3 "'Published Information"' in *SWS_BSWGeneral* [5]

A Not applicable requirements

[SWS_TtCan_00726] ⌈ These requirements are not applicable to this specification. ⌋*()*