Document Title	Specification of TTCAN Interface
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	433
Document Classification	Standard
Document Version	1.2.2
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History

Date	Version	Changed by	Change Description
31.03.2014	1.2.2	AUTOSAR Release Management	• Adapted description of exported TTCAN EcuC containers • Editorial changes
31.10.2013	1.2.1	AUTOSAR Release Management	• Editorial changes
31.01.2013	1.2.0	AUTOSAR Administration	• Updated scope of parameters • Formal update for traceability analysis • Aligned to General Documents • Adapted Production Error Specification
03.11.2010	1.1.0	AUTOSAR Administration	• Updated <User_TriggerTransmit> function with generated artifact from ComStack harmonization • Described behaviour of negative return value of <User_TriggerTransmit>
07.12.2009	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 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 5

2 Acronyms and abbreviations 7

3 Related documentation 8

3.1 Input documents 8
3.2 Related standards and norms 8
3.3 Related specification 9

4 Constraints and assumptions 10

5 Dependencies to other modules 11

5.1 Additional TTCAN specific dependencies to other modules 11
5.1.1 AUTOSAR Operating System 11
5.1.2 AUTOSAR PDU router 11
5.1.3 Upper Protocol Layers 11
5.1.4 TTCAN Driver 11

6 Requirements traceability 12

7 Functional specification 16

7.1 General functionality 16
7.2 TTCAN Interface state machine 16
7.3 TTCAN Job List 16
7.4 TTCAN Job List Execution Function 17
7.5 Data communication via TTCAN 18
7.6 TTCAN Controller mode 19
7.6.1 Controller operation modes 19
7.6.2 TTCAN Severe error 19
7.7 Extended Production Errors 20

8 API specification 21

8.1 Imported types 21
8.2 Type definitions 21
8.2.1 CanIf_TTTimeType 21
8.2.2 CanIf_TTMasterSlaveModeType 21
8.2.3 CanIf_TTSyncModeEnumType 22
8.2.4 CanIf_TTMasterStateType 22
8.2.5 CanIf_TTErrorLevelEnumType 22
8.2.6 CanIf_TTErrorLevelType 22
8.2.7 CanIf_TTSevereErrorEnumType 23
8.2.8 CanIf_TTTimeSourceType 23
8.2.9 CanIf_TTEventEnumType 23
8.2.10 CanIf_TTTimingErrorIRQType 23
8.3 Function definitions 24
8.3.1 CanIf_TTGetControllerTime 24
8.3.2 CanIf_TTGetMasterState 25
8.3.3 CanIf_TTGetNTUActual 26

8.3.4 CanIf_TTGetErrorLevel ... 27
8.3.5 CanIf_TTSetNextIsGap ... 28
8.3.6 CanIf_TTSetEndOfGap ... 28
8.3.7 CanIf_TTSetTimeCommand ... 29
8.3.8 CanIf_TTGlobalTimePreset ... 30
8.3.9 CanIf_TTSetExtClockSyncCommand ... 31
8.3.10 CanIf_TTSetNTUAdjust ... 31
8.4 Optional Function definitions ... 32
8.4.1 CanIf_TTJobListExec_<Controller> ... 32
8.4.2 CanIf_TTGetSyncQuality ... 33
8.4.3 CanIf_TTSetTimeMark ... 34
8.4.4 CanIf_TTCancelTimeMark ... 35
8.4.5 CanIf_TTAckTimeMark ... 36
8.4.6 CanIf_TTEnableTimeMarkIRQ ... 36
8.4.7 CanIf_TTDisableTimeMarkIRQ ... 37
8.4.8 CanIf_TTGetTimeMarkIRQStatus ... 38
8.5 Scheduled functions ... 39
8.6 Callback notifications ... 39
8.6.1 CanIf_TTApplWatchdogError ... 39
8.6.2 CanIf_TTTimingError ... 40
8.6.3 CanIf_TTSevereError ... 40
8.6.4 CanIf_TTGap ... 41
8.6.5 CanIf_TTStartOfCycle ... 42
8.6.6 CanIf_TTTimeDisc ... 42
8.6.7 CanIf_TTMasterStateChange ... 43
8.7 Expected interfaces ... 44
8.7.1 Mandatory interfaces ... 44
8.7.2 Optional interfaces ... 44
8.7.3 Configurable Interfaces ... 45
9 Sequence diagrams ... 47
9.1 Transmission with JobList (TriggerTransmit with decoupled buffer access) ... 47
9.2 Reception with Joblist ... 48
9.3 Job List Execution Function ... 49
10 Configuration specification ... 50
10.1 How to read this chapter ... 50
10.2 Containers and configuration parameters ... 50
10.2.1 CanIfTTGeneral ... 51
10.2.2 CanIfTTTxFrameTriggering ... 52
10.2.3 CanIfTTRxFrameTriggering ... 53
10.3 Published information ... 54
11 Not applicable requirements ... 55

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Basic Software module TTCAN Interface.

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 Interface is located in the communication hardware abstraction under the communication service layers (i.e. TTCAN State Manager, TTCAN Network Management, TTCAN Transport Protocol, PDU Router). It represents the interface to the services of the TTCAN Driver for the upper communication layers.

The TTCAN Interface module is an extension of the CAN Interface module [7] so this document shall only provide information and specifications which differ from the CAN Interface module.

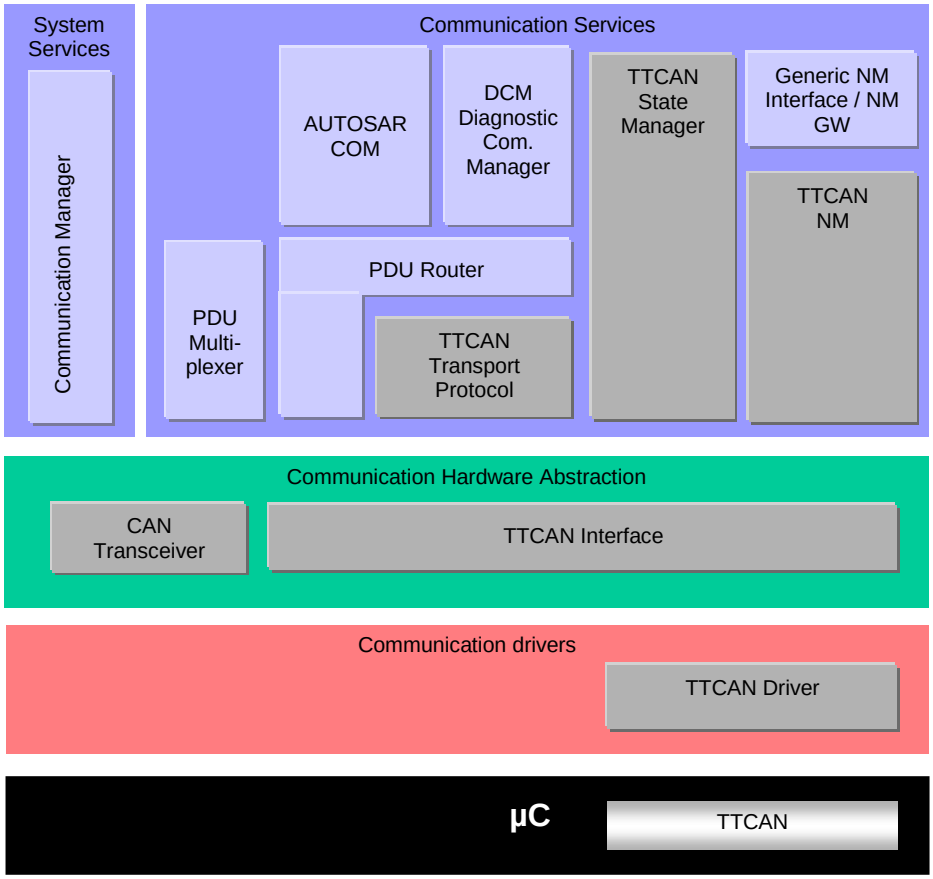

Figure 1-1 AUTOSAR TTCAN Layer Model (see [1])

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).

The TTCAN Interface consists of all TTCAN hardware independent tasks, which belong to the TTCAN communication device drivers of the corresponding ECU. This functionality is implemented once in the TTCAN Interface, so that underlying TTCAN device drivers only focus on access and control of the corresponding specific TTCAN hardware device.
The TTCAN Interface fulfils main control flow and data flow requirements of the PDU Router and upper layer communication modules of the AUTOSAR COM stack: transmit request processing, transmit confirmation / receive indication / error notification and start / stop of a TTCAN Controller and thus waking up / participating on a network. Its data processing and notification API is based on CAN L-PDUs, whereas the APIs for control and mode handling provide a TTCAN Controller related view.

In case of transmit requests the TTCAN Interface completes the L-PDU transmission with corresponding parameters and relays the CAN L-PDU via the appropriate TTCAN Driver to the TTCAN Controller. At reception the TTCAN Interface distributes the received L-PDUs to the upper layer. The assignment between receive L-PDU and upper layer is statically configured. At transmit confirmation the TTCAN Interface is responsible for the notification of upper layers about successful transmission.

The TTCAN Interface provides TTCAN communication abstracted access to the lower layer services for control and supervision of the TTCAN network. The TTCAN Interface forwards the status change requests from the CAN State Manager downwards to the lower layer TTCAN device drivers, and upwards the lower layer events are forwarded by the TTCAN Interface to e.g. the corresponding NM module.

2 Acronyms and abbreviations

Abbreviation/Acronym:	*Description:*
“at system configuration time”	= static configuration parameters stored in the TTCAN Interface; may be defined *after* compilation of the code of the TTCAN Interface, but have to be defined before the first execution of the TTCAN Interface code.
Arbitrating time window	See ISO 11898-4 [12]
OS	(AUTOSAR) Operating System
Basic cycle	See ISO 11898-4 [12]
BSW	Basic Software
CANIF; CanIf	CAN Interface
Communication Job	A TTCAN Communication Job defines the specific communication operation and the assigned execution time.
Continuous transmission	Contrary to ‘single shot’ a message will be transmitted cyclically even without a new transmit request.
Controller	A (TTCAN-)Controller is a CPU on-chip or external standalone hardware device. One Controller is connected to one physical channel.
Cycle time	See ISO 11898-4 [12]
Dem	Diagnostic Event Manager
DLC	Data Length Code (part of CAN L-PDU that describes the SDU length)
DLL	Data Link Layer
EcuM	ECU Manager
Exclusive time window	See ISO 11898-4 [12]
Gap	See ISO 11898-4 [12]
Global time	See ISO 11898-4 [12]
Hardware object	A CAN hardware object is defined as a PDU buffer inside the CAN RAM of the CAN hardware unit / CAN Controller.
ISR	Interrupt service routine
JLEF	(TTCAN) Job List Execution Function
Job List	A TTCAN Job List is a list of (maybe different) Communication Jobs sorted according to their respective execution start time.
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]
NTU	See ISO 11898-4 [12]
OS	Operating system
PduR	PDU Router
Reference message	See ISO 11898-4 [12]
SDU	Service Data Unit
Single shot	A message will be transmitted only once contrary to ‘continuous transmission’.
System matrix	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]
TtcanIf	TTCAN Interface
CanNm	CAN Network Management
CanSm	CAN State Manager
CanTp	CAN Transport Protocol
TX	Transmission or transmit
Tx_Trigger	See ISO 11898-4 [12]
UL	Upper layer

3 Related documentation

All documents of the referenced CAN Interface document [7] are also valid for this document.

3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[4] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[5] Requirements on CAN
AUTOSAR_SRS_CAN.pdf

[6] Specification of CAN Driver
AUTOSAR_SWS_CANDriver.pdf

[7] Specification of CAN Interface
AUTOSAR_SWS_CANInterface.pdf

[8] Specification of TTCAN Driver
AUTOSAR_SWS_TTCANDriver.pdf]

[9] Specification of ECU State Manager
AUTOSAR_SWS_ECUStateManager.pdf

[10] Requirements on TTCAN
AUTOSAR_SRS_TTCAN.pdf

[11] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.Pdf

3.2 Related standards and norms

[12] ISO11898-4 Road vehicles – Controller Area Network (CAN) Part4: Time-triggered communication

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [11] (SWS BSW General), which is also valid for TTCAN Interface.

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

4 Constraints and assumptions

The constraints and assumptions of the TTCAN Interface module are the same as for the CAN Interface module [7].

5 Dependencies to other modules

5.1 Additional TTCAN specific dependencies to other modules

This section describes the relations to other modules within the AUTOSAR basic software architecture. It contains brief descriptions of configuration information and services, which are additional required by the TTCAN Interface module from other modules. The dependencies described in the referenced CAN Interface module [7] also apply for the TTCAN Interface module.

5.1.1 AUTOSAR Operating System

It's possible to use dedicated Job List Execution Functions for each TTCAN Controller.

Whether the optional JLEF runs in a task concept or in an ISR is implementation specific. Refer to chapter 7.4.

5.1.2 AUTOSAR PDU router

Additional to the data access through the CAN Interface, as described in [7], the TTCAN Interface can call a Job List Execution Function synchronously to the TTCAN local lime. This shall ensure the request for data to be sent occur synchronously to the TTCAN local time. Within the Job List Execution Function the TTCAN Interface calls the callback function <UL_TriggerTransmit> of the PDU-Router in order to start the copy operation of PDU data. Additionally the Job List Execution Function can be used to read out received data synchronoulsy to the TTCAN local time.

5.1.3 Upper Protocol Layers

Inside the AUTOSAR BSW architecture the upper layers of the TTCAN Interface are represented by the PduR, CanNm, CanTp, CanSM and EcuM.

If the respective upper layer BSW module does not operate synchronously to the TTCAN local time, all occurrences are asynchronous to the code execution of this BSW module.

5.1.4 TTCAN Driver

The TTCAN Interface provides additional notification services used by the TTCAN Driver (refer to 0).

6 Requirements traceability

Requirement	Description	Satisfied by
-	-	SWS_TtCanIf_00010
-	-	SWS_TtCanIf_00011
-	-	SWS_TtCanIf_00012
-	-	SWS_TtCanIf_00013
-	-	SWS_TtCanIf_00014
-	-	SWS_TtCanIf_00015
-	-	SWS_TtCanIf_00016
-	-	SWS_TtCanIf_00017
-	-	SWS_TtCanIf_00018
-	-	SWS_TtCanIf_00019
-	-	SWS_TtCanIf_00020
-	-	SWS_TtCanIf_00021
-	-	SWS_TtCanIf_00022
-	-	SWS_TtCanIf_00023
-	-	SWS_TtCanIf_00024
-	-	SWS_TtCanIf_00025
-	-	SWS_TtCanIf_00026
-	-	SWS_TtCanIf_00027
-	-	SWS_TtCanIf_00028
-	-	SWS_TtCanIf_00029
-	-	SWS_TtCanIf_00034
-	-	SWS_TtCanIf_00035
-	-	SWS_TtCanIf_00036
-	-	SWS_TtCanIf_00037
-	-	SWS_TtCanIf_00038
-	-	SWS_TtCanIf_00039
-	-	SWS_TtCanIf_00040
-	-	SWS_TtCanIf_00041
-	-	SWS_TtCanIf_00042
-	-	SWS_TtCanIf_00043
-	-	SWS_TtCanIf_00044
-	-	SWS_TtCanIf_00045
-	-	SWS_TtCanIf_00046
-	-	SWS_TtCanIf_00047
-	-	SWS_TtCanIf_00048
-	-	SWS_TtCanIf_00050

-	-	SWS_TtCanIf_00051
-	-	SWS_TtCanIf_00052
-	-	SWS_TtCanIf_00053
-	-	SWS_TtCanIf_00054
-	-	SWS_TtCanIf_00055
-	-	SWS_TtCanIf_00056
-	-	SWS_TtCanIf_00057
-	-	SWS_TtCanIf_00059
-	-	SWS_TtCanIf_00060
-	-	SWS_TtCanIf_00061
-	-	SWS_TtCanIf_00063
-	-	SWS_TtCanIf_00064
-	-	SWS_TtCanIf_00066
-	-	SWS_TtCanIf_00068
-	-	SWS_TtCanIf_00071
-	-	SWS_TtCanIf_00081
-	-	SWS_TtCanIf_00088
-	-	SWS_TtCanIf_00096
-	-	SWS_TtCanIf_00097
-	-	SWS_TtCanIf_00098
-	-	SWS_TtCanIf_00099
-	-	SWS_TtCanIf_00123
-	-	SWS_TtCanIf_00124
-	-	SWS_TtCanIf_00137
-	-	SWS_TtCanIf_00144
BSW	-	SWS_TtCanIf_99999
SRS_BSW_00387	The Basic Software Module specifications shall specify how the callback function is to be implemented	SWS_TtCanIf_00058
SRS_Can_01121	CAN Interface shall be the interface layer between the underlying CAN Driver(s) and CAN transceiver Driver(s) and Upper Layers	SWS_TtCanIf_00065, SWS_TtCanIf_00067, SWS_TtCanIf_00069, SWS_TtCanIf_00070, SWS_TtCanIf_00072, SWS_TtCanIf_00073, SWS_TtCanIf_00074, SWS_TtCanIf_00075, SWS_TtCanIf_00076, SWS_TtCanIf_00077, SWS_TtCanIf_00080, SWS_TtCanIf_00082, SWS_TtCanIf_00083, SWS_TtCanIf_00084, SWS_TtCanIf_00085, SWS_TtCanIf_00086, SWS_TtCanIf_00087, SWS_TtCanIf_00101, SWS_TtCanIf_00102, SWS_TtCanIf_00103, SWS_TtCanIf_00104, SWS_TtCanIf_00105, SWS_TtCanIf_00106, SWS_TtCanIf_00107, SWS_TtCanIf_00108, SWS_TtCanIf_00109, SWS_TtCanIf_00110, SWS_TtCanIf_00112, SWS_TtCanIf_00113, SWS_TtCanIf_00114, SWS_TtCanIf_00115, SWS_TtCanIf_00116,

		SWS_TtCanIf_00117, SWS_TtCanIf_00119
SRS_Can_01131	The CAN Interface module shall provide the possibility to have polling and callback notification mechanism in parallel	SWS_TtCanIf_00089, SWS_TtCanIf_00090, SWS_TtCanIf_00091, SWS_TtCanIf_00092, SWS_TtCanIf_00093, SWS_TtCanIf_00094
SRS_TtCan_41010	A Job List shall be configurable.	SWS_TtCanIf_00002, SWS_TtCanIf_00141, SWS_TtCanIf_00143
SRS_TtCan_41011	If a Job List is available (see SRS_TtCan_41010) it shall be executed by a separate Job List Execution Function.	SWS_TtCanIf_00004, SWS_TtCanIf_00006, SWS_TtCanIf_00007, SWS_TtCanIf_00032, SWS_TtCanIf_00033, SWS_TtCanIf_00079
SRS_TtCan_41013	An occurred severe error (S3) shall be processed as a BusOff (see SRS_Can_01029 of CAN SRS)	SWS_TtCanIf_00120, SWS_TtCanIf_00121, SWS_TtCanIf_00122

Document: General requirements on Basic Software Modules [3]

Requirement	***Satisfied by***
[SRS_BSW_00337] Classification of errors	TTCANIF008
[SRS_BSW_00387] Specify the configuration class of call-out function	SWS_TtCanIf_00058

Usually the General requirements on BASIS Software Modules are realized by the CAN Interface SWS, which is the main Interface document. The requirements in this table only are mentioned for traceability reasons for the additional TTCAN SWS Item Ids.

Document: Requirements on CAN [5]

Requirement	***Satisfied by***
[SRS_Can_01121] Interfaces of the CAN Interface module	SWS_TtCanIf_00065, SWS_TtCanIf_00067, SWS_TtCanIf_00069, SWS_TtCanIf_00070, SWS_TtCanIf_00072, SWS_TtCanIf_00073, SWS_TtCanIf_00074, SWS_TtCanIf_00075, SWS_TtCanIf_00076, SWS_TtCanIf_00077, SWS_TtCanIf_00080, SWS_TtCanIf_00082, SWS_TtCanIf_00083, SWS_TtCanIf_00084, SWS_TtCanIf_00085, SWS_TtCanIf_00086, SWS_TtCanIf_00087, SWS_TtCanIf_00101, SWS_TtCanIf_00102, SWS_TtCanIf_00103, SWS_TtCanIf_00104, SWS_TtCanIf_00105, SWS_TtCanIf_00106, SWS_TtCanIf_00107, SWS_TtCanIf_00108, SWS_TtCanIf_00109, SWS_TtCanIf_00110, SWS_TtCanIf_00112, SWS_TtCanIf_00113, SWS_TtCanIf_00114, SWS_TtCanIf_00115, SWS_TtCanIf_00116, SWS_TtCanIf_00117, SWS_TtCanIf_00119
[SRS_Can_01131] Mixed mode of notification and polling mechanism	SWS_TtCanIf_00089, SWS_TtCanIf_00090, SWS_TtCanIf_00091, SWS_TtCanIf_00092, SWS_TtCanIf_00093, SWS_TtCanIf_00094

Usually the requirements on CAN are realized by the CAN Interface SWS, which is the main Interface document. The requirements in this table only are mentioned for traceability reasons for the additional TTCAN SWS Item Ids.

Document: Requirements on TTCAN [10] (includes TTCAN requirements additional to CAN)

Requirement	***Satisfied by***
[SRS_TtCan_41001] TTCAN support	Chapters 1-10
[SRS_TtCan_41002] CAN dependence	Linkage of configuration parameters to the CAN parameters (see ECUC_CanIf_00003, ECUC_CanIf_00005, ECUC_CanIf_00142). Delta description of chapter 7. Same namespace (prefix) in chapter 8.
[SRS_TtCan_41010] Job List	SWS_TtCanIf_00002, ECUC_CanIf_00003, ECUC_CanIf_00126, ECUC_CanIf_00132, ECUC_CanIf_00136, SWS_TtCanIf_00141, ECUC_CanIf_00142, SWS_TtCanIf_00143
[SRS_TtCan_41011] Job List Execution Function	SWS_TtCanIf_00004, SWS_TtCanIf_00006, SWS_TtCanIf_00007, SWS_TtCanIf_00032, SWS_TtCanIf_00033, SWS_TtCanIf_00079, ECUC_CanIf_00127
[SRS_TtCan_41012] Time Mark	ECUC_CanIf_00128, ECUC_CanIf_00132, ECUC_CanIf_00133, ECUC_CanIf_00136
[SRS_TtCan_41013] Handling of Severe Errors as BusOff	SWS_TtCanIf_00120, SWS_TtCanIf_00121, SWS_TtCanIf_00122

7 Functional specification

7.1 General functionality

Time-triggered CAN is a higher level protocol layer additional to the CAN protocol itself, which remains unchanged within the time-triggered communication.

This functional specification only provide specifications, which are additional to the CAN stack, to realize the mode Time Triggered CAN (TTCAN). Nevertheless the implementation shall provide the Standard CAN mode anyway.

7.2 TTCAN Interface state machine

The TTCAN Interface use the same states as the CAN Interface.

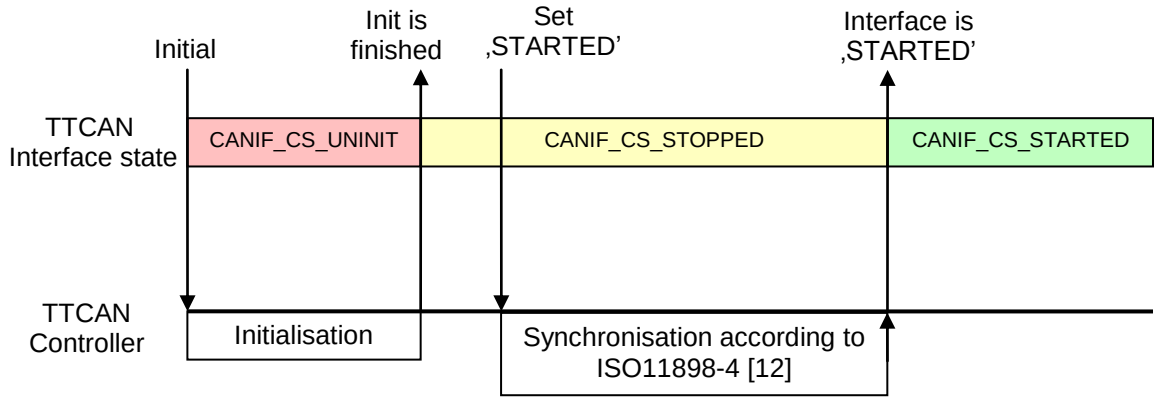

Figure 7-1: Exemplary Startup of TTCAN

7.3 TTCAN Job List

A TTCAN Job List is a list of Communication Jobs sorted according to their respective execution start time.

The TTCAN Job List shall be used if a synchronized copy operation into the Controller is required and/or a synchronized readout of the Controller (optional feature) shall be realized. Otherwise the normal CAN procedure without a Job List can be used.

[SWS_TtCanIf_00002] 「The Copy Operation into/from the TTCAN Controller shall be scheduled within a Job List.」(SRS_TtCan_41010)

[SWS_TtCanIf_00143] 「For each Controller that is controlled by the TTCAN Interface one dedicated Job List and one dedicated JLEF (refer to chapter 7.4) shall

be used. It's possible to mixture both variants, with and without the usage of a Job List.」(SRS_TtCan_41010)

7.4 TTCAN Job List Execution Function

[SWS_TtCanIf_00004]「If a Job List is used, the TTCAN Job List Execution Function (JLEF) shall execute the Communication Jobs of the Job List synchronously to the Controller time (i.e. at well-defined points in time).」(SRS_TtCan_41011)

The execution of JLEF is implementation specific.

[SWS_TtCanIf_00006]「The API names of the JLEF shall obey the following pattern:

- `CanIf_TTJobListExec_0()` for Controller # 0
- `CanIf_TTJobListExec_1()` for Controller # 1
- `CanIf_TTJobListExec_2()` for Controller # 2
- `CanIf_TTJobListExec_3()` for Controller # 3
- … and so on, if more than 4 Controller are supported.」(SRS_TtCan_41011)

[SWS_TtCanIf_00007]「If the JLEF lost synchronisation to the local time of the TTCAN controller then the function `Dem_ReportErrorStatus(CANIF_TT_E_JLE_SYNC, DEM_EVENT_STATUS_FAILED)` shall be called」(SRS_TtCan_41011)

Exemplary the JLEF performs the following steps:

1. Retrieve the cycle time of the Controller by calling `Can_TTGetControllerTime()`.
2. If the cycle time cannot be retrieved
   a. Call `Dem_ReportErrorStatus(CANIF_TT_E_JLE_SYNC, DEM_EVENT_STATUS_FAILED)`
   b. Terminate the execution of JLEF.
   Otherwise, the JLEF continues with step 3.
3. Check whether the JLEF was called by start of new Basic cycle.
   If it is false, continue with step 4.
   Otherwise check whether the next job is scheduled for this Basic cycle.
   If it is true, set the interrupt timer to the next job's start time in order to invoke the JLEF again and terminate the execution of JLEF
   Otherwise terminate execution of JLEF.
4. If the cycle time delay compared to the job start time is larger than a maximum delay (configuration parameter `CanIfTTMaxIsrDelay`, see ECUC_CanIf_00005), the execution of the Job List is considered to be asynchronous to the local time and thus the following actions are performed:

a. Call `Dem_ReportErrorStatus( CANIF_TT_E_JLE_SYNC, DEM_EVENT_STATUS_FAILED)`
b. Add some 'safety margin' (i.e. some timespan which takes jitter into account)
c. Search the Job List for the subsequent job, i.e. that job with an invocation time greater than the current local time + safety margin.
d. Search for the next job list entry, which is valid for the current basic cycle. If the end of the job-list is reached, wrap around to the next basic cycle and continue the search for that respective basic cycle.
e. If the next job is scheduled for this Basic cycle:
   Schedule next job, exemplary by using the time mark interrupt
   Otherwise disable timer interrupt
f. Terminate the execution of JLEF.

Otherwise, the JLEF continues with step 5.

5. Retrieve the sorted list of Communication Operations of the current Job pointed to by the current job-pointer and execute the retrieved communication operations in the configured order.
6. Search for the next job list entry, which is valid for the current basic cycle. If the end of the job-list is reached, wrap around to the next basic cycle and continue the search for that respective basic cycle.
7. If the next job is scheduled for this Basic cycle:
   set the interrupt timer to this job's start time
   Otherwise disable timer interrupt
8. Terminate the execution of JLEF.

7.5 Data communication via TTCAN

TTCAN is a deterministic time driven communication system. Each datum that should be transmitted or received has to be scheduled at system configuration time.

A detailed description of Synchronization, Transmission Triggering, Reception Triggering, Initialization and Failure handling can be found in ISO 11898-4 [12].

Additional TTCAN specific requirements:

[SWS_TtCanIf_00141] ⌈ If a job list is configured for a Tx L-PDU (see ECUC_CanIf_00126), a function call of `CanIf_Transmit()` (see SWS_CanIf_00318) shall not directly call `Can_Write()`. The information that a call of `CanIf_Transmit()` occurred has to be buffered within the TtcanIf until the data is transmitted by the job list.⌋(SRS_TtCan_41010)

Note: The kind of buffering the information of SWS_TtCanIf_00141 is implementation specific.

Rationale for SWS_TtCanIf_00141: A job list needs to be configured for HW objects which transmit in BasicCAN mode, where one HW object can be used to serve different time slots within the TTCAN system matrix. In this case a job list has to take

care, which message is available in the HW object at the correct time. A `Can_Write()` call directly after `CanIf_Transmit()` can violate this.

7.6 TTCAN Controller mode

This chapter corresponds to the chapter “CAN Controller mode” of the CAN Interface SWS.

7.6.1 Controller operation modes

7.6.1.1 Additional items to CANIF_CS_INIT

[SWS_TtCanIf_00120] ⌈If a CanIf Controller mode state machine is in state CANIF_CS_INIT and when function CanIf_TTSevereError() is called, then the CAN Interface module shall take that CanIf Controller mode state machine to state CANIF_CS_INIT, and the CAN Interface module shall call the function CanSM_ControllerBusOff() for the CAN Network assigned to parameter Controller of CanIf_TTSevereError().⌋(SRS_TtCan_41013)

This API is mapped to a BusOff API of the CanSM, because, this API indicates a severe error of the TTCAN controller. The handling and recovery of such an error is equal to BusOff.

7.6.2 TTCAN Severe error

[SWS_TtCanIf_00121] ⌈If a CanIf Controller mode state machine is in state CANIF_CS_STARTED when the function `CanIf_TTSevereError(ControllerId, CanIf_TTSevereError)` is called with parameter `ControllerId` referencing that CanIf Controller mode state machine, then the CanIf shall call `Can_SetControllerMode(Controller, CAN_T_STOP)` and the CAN Interface module shall call `CanSM_ControllerBusOff(ControllerId)` of the CanSm. ⌋ (SRS_TtCan_41013)

This API is mapped to a BusOff API of the CanSM, because, this API indicates a severe error of the TTCAN controller. The handling and recovery of such an error is equal to BusOff.

7.7 Extended Production Errors

[TTCANIF008]

⌈*Type or error*	*Related error code*	*Value [hex]*
Job List Execution lost synchronization to the TTCAN local time	CANIF_TT_E_JLE_SYNC	Assigned by DEM

⌋(SRS_BSW_00337)

8 API specification

In the following sections the TTCAN specific APIs and types are described.

8.1 Imported types

Additional TTCAN specific imported types

[SWS_TtCanIf_00124]⌈

Module	*Imported Type*
Can	Can_TTErrorLevelType
	Can_TTMasterStateType
	Can_TTTURType
	Can_TTTimeSourceType
	Can_TTTimeType
Can_GeneralTypes	Can_IdType
ComStack_Types	PduIdType
	PduInfoType
Std_Types	Std_ReturnType

⌋()

Note: PduIdType is missing as of ComStack_Types.

8.2 Type definitions

Additional TTCAN specific type definitions

8.2.1 CanIf_TTTimeType

[SWS_TtCanIf_00059]⌈

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

⌋()

8.2.2 CanIf_TTMasterSlaveModeType

[SWS_TtCanIf_00096]⌈

Name:	`CanIf_TTMasterSlaveModeType`	
Type:	`Enumeration`	
Range:	`CANIF_TT_BACKUP_MASTER`	Master-Slave Mode: Backup master
	`CANIF_TT_CURRENT_MASTER`	Master-Slave Mode: Current master
	`CANIF_TT_MASTER_OFF`	Master-Slave Mode: Master off
	`CANIF_TT_SLAVE`	Master-Slave Mode: Slave
Description:	Master-Slave Mode	

⌋()

8.2.3 CanIf_TTSyncModeEnumType

[SWS_TtCanIf_00097]⌈

Name:	CanIf_TTSyncModeEnumType	
Type:	Enumeration	
Range:	CANIF_TT_IN_GAP	Sync mode: In_Gap
	CANIF_TT_IN_SCHEDULE	Sync mode: In_Schedule
	CANIF_TT_SYNC_OFF	Sync mode: Sync_Off
	CANIF_TT_SYNCHRONIZING	Sync mode: Synchronizing
Description:	Sync mode	

⌋()

8.2.4 CanIf_TTMasterStateType

[SWS_TtCanIf_00060]⌈

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

⌋()

8.2.5 CanIf_TTErrorLevelEnumType

[SWS_TtCanIf_00098]⌈

Name:	CanIf_TTErrorLevelEnumType	
Type:	Enumeration	
Range:	CANIF_TT_ERROR_S0	Error level S0: No Error
	CANIF_TT_ERROR_S1	Error level S1: Warning
	CANIF_TT_ERROR_S2	Error level S2: Error
	CANIF_TT_ERROR_S3	Error level S3: Fatal Error
Description:	Error level (S0-S3)	

⌋()

8.2.6 CanIf_TTErrorLevelType

[SWS_TtCanIf_00061]⌈

Name:	CanIf_TTErrorLevelType		
Type:	Structure		
Element:	CanIf_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 CanIf_TTSevereErrorEnumType

[SWS_TtCanIf_00137]⌈

Name:	CanIf_TTSevereErrorEnumType	
Type:	Enumeration	
Range:	CANIF_TT_CONFIG_ERROR	Event: see ISO11898-4
	CANIF_TT_WATCH_TRIGGER_REACHED	Event: Watch Trigger reached
	CANIF_TT_APPL_WATCHDOG	Event: see ISO 11898-4
Description:	Event that causes a severe error	

⌋()

8.2.8 CanIf_TTTimeSourceType

[SWS_TtCanIf_00063]⌈

Name:	CanIf_TTTimeSourceType	
Type:	Enumeration	
Range:	CANIF_TT_CYCLE_TIME	Time source: Cycle Time
	CANIF_TT_GLOBAL_TIME	Time source: Global Time
	CANIF_TT_LOCAL_TIME	Time source: Local Time
	CANIF_TT_UNDEFINED	Time source: Undefined
Description:	Time source of time values in TTCAN	

⌋()

8.2.9 CanIf_TTEventEnumType

[SWS_TtCanIf_00099]⌈

Name:	CanIf_TTEventEnumType	
Type:	Enumeration	
Range:	CANIF_TT_ERROR_LEVEL_CHANGED	Event: Error Level changed
	CANIF_TT_INIT_WATCH_TRIGGER	Event: Init Watch Trigger reached
	CANIF_TT_NO_ERROR	No error
	CANIF_TT_SYNC_FAILED	Event: Sync failed
	CANIF_TT_TX_OVERFLOW	Event: Tx Overflow
	CANIF_TT_TX_UNDERFLOW	Event: Tx Underflow
Description:	Event that causes a Timing/Error IRQ	

⌋()

8.2.10 CanIf_TTTimingErrorIRQType

[SWS_TtCanIf_00064]⌈

Name:	CanIf_TTTimingErrorIRQType		
Type:	Structure		
Element:	CanIf_TTErrorLevelType	errorLevel	Current error level
	CanIf_TTEventEnumType	event	Event that caused the IRQ

Description:	Combines all events that are reported by CanIf_TTTimingError (event indication and error level)

⌋()

8.3 Function definitions

Additional TTCAN specific function definitions

8.3.1 CanIf_TTGetControllerTime

[SWS_TtCanIf_00065]⌈

Service name:	CanIf_TTGetControllerTime	
Syntax:	`Std_ReturnType CanIf_TTGetControllerTime( uint8 ControllerId, CanIf_TTTimeType* CanIf_TTGlobalTime, CanIf_TTTimeType* CanIf_TTLocalTime, CanIf_TTTimeType* CanIf_TTCycleTime, uint8* CanIf_TTCycleCount )`	
Service ID[hex]:	0x33	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Controller from which the time information shall be retrieved
Parameters (inout):	None	
Parameters (out):	CanIf_TTGlobalTime	Address to store return value: Global time
	CanIf_TTLocalTime	Address to store return value: Local time
	CanIf_TTCycleTime	Address to store return value: Cycle time
	CanIf_TTCycleCount	Address to store return value: Cycle count value
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Gets the current values for the global, local and cycle time and the cycle count of the controller	

⌋(SRS_Can_01121)

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

Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred

[SWS_TtCanIf_00101] ⌈The function `CanIf_TTGetControllerTime()` shall call (wraps) `Can_TTGetControllerTime(Controller, Can_TTGlobalTime, CanTTLocalTime, Can_TTCycleTime, Can_TTCycleCount).` ⌋ (SRS_Can_01121)

[SWS_TtCanIf_00010] ⌈ If parameter `Controller` of `CanIf_TTGetControllerTime()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals ON), the function

CanIf_TTGetControllerTime() shall report development error code CANIF_E_PARAM_CONTROLLER to the Det_ReportError service of the DET module.」()

[SWS_TtCanIf_00011]「Caveats of CanIf_TTGetControllerTime(): The TTCAN Interface has to be initialized before this API service may be called.」()

[SWS_TtCanIf_00066]「If development error detection for the TtcanIf module is enabled: The function CanIf_TTGetControllerTime shall raise the error CANIF_E_PARAM_POINTER and shall return E_NOT_OK if one of the parameter CanIf_TTCycleCount, CanIf_TTGlobalTime, CanIf_TTLocalTime and CanIf_TTCycleTime is a NULL pointer.」()

8.3.2 CanIf_TTGetMasterState

[SWS_TtCanIf_00067]「

Service name:	CanIf_TTGetMasterState	
Syntax:	`Std_ReturnType CanIf_TTGetMasterState( uint8 ControllerId, CanIf_TTMasterStateType* CanIf_TTMasterState )`	
Service ID[hex]:	0x34	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
Parameters (inout):	None	
Parameters (out):	CanIf_TTMasterState	Address to store return value: Master state
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
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.	

」(SRS_Can_01121)

[SWS_TtCanIf_00102]「The function CanIf_TTGetMasterState() shall call (wraps) Can_TTGetMasterState(Controller, Can_TTMasterState).」(SRS_Can_01121)

[SWS_TtCanIf_00012]「If parameter Controller of CanIf_TTGetMasterState() has an invalid value and if development error detection is enabled (i.e. CANIF_DEV_ERROR_DETECT equals ON), the function CanIf_TTGetMasterState() shall report development error code

CANIF_E_PARAM_CONTROLLER to the Det_ReportError service of the DET module. ⌋()

[SWS_TtCanIf_00013] ⌈Caveats of CanIf_TTGetMasterState(): The TTCAN Interface has to be initialized before this API service may be called.⌋()

[SWS_TtCanIf_00068] ⌈If development error detection for the TtcanIf module is enabled: The function CanIf_TTGetMasterState shall raise the error CAN_E_PARAM_POINTER and shall return E_NOT_OK if the parameter CanIf_TTMasterState is a NULL pointer.⌋()

8.3.3 CanIf_TTGetNTUActual

[SWS_TtCanIf_00069]⌈

Service name:	CanIf_TTGetNTUActual	
Syntax:	Std_ReturnType CanIf_TTGetNTUActual(uint8 ControllerId, float32 CanIf_TTNTUAct)	
Service ID[hex]:	0x35	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
Parameters (inout):	None	
Parameters (out):	CanIf_TTNTUAct	Address to store return value: Actual value of NTU. Value is given in microseconds
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
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.	

⌋(SRS_Can_01121)

[SWS_TtCanIf_00103]⌈The function CanIf_TTGetNTUActual() shall call (wraps) Can_TTGetNTUActual(Controller, Can_TTTURAct).⌋(SRS_Can_01121)

[SWS_TtCanIf_00014] ⌈If parameter Controller of CanIf_TTGetNTUActual() has an invalid value and if development error detection is enabled (i.e. CANIF_DEV_ERROR_DETECT equals ON), the function CanIf_TTGetNTUActual() shall report development error code CANIF_E_PARAM_CONTROLLER to the Det_ReportError service of the DET module.⌋()

[SWS_TtCanIf_00015]⌈ Caveats of `CanIf_TTGetNTUActual()`: The TTCAN Interface has to be initialized before this API service may be called.⌋()

8.3.4 CanIf_TTGetErrorLevel

[SWS_TtCanIf_00070]⌈

Service name:	CanIf_TTGetErrorLevel	
Syntax:	`Std_ReturnType CanIf_TTGetErrorLevel( uint8 ControllerId, CanIf_TTErrorLevelType* CanIf_TTErrorLevel )`	
Service ID[hex]:	0x36	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller from which the error level shall be retrieved
Parameters (inout):	None	
Parameters (out):	CanIf_TTErrorLevel	Address to store return value: Error level
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
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.	

⌋(SRS_Can_01121)

[SWS_TtCanIf_00104]⌈ The function `CanIf_TTGetErrorLevel()` shall call (wraps) `Can_TTGetErrorLevel(Controller, Can_TTErrorLevel).` ⌋ (SRS_Can_01121)

[SWS_TtCanIf_00016]⌈If parameter `Controller` of `CanIf_TTGetErrorLevel()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), the function `CanIf_TTGetErrorLevel()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module. ⌋()

[SWS_TtCanIf_00017]⌈ Caveats of `CanIf_TTGetErrorLevel()`: The TTCAN Interface has to be initialized before this API service may be called.⌋()

[SWS_TtCanIf_00071]⌈ If development error detection for the TtcanIf module is enabled: The function `CanIf_TTGetErrorLevel()` shall raise the error `CAN_E_PARAM_POINTER` and shall return `CAN_NOT_OK` if the parameter `CanIf_TTErrorLevel` is a NULL pointer.⌋()

8.3.5 CanIf_TTSetNextIsGap

[SWS_TtCanIf_00072]⌈

Service name:	CanIf_TTSetNextIsGap	
Syntax:	`Std_ReturnType CanIf_TTSetNextIsGap( uint8 ControllerId )`	
Service ID[hex]:	0x37	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Sets the "Next_is_Gap" bit.	

⌋(SRS_Can_01121)

[SWS_TtCanIf_00105]⌈The function `CanIf_TTSetNextIsGap()` shall call (wraps) `Can_TTSetNextIsGap(Controller).`⌋(SRS_Can_01121)

[SWS_TtCanIf_00018]⌈If parameter `Controller` of CanIf_TTSetNextIsGap() has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), the function `CanIf_TTSetNextIsGap()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module.⌋()

[SWS_TtCanIf_00019]⌈ Caveats of `CanIf_TTSetNextIsGap()`: The TTCAN Interface has to be initialized before this API service may be called.⌋()

8.3.6 CanIf_TTSetEndOfGap

[SWS_TtCanIf_00073]⌈

Service name:	CanIf_TTSetEndOfGap	
Syntax:	`Std_ReturnType CanIf_TTSetEndOfGap( uint8 ControllerId )`	
Service ID[hex]:	0x38	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
Parameters (inout):	None	

Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Signals the end of a gap.	

⌋(SRS_Can_01121)

[SWS_TtCanIf_00106]⌈The function `CanIf_TTSetEndOfGap()` shall call (wraps) `Can_TTSetNextIsGap(Controller)`.⌋(SRS_Can_01121)

[SWS_TtCanIf_00020]⌈If parameter `Controller` of CanIf_TTSetEndOfGap() has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals ON), the function CanIf_TTSetEndOfGap() shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module.⌋()

[SWS_TtCanIf_00021]⌈Caveats of CanIf_TTSetEndOfGap(): The TTCAN Interface has to be initialized before this API service may be called.⌋()

8.3.7 CanIf_TTSetTimeCommand

[SWS_TtCanIf_00074]⌈

Service name:	CanIf_TTSetTimeCommand	

Syntax:	`Std_ReturnType CanIf_TTSetTimeCommand(`
	`  uint8 ControllerId`
	`)`	

Service ID[hex]:	0x39	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Adjusts the global time at the beginning of the next basic cycle by the amount of "global time preset"	

⌋(SRS_Can_01121)

[SWS_TtCanIf_00107]⌈The function `CanIf_TTSetTimeCommand()` shall call (wraps) `Can_TTSetTimeCommand(Controller)`.⌋(SRS_Can_01121)

[SWS_TtCanIf_00022]⌈If parameter `Controller` of `CanIf_TTSetTimeCommand()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), the function

`CanIf_TTSetTimeCommand()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module. ⌋()

[SWS_TtCanIf_00023]⌈ Caveats of `CanIf_TTSetTimeCommand()`: The TTCAN Interface has to be initialized before this API service may be called.⌋()

8.3.8 CanIf_TTGlobalTimePreset

[SWS_TtCanIf_00075]⌈

Service name:	CanIf_TTGlobalTimePreset	
Syntax:	`Std_ReturnType CanIf_TTGlobalTimePreset( uint8 ControllerId, CanIf_TTTimeType CanIf_TTGlobalTimePreset )`	
Service ID[hex]:	0x3a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
	CanIf_TTGlobalTimePreset	New value for "global time preset"
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Sets the value of "global time preset".	

⌋(SRS_Can_01121)

[SWS_TtCanIf_00108]⌈ The function `CanIf_TTGlobalTimePreset()` shall call (wraps) `Can_TTGlobalTimePreset(Controller, Can_TTGlobalTimePreset).`⌋(SRS_Can_01121)

[SWS_TtCanIf_00024] ⌈ If parameter `Controller` of `CanIf_TTGlobalTimePreset()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), the function `CanIf_TTGlobalTimePreset()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module. ⌋()

[SWS_TtCanIf_00025]⌈Caveats of `CanIf_TTGlobalTimePreset()`: The TTCAN Interface has to be initialized before this API service may be called.⌋()

8.3.9 CanIf_TTSetExtClockSyncCommand

[SWS_TtCanIf_00076]⌈

Service name:	CanIf_TTSetExtClockSyncCommand	
Syntax:	`Std_ReturnType CanIf_TTSetExtClockSyncCommand( uint8 ControllerId )`	
Service ID[hex]:	0x3b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
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".	

⌋(SRS_Can_01121)

[SWS_TtCanIf_00109] ⌈ The function `CanIf_TTSetExtClockSyncCommand()` shall call (wraps) `Can_TTSetExtClockSyncCommand(Controller).` ⌋ (SRS_Can_01121)

[SWS_TtCanIf_00026] ⌈ If parameter `Controller` of `CanIf_TTSetExtClockSyncCommand()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), the function `CanIf_TTSetExtClockSyncCommand()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module. ⌋()

[SWS_TtCanIf_00027]⌈Caveats of `CanIf_TTSetExtClockSyncCommand()`: The TTCAN Interface has to be initialized before this API service may be called.⌋()

8.3.10 CanIf_TTSetNTUAdjust

[SWS_TtCanIf_00077]⌈

Service name:	CanIf_TTSetNTUAdjust
Syntax:	`Std_ReturnType CanIf_TTSetNTUAdjust( uint8 ControllerId, float32 CanIf_TTNTUAdjust )`
Service ID[hex]:	0x3c
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant

Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
	CanIf_TTNTUAdjust	New value for "NTU adjust". Value is given in microseconds.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Sets the value of "NTU adjust". Together with the local oscillator period, "TUR adjust" can be derived from "NTU adjust".	

⌋(SRS_Can_01121)

[SWS_TtCanIf_00110]⌈The function `CanIf_TTSetNTUAdjust()` shall call (wraps) `Can_TTSetNTUAdjust(Controller, Can_TTNTUAdjust).`⌋(SRS_Can_01121)

[SWS_TtCanIf_00028]⌈If parameter `Controller` of `CanIf_TTSetNTUAdjust()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), the function `CanIf_ TTSetNTUAdjust ()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module.⌋()

[SWS_TtCanIf_00029]⌈Caveats of `CanIf_ TTSetNTUAdjust ()`: The TTCAN Interface has to be initialized before this API service may be called.⌋()

8.4 Optional Function definitions

Additional optional TTCAN specific function definitions

8.4.1 CanIf_TTJobListExec_<Controller>

[SWS_TtCanIf_00079]⌈

Service name:	CanIf_TTJobListExec_<Controller>

Syntax:	`void CanIf_TTJobListExec_<Controller>(`
	`  void`
	`)`

Service ID[hex]:	0x50
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Processes the job list of the TTCAN controller <Controller>.

」(SRS_TtCan_41011)

[SWS_TtCanIf_00032] 「The function `CanIf_TTJobListExec_<Controller>()` shall exist once per TTCAN Controller, which use a Job List.」(SRS_TtCan_41011)

[SWS_TtCanIf_00033] 「The function name of each instance of `CanIf_TTJobListExec_<Controller>()` shall contain the index of the respective TTCAN Controller (`Controller`).」(SRS_TtCan_41011)

[SWS_TtCanIf_00034] 「Caveats of `CanIf_TTJobListExec_<Controller>()`: The TTCAN Interface has to be initialized before this API service may be called.」()

For each TTCAN Controller (identified by index `Controller`), the execution of `CanIf_TTJobListExec_<Controller>()` can either run in a regular OS task or it is registered in the AUTOSAR OS as ISR, triggered by the TTCAN Controller.

8.4.2 CanIf_TTGetSyncQuality

[SWS_TtCanIf_00080]「

Service name:	CanIf_TTGetSyncQuality	
Syntax:	`Std_ReturnType CanIf_TTGetSyncQuality( uint8 ControllerId, boolean* CanIf_TTClockSpeed, boolean* CanIf_TTGlobalTimePhase )`	
Service ID[hex]:	0x47	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
Parameters (inout):	None	
Parameters (out):	CanIf_TTClockSpeed	Address to store return value: True if the synchronization deviation is smaller than the "Synchronization deviation limit"
	CanIf_TTGlobalTimePhase	Address to store return value: True if the the global time is in phase with the time master.
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Gets the synchronization quality.	

」(SRS_Can_01121)

[SWS_TtCanIf_00112] 「 The function `CanIf_TTGetSyncQuality()` shall call (wraps) `Can_TTGetSyncQuality(Controller, Can_TTClockSpeed, Can_TTGlobalTimePhase).`」(SRS_Can_01121)

[SWS_TtCanIf_00035] ⌈ If parameter `Controller` of `CanIf_TTGetSyncQuality()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), the function `CanIf_TTGetSyncQuality()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module. ⌋()

[SWS_TtCanIf_00036] ⌈ Caveats of `CanIf_TTGetSyncQuality()`: The TTCAN Interface has to be initialized before this API service may be called. ⌋()

[SWS_TtCanIf_00081] ⌈ If development error detection for the TtcanIf module is enabled: The function `CanIf_TTGetSyncQuality()` shall raise the error `CAN_E_PARAM_POINTER` and shall return `E_NOT_OK` if one of the parameter `CanIf_ClockSpeed` and `CanIf_GlobalTimePhase` is a NULL pointer. ⌋()

8.4.3 CanIf_TTSetTimeMark

[SWS_TtCanIf_00082] ⌈

Service name:	CanIf_TTSetTimeMark	
Syntax:	`Std_ReturnType CanIf_TTSetTimeMark( uint8 ControllerId, CanIf_TTTimeType CanIf_TTTimeMark, CanIf_TTTimeSourceType CanIf_TTTimeSource )`	
Service ID[hex]:	0x48	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
	CanIf_TTTimeMark	Gives the value of the time mark to be set.
	CanIf_TTTimeSource	Defines the time source for the time mark to be set.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Sets a new value for the time mark for the given time source.	

⌋(SRS_Can_01121)

[SWS_TtCanIf_00113] ⌈ The function `CanIf_TTSetTimeMark()` shall call (wraps) `Can_TTSetTimeMark(Controller, Can_TTTimeMark, Can_TTTimeSource)`. ⌋(SRS_Can_01121)

[SWS_TtCanIf_00037]⌈ If parameter `Controller` of `CanIf_TTSetTimeMark()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), the function `CanIf_TTSetTimeMark()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module.⌋()

[SWS_TtCanIf_00038]⌈ Caveats of `CanIf_TTSetTimeMark()`: The TTCAN Interface has to be initialized before this API service may be called.⌋()

8.4.4 CanIf_TTCancelTimeMark

[SWS_TtCanIf_00083]⌈

Service name:	CanIf_TTCancelTimeMark	
Syntax:	`Std_ReturnType CanIf_TTCancelTimeMark( uint8 ControllerId )`	
Service ID[hex]:	0x49	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Cancels the time mark.	

⌋(SRS_Can_01121)

[SWS_TtCanIf_00114]⌈ The function `CanIf_TTCancelTimeMark()` shall call (wraps) `Can_TTCancelTimeMark(Controller)`.⌋(SRS_Can_01121)

[SWS_TtCanIf_00039]⌈ If parameter `Controller` of `CanIf_TTCancelTimeMark()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), the function `CanIf_TTCancelTimeMark()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module. ⌋()

[SWS_TtCanIf_00040]⌈ Caveats of `CanIf_TTSetTimeMark()`: The TTCAN Interface has to be initialized before this API service may be called.⌋()

8.4.5 CanIf_TTAckTimeMark

[SWS_TtCanIf_00084]⌈

Service name:	CanIf_TTAckTimeMark	
Syntax:	`Std_ReturnType CanIf_TTAckTimeMark( uint8 ControllerId )`	
Service ID[hex]:	0x4a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Acknowledges the time mark interrupt by resetting the flag in the interrupt vector register.	

⌋(SRS_Can_01121)

[SWS_TtCanIf_00115]⌈The function `CanIf_TTAckTimeMark()` shall call (wraps) `Can_TTAckTimeMark(Controller).`⌋(SRS_Can_01121)

[SWS_TtCanIf_00041]⌈If parameter `Controller` of `CanIf_TTAckTimeMark()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), the function `CanIf_TTAckTimeMark()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module.⌋()

[SWS_TtCanIf_00042]⌈ Caveats of `CanIf_TTAckTimeMark()`: The TTCAN Interface has to be initialized before this API service may be called.⌋()

8.4.6 CanIf_TTEnableTimeMarkIRQ

[SWS_TtCanIf_00085]⌈

Service name:	CanIf_TTEnableTimeMarkIRQ	
Syntax:	`Std_ReturnType CanIf_TTEnableTimeMarkIRQ( uint8 ControllerId )`	
Service ID[hex]:	0x4b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
Parameters (inout):	None	

Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Enables the time mark interrupt.	

⌋(SRS_Can_01121)

[SWS_TtCanIf_00116]⌈The function `CanIf_TTEnableTimeMarkIRQ()` shall call (wraps) `Can_TTEnableTimeMarkIRQ(Controller)`.⌋(SRS_Can_01121)

[SWS_TtCanIf_00043] ⌈ If parameter `Controller` of `CanIf_TTEnableTimeMarkIRQ()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), the function `CanIf_TTEnableTimeMarkIRQ()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module. ⌋()

[SWS_TtCanIf_00044] ⌈ Caveats of `CanIf_TTEnableTimeMarkIRQ()`: The TTCAN Interface has to be initialized before this API service may be called.⌋()

8.4.7 CanIf_TTDisableTimeMarkIRQ

[SWS_TtCanIf_00086]⌈

Service name:	CanIf_TTDisableTimeMarkIRQ	
Syntax:	`Std_ReturnType CanIf_TTDisableTimeMarkIRQ( uint8 ControllerId )`	
Service ID[hex]:	0x4c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Disables the time mark interrupt.	

⌋(SRS_Can_01121)

[SWS_TtCanIf_00117]⌈The function `CanIf_TTDisableTimeMarkIRQ()` shall call (wraps) `Can_TTDisableTimeMarkIRQ(Controller).`⌋(SRS_Can_01121)

[SWS_TtCanIf_00045] ⌈ If parameter `Controller` of `CanIf_TTDisableTimeMarkIRQ()` has an invalid value and if development error

detection is enabled (i.e. CANIF_DEV_ERROR_DETECT equals ON), the function CanIf_TTDisableTimeMarkIRQ() shall report development error code CANIF_E_PARAM_CONTROLLER to the Det_ReportError service of the DET module. ⌋()

[SWS_TtCanIf_00046] ⌈ Caveats of CanIf_TTDisableTimeMarkIRQ(): The TTCAN Interface has to be initialized before this API service may be called. ⌋()

8.4.8 CanIf_TTGetTimeMarkIRQStatus

[SWS_TtCanIf_00087]⌈

Service name:	CanIf_TTGetTimeMarkIRQStatus	
Syntax:	`Std_ReturnType CanIf_TTGetTimeMarkIRQStatus( uint8 ControllerId, boolean* CanIf_TTIRQStatus )`	
Service ID[hex]:	0x4d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller
Parameters (inout):	None	
Parameters (out):	CanIf_TTIRQStatus	Address to store return value: True if the timer for the time mark is pending.
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Gets the IRQ status of the time mark.	

⌋(SRS_Can_01121)

[SWS_TtCanIf_00119]⌈ The function CanIf_TTGetTimeMarkIRQStatus() shall call (wraps) Can_TTGetTimeMarkIRQStatus(Controller, Can_TTIRQStatus).⌋(SRS_Can_01121)

[SWS_TtCanIf_00047] ⌈ If parameter Controller of CanIf_TTGetTimeMarkIRQStatus() has an invalid value and if development error detection is enabled (i.e. CANIF_DEV_ERROR_DETECT equals ON), the function CanIf_TTGetTimeMarkIRQStatus() shall report development error code CANIF_E_PARAM_CONTROLLER to the Det_ReportError service of the DET module. ⌋()

[SWS_TtCanIf_00048] ⌈ Caveats of CanIf_TTGetTimeMarkIRQStatus(): The TTCAN Interface has to be initialized before this API service may be called. ⌋()

[SWS_TtCanIf_00088]⌈ If development error detection for the TtcanIf module is enabled: The function `CanIf_TTGetTimeMarkIRQStatus()` shall raise the error `CAN_E_PARAM_POINTER` and shall return `E_NOT_OK` if the parameter `CanIf_IRQStatus` is a NULL pointer.⌋()

8.5 Scheduled functions

Additional TTCAN specific function definitions

The TTCAN Interface module has no additional scheduled functions.

8.6 Callback notifications

This is a list of functions provided for other modules.

Additional TTCAN specific callback notifications

The callback notification specified within this chapter will be called by the CAN Driver module either in context of a main function or an interrupt.

8.6.1 CanIf_TTApplWatchdogError

[SWS_TtCanIf_00089]⌈

Service name:	CanIf_TTApplWatchdogError	
Syntax:	`Std_ReturnType CanIf_TTApplWatchdogError( uint8 ControllerId )`	
Service ID[hex]:	0x5b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller for which the application watchdog error shall be reported.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Reports an application watchdog error.	

⌋(SRS_Can_01131)

[SWS_TtCanIf_00050] ⌈ If parameter `ControllerId` of `CanIf_TTApplWatchdogError()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), then the function `CanIf_TTApplWatchdogError()` shall report development error code

CANIF_E_PARAM_CONTROLLER to the Det_ReportError service of the DET module.」
()

8.6.2 CanIf_TTTimingError

[SWS_TtCanIf_00090]「

Service name:	CanIf_TTTimingError	
Syntax:	`Std_ReturnType CanIf_TTTimingError( uint8 ControllerId, CanIf_TTTimingErrorIRQType CanIf_TTTimingErrorIRQ )`	
Service ID[hex]:	0x5c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller for which the timing error shall be reported.
	CanIf_TTTimingErrorIRQ	Type of timing error.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Reports one of the following errors: - Change of error level - Tx overflow / underflow - Synchronization failed - Init watch trigger	

」(SRS_Can_01131)

Note: This callback service is called by the CAN Driver module (supporting TTCAN) and implemented in the CAN Interface module (supporting TTCAN). It is called if error level S1 or S2 (see ISO 11898-4 [12]) have been detected in the corresponding controller.

[SWS_TtCanIf_00051] 「 If parameter ControllerId of CanIf_TTTimingError() has an invalid value and if development error detection is enabled (i.e. CANIF_DEV_ERROR_DETECT equals ON), then the function CanIf_TTTimingError() shall report development error code CANIF_E_PARAM_CONTROLLER to the Det_ReportError service of the DET module.」
()

8.6.3 CanIf_TTSevereError

[SWS_TtCanIf_00122]「

Service name:	CanIf_TTSevereError
Syntax:	`void CanIf_TTSevereError( uint8 ControllerId,`

	CanIf_TTSevereErrorEnumType CanIf_TTSevereError)	
Service ID[hex]:	0x5c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller at which the severe error occured
	CanIf_TTSevereError	type of severe error
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Reports one of the following errors: - failed to serve appl. watchdog - config error - watch trigger reached	

」(SRS_TtCan_41013)

Note: This callback service is called by the CAN Driver module (supporting TTCAN) and implemented in the CAN Interface module (supporting TTCAN). It is called if error level S3 (severe error, see ISO 11898-4 [12]) has been detected in the corresponding controller.

[SWS_TtCanIf_00123] 「 If parameter `ControllerId` of `CanIf_TTSevereError()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), then the function `CanIf_TTSevereError()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module.」 ()

8.6.4 CanIf_TTGap

[SWS_TtCanIf_00091]「

Service name:	CanIf_TTGap	
Syntax:	Std_ReturnType CanIf_TTGap(uint8 ControllerId)	
Service ID[hex]:	0x5d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller for which the gap shall be reported.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Reports the occurrence of a gap.	

」(SRS_Can_01131)

[SWS_TtCanIf_00052] ⌈If parameter `ControllerId` of `CanIf_TTGap()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), then the function `CanIf_TTGap()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module.⌋()

8.6.5 CanIf_TTStartOfCycle

[SWS_TtCanIf_00092]⌈

Service name:	CanIf_TTStartOfCycle	
Syntax:	`Std_ReturnType CanIf_TTStartOfCycle( uint8 ControllerId, uint8 CanIf_TTCycleCount )`	
Service ID[hex]:	0x5e	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller for which the start of cycle shall be reported.
	CanIf_TTCycleCount	Cycle count value for the cycle that is started
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Reports the start of a basic cycle.	

⌋(SRS_Can_01131)

[SWS_TtCanIf_00053] ⌈ If parameter `ControllerId` of `CanIf_TTStartOfCycle()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), then the function `CanIf_TTStartOfCycle()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module.⌋ ()

8.6.6 CanIf_TTTimeDisc

[SWS_TtCanIf_00093]⌈

Service name:	CanIf_TTTimeDisc	
Syntax:	`Std_ReturnType CanIf_TTTimeDisc( uint8 ControllerId )`	
Service ID[hex]:	0x5f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN

		controller for which the time discontinuity shall be reported.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Reports a time discontinuity.	

」(SRS_Can_01131)

[SWS_TtCanIf_00054]「If parameter `ControllerId` of `CanIf_TTTimeDisc()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), then the function `CanIf_TTTimeDisc()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module.」()

8.6.7 CanIf_TTMasterStateChange

[SWS_TtCanIf_00094]「

Service name:	CanIf_TTMasterStateChange	

Syntax:	`Std_ReturnType CanIf_TTMasterStateChange(`
	`  uint8 ControllerId,`
	`  CanIf_TTMasterStateType CanIf_TTMasterState`
	`)`	

Service ID[hex]:	0x60	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerId	Abstracted CanIf ControllerId which is assigned to a CAN controller for which the master state change shall be reported.
	CanIf_TTMasterState	Master state including sync mode, master-slave mode and current ref trigger offset
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Function successful E_NOT_OK: Development error occurred
Description:	Reports change of the master state between potential and current master.	

」(SRS_Can_01131)

[SWS_TtCanIf_00055]「If parameter `ControllerId` of `CanIf_TTMasterStateChange()` has an invalid value and if development error detection is enabled (i.e. `CANIF_DEV_ERROR_DETECT` equals `ON`), then the function `CanIf_TTMasterStateChange()` shall report development error code `CANIF_E_PARAM_CONTROLLER` to the Det_ReportError service of the DET module.」()

8.7 Expected interfaces

8.7.1 Mandatory interfaces

Additional TTCAN specific mandatory interfaces

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

[SWS_TtCanIf_00056]⌈

API function	*Description*
Can_TTGetControllerTime	Gets the current values for the global, local and cycle time and the cycle count of the controller
Can_TTGetErrorLevel	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.
Can_TTGetMasterState	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.
Can_TTGetNTUActual	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.
Can_TTGlobalTimePreset	Sets the value of "global time preset".
Can_TTSetEndOfGap	Signals the end of a gap.
Can_TTSetExtClockSyncCommand	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".
Can_TTSetNTUAdjust	Sets the value of "NTU adjust". Together with the local oscillator period, "TUR adjust" can be derived from "NTU adjust".
Can_TTSetNextIsGap	Sets the "Next_is_Gap" bit.
Can_TTSetTimeCommand	Adjusts the global time at the beginning of the next basic cycle by the amount of "global time preset"

⌋()

8.7.2 Optional interfaces

Additional TTCAN specific optional interfaces

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

[SWS_TtCanIf_00057]⌈

API function	*Description*
Can_TTAckTimeMark	Acknowledges the time mark interrupt by resetting the flag in the interrupt vector register.
Can_TTCancelTimeMark	Cancels the time mark.
Can_TTDisableTimeMarkIRQ	Disables the time mark interrupt.
Can_TTEnableTimeMarkIRQ	Enables the time mark interrupt.
Can_TTGetSyncQuality	Gets the synchronization quality.
Can_TTGetTimeMarkIRQStatus	Gets the IRQ status of the time mark.

Can_TTReceive	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.
Can_TTSetTimeMark	Sets a new value for the time mark for the given time source.

⌋()

8.7.3 Configurable Interfaces

Additional TTCAN specific configurable interfaces

This chapter lists all interfaces where the target API service of any upper layer, which require one or more of these mentioned interfaces to be called has to be set up by static configuration of the TTCAN Interface. The target function is usually a call-back function. The names of these kinds of interfaces are not fixed because they are configurable.

8.7.3.1 <User_TriggerTransmit>

[SWS_TtCanIf_00058]⌈

Service name:	<User_TriggerTransmit>	

Syntax:	`Std_ReturnType <User_TriggerTransmit>(`
	`  PduIdType TxPduId,`
	`  PduInfoType* PduInfoPtr`
	`)`	

Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the SDU that is requested to be transmitted.
	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU data has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description:	Within this API, the upper layer module (called module) shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->SduLength.	

⌋(SRS_BSW_00387)

When calling the PduR, this function has to be named `PduR_CanIfTriggerTransmit()`.

This API service of an upper layer BSW module <User_> (e.g. PduR) is called by the TTCAN Interface module to request from this upper layer BSW module that the PDU with index ‘TxPduId’ has to be copied to the location in a temporary L-SDU buffer of the TtcanIf to which this part of ‘PduInfoPtr’ points.

[SWS_TtCanIf_00144] ⌈ If during JLEF `<User_TriggerTransmit>()` returns E_NOT_OK, the TtcanIf shall not call `Can_Write()` afterwards (see sequence diagram 9.1) Sequence diagram 9.1 shows only the case when <User_TriggerTransmit>() returns E_OK.⌋()

Reason for SWS_TtCanIf_00144: It is possible that e.g. the PDU is not available in COM module. This may be due to a stopped PDU group in COM module.

Caveats of `<User_TriggerTransmit>()`: This API service is called during the execution of the TTCAN JLEF.

9 Sequence diagrams

The following sequence diagrams show the interactions of the TTCAN Interface additional to the CAN Interface.

9.1 Transmission with JobList (TriggerTransmit with decoupled buffer access)

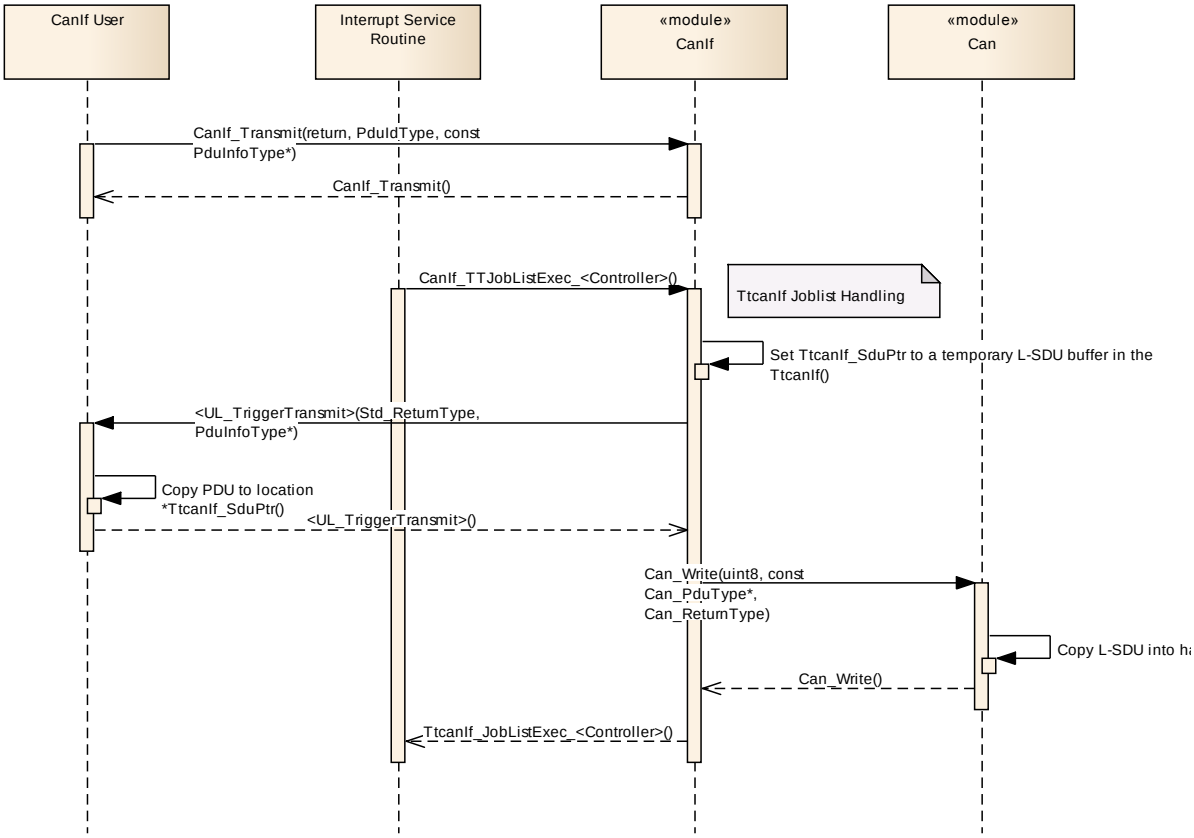

Figure 9-1: CAN Interface Time Triggered transmission with joblist

9.2 Reception with Joblist

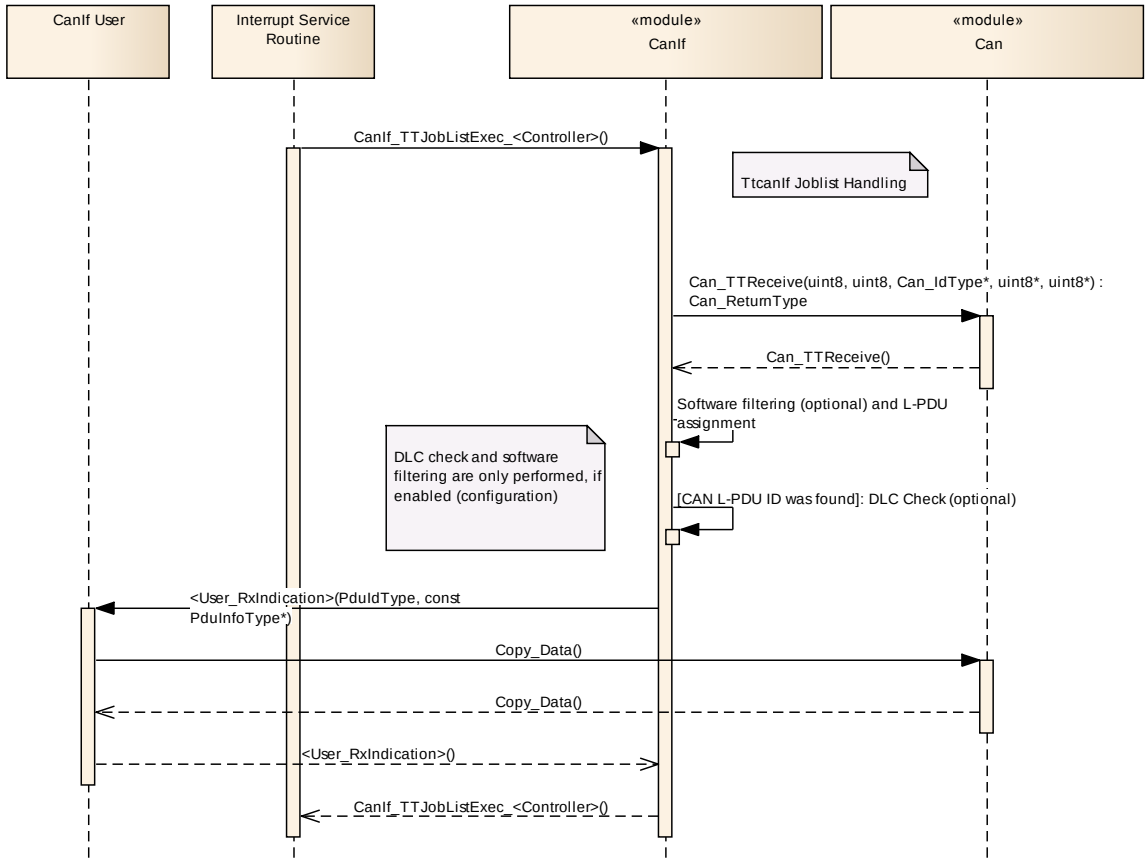

Figure 9-2: CAN Interface Time Triggered reception with joblist

9.3 Job List Execution Function

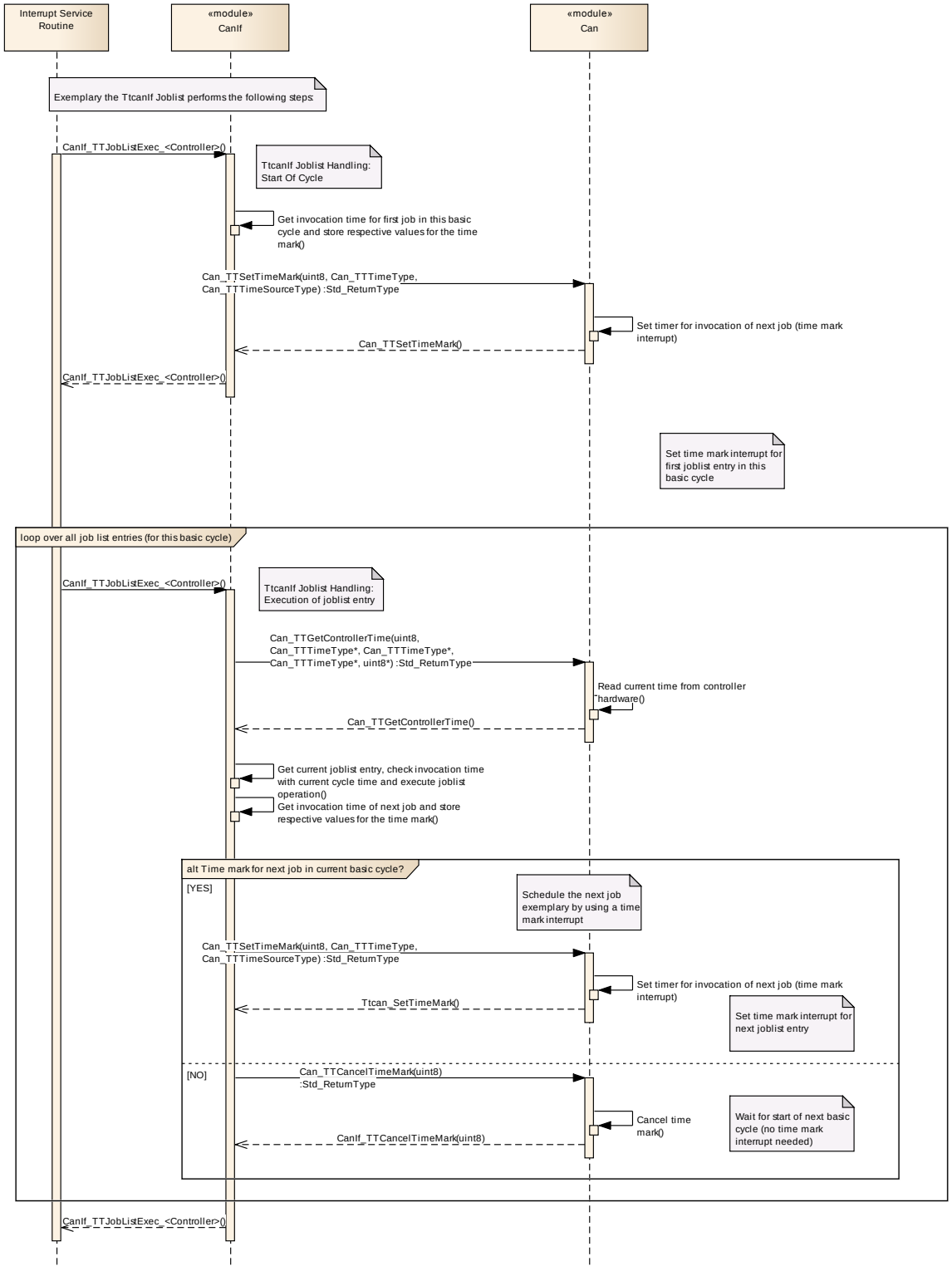

Figure 9-3: CAN Interface Time Triggered job list execution function

10 Configuration specification

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

Chapter 10.2 specifies the structure (containers) and the parameters of the module TTCAN Interface.

Chapter 10.3 specifies published information of the module TTCAN Interface.

10.1 How to read this chapter

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

10.2 Containers and configuration parameters

Additional TTCAN specific configuration parameters

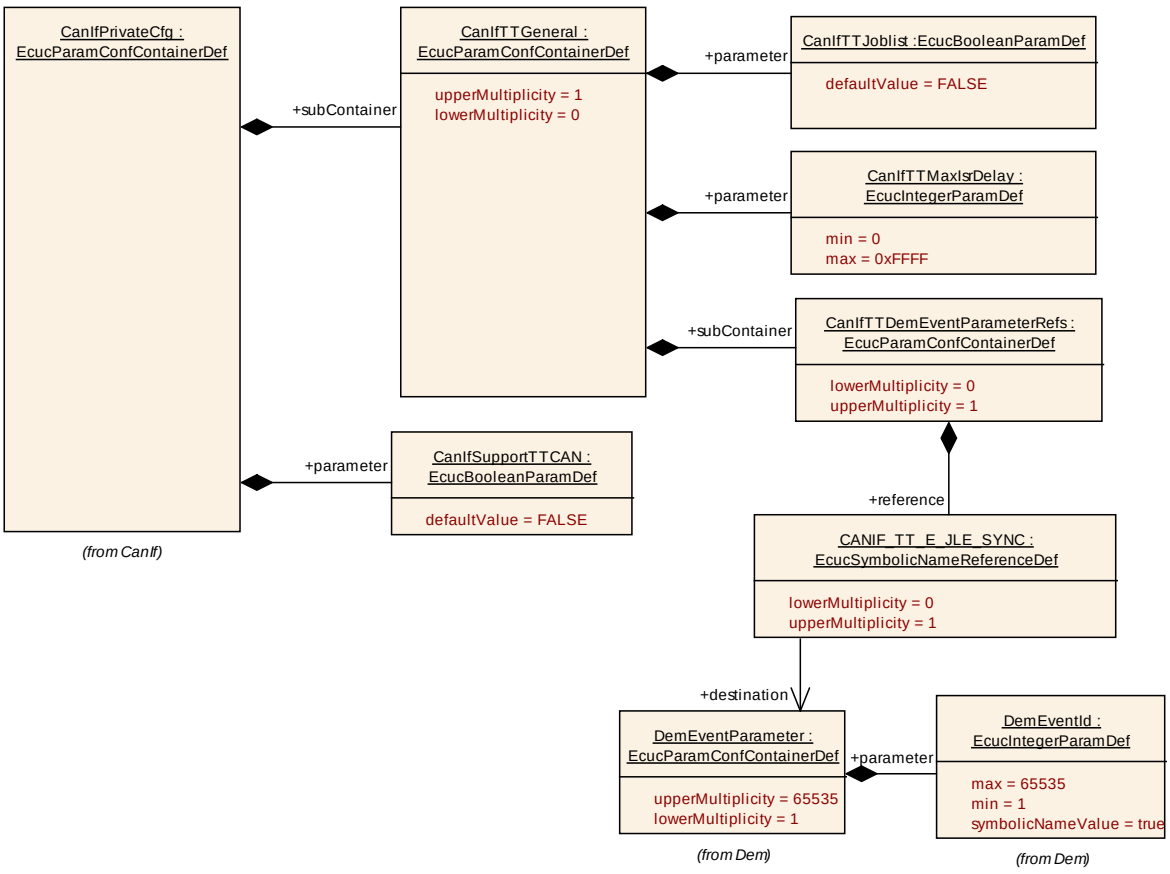

Figure 10-1: CAN Interface Time Triggered Private Configuration

The parameter CanIfSupportTTCAN is described in Specification of CAN Interface [7], SWS Item Id ECUC_CanIf_00675.

10.2.1 CanIfTTGeneral

SWS Item	**ECUC_CanIf_00005 :**
Container Name	CanIfTTGeneral
Description	CanIfTTGeneral is specified in the SWS TTCAN Interface and defines if and in which way TTCAN is supported. This container is only included and valid if TTCAN is supported by the controller, enabled (see CanIfSupportTTCAN, ECUC_CanIf_00675), and used.
Configuration Parameters	

SWS Item	**ECUC_CanIf_00126 :**		
Name	CanIfTTJoblist		
Description	Defines whether TTCAN is processed via a joblist. TRUE: Joblist is used. FALSE: No joblist is used. This parameter is only configurable if TTCAN is enabled by parameter CanIfSupportTTCAN.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: CanIfSupportTTCAN		

SWS Item	**ECUC_CanIf_00127 :**		
Name	CanIfTTMaxIsrDelay		
Description	Defines the maximum delay for the execution of the joblist execution function JLEF. This parameter is only configurable if a joblist is enabled by parameter CanIfTTJobList.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: CanIfTTJobList		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CanIfTTDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.

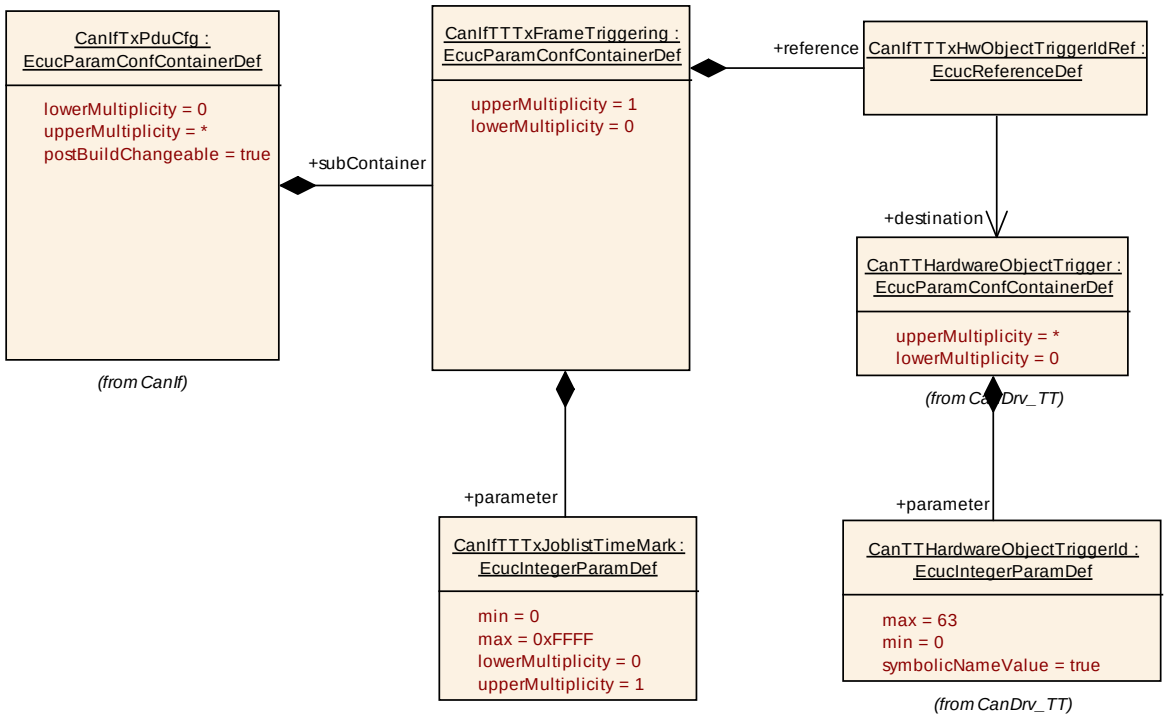

Figure 10-2: CAN Interface Time Triggered Transmit PDU Configuration

10.2.2 CanIfTTTxFrameTriggering

SWS Item	**ECUC_CanIf_00142 :**
Container Name	CanIfTTTxFrameTriggering
Description	CanIfTTTxFrameTriggering is specified in the SWS TTCAN Interface and defines Frame trigger for TTCAN transmission. This container is only included and valid if TTCAN is supported by the controller, enabled (see CanIfSupportTTCAN, ECUC_CanIf_00675), and a joblist is used.
Configuration Parameters	

SWS Item	**ECUC_CanIf_00132 :**		
Name	CanIfTTTxJoblistTimeMark		
Description	Defines the point in time, when the joblist execution funciton (JLEF) shall be called for the referenced tx frame trigger. Value is given in cycle time. This parameter is only configurable if a joblist is enabled by parameter CanIfTTJobList.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: CanIfTTJoblist		

SWS Item	**ECUC_CanIf_00128 :**
Name	CanIfTTTxHwObjectTriggerIdRef
Description	This parameter refers to a particular TTCAN hardware transmit object

	Trigger of a hardware object in the TTCAN Driver Module, which is referred via plain CAN parameter CANIF_HTH_HANDLETYPE_REF. This parameter is only configurable if a joblist is enabled by parameter CanIfTTJobList.		
Multiplicity	1		
Type	Reference to [CanTTHardwareObjectTrigger]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: CanIfTTJoblist		

No Included Containers

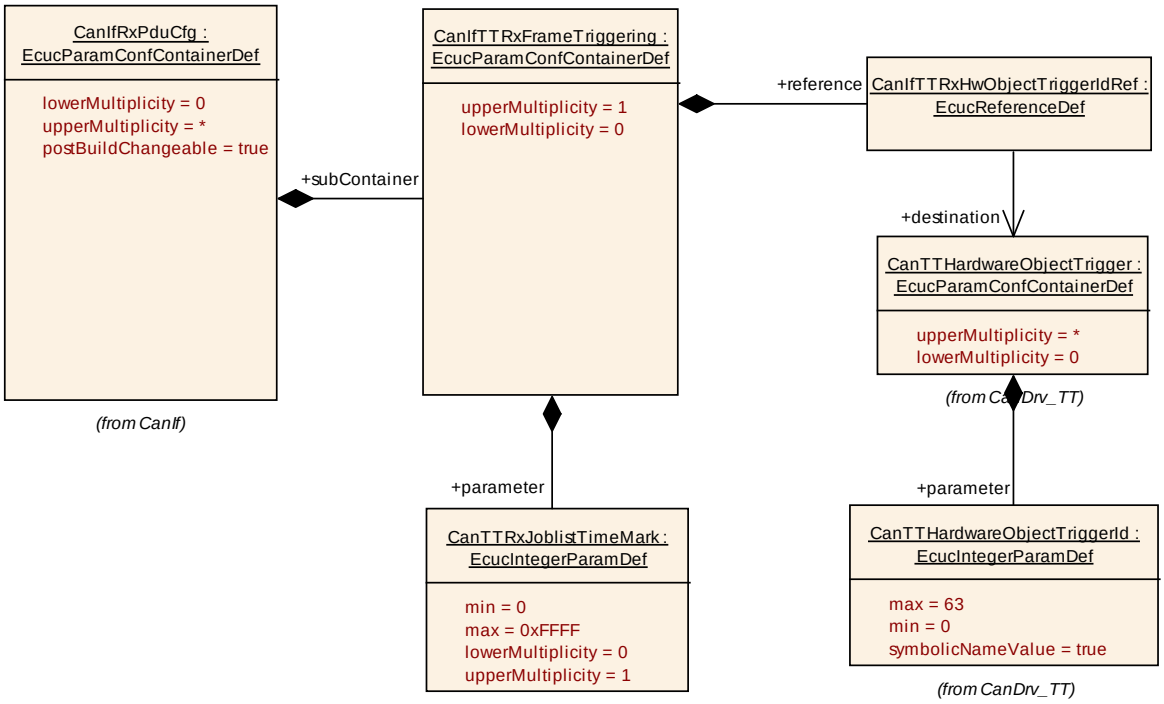

Figure 10-3: CAN Interface Time Triggered Receive PDU Configuration

10.2.3 CanIfTTRxFrameTriggering

SWS Item	**ECUC_CanIf_00003 :**
Container Name	CanIfTTRxFrameTriggering
Description	CanIfTTRxFrameTriggering is specified in the SWS TTCAN Interface and defines Frame trigger for TTCAN reception. This container is only included and valid if TTCAN is supported by the controller, enabled (see CanIfSupportTTCAN, ECUC_CanIf_00675), and a joblist is used for reception.
Configuration Parameters	

SWS Item	**ECUC_CanIf_00136 :**
Name	CanTTRxJoblistTimeMark
Description	Defines the point in time, when the joblist execution funciton (JLEF) shall

	be called for the referenced rx trigger. Value is given in cycle time. This parameter is only configurable if a joblist is enabled by parameter CanIfTTJobList.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: CanIfTTJoblist		

SWS Item	**ECUC_CanIf_00133 :**		
Name	CanIfTTRxHwObjectTriggerIdRef		
Description	This parameter refers to a particular TTCAN hardware receive object Trigger of a hardware object in the TTCAN Driver Module, which is referred via plain CAN parameter CANIF_HRH_HANDLETYPE_REF. This parameter is only configurable if a joblist is enabled by parameter CanIfTTJobList.		
Multiplicity	1		
Type	Reference to [CanTTHardwareObjectTrigger]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: CanIfTTJoblist		

No Included Containers

10.3 Published information

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

11 Not applicable requirements

[SWS_TtCanIf_99999]「These requirements are not applicable to this specification.」 (BSW)