Document Title	Specification of Time Synchronization over CAN
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	674

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R21-11

Document Change History

Date	Release	Changed by	Description
2021-11-25	R21-11	AUTOSAR Release Management	• CAN HW timestamping added • Hysteresis added for sequence counter validation
2020-11-30	R20-11	AUTOSAR Release Management	• Time Validation updated for gateways • Time out handling of Synchronized and Offset Time messages corrected • Post build variant value corrected for CanTSynGlobalTimeMasterConfirmationHandleId and CanTSynGlobalTimeSlaveHandleId
2019-11-28	R19-11	AUTOSAR Release Management	• Time Validation (draft) • Clarification regarding messages with stuck sequence counter • Clarification regarding cyclic operation entry after timebase startup • Clarification regarding transmission and reception of User Bytes • Changed Document Status from Final to published

2018-10-31	4.4.0	AUTOSAR Release Management	• Modifications to enhance the precision of Global Time Synchronization • Additional minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2017-12-08	4.3.1	AUTOSAR Release Management	• Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2016-11-30	4.3.0	AUTOSAR Release Management	• Offset message formats changed • Extended Offset message formats added • Immediate Time Synchronization message transmission • Various enhancements and corrections
2015-07-31	4.2.2	AUTOSAR Release Management	• CanTSyn_SetTransmissionMode changed to return "void" • Minor corrections / clarifications / editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	• Initial Release

Disclaimer

Table of Contents

1 Introduction and functional overview

The CanTSyn module handles the distribution of time information over CAN buses.

Just transmitting the time information from the master to the slaves in a broadcast CAN message has the disadvantage that the time value becomes inaccurate due to CAN specific effects like arbitration and BSW specific delays.

The concept proposes a two-step mechanism:

- In a first broadcast message (the so-called SYNC message), the second portion of the time information (t0r) is transmitted. The transmitting ECU, i.e. the Time Master, uses CAN low-level mechanisms like the "CAN transmit confirmation" to detect the point in time (t1r) when the message was actually transmitted, i.e. it takes a timestamp.

  A receiving ECU, i.e. the Time Slave, receives the message and uses CAN low-level mechanisms like the "CAN receive indication" to detect the point in time (t2r) when the message was actually received.

- In a second broadcast message (the so-called Follow-Up (FUP) message), the Time Master transmits the offset between the time information transmitted in the previous SYNC message and the actual detected transmission time. No timestamp is taken for the FUP message, neither on the transmitting nor on the receiving side.

- The Time Slave can now combine the information within the SYNC and within the FUP message and with its previously taken timestamp for the received SYNC message and determine the transmitted time information in a more precise way by just receiving one message and omitting timestamps.

Figure 1.1 shows the CAN Time Synchronization mechanism.

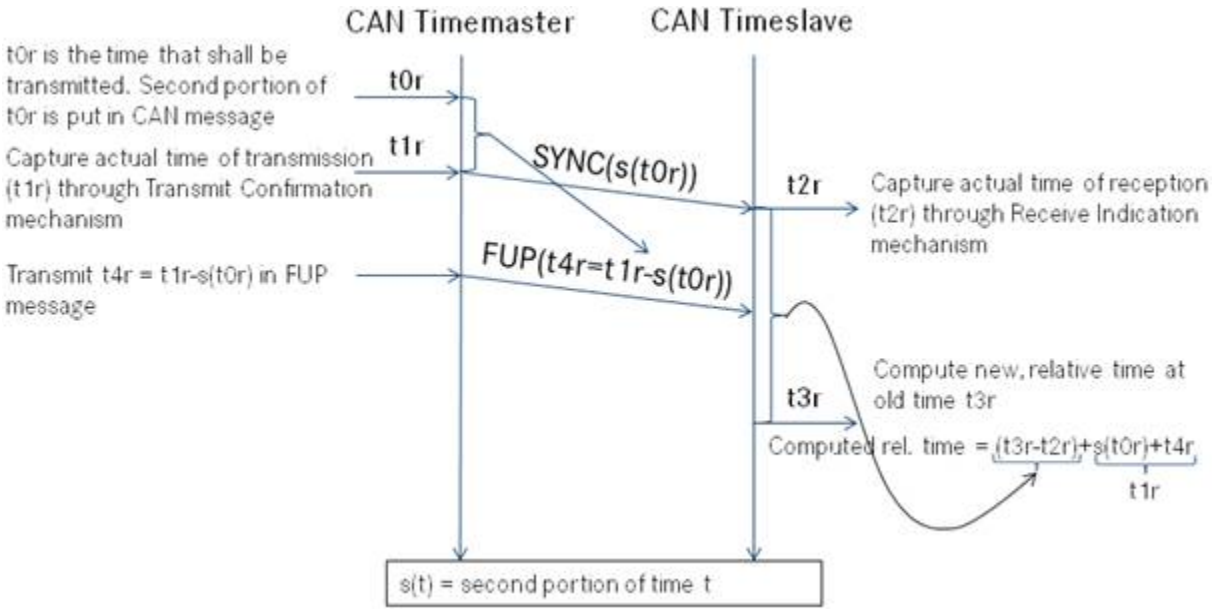

Figure 1.1: CAN Time Synchronization Mechanism

2 Acronyms and Abbreviations

This section lists module local abbreviations and definitions. For additional Time Synchronization related abbreviations and definitions refer to chapter 3 in the RS Time Synchronization [1]. For general terms and abbreviations refer to the AUTOSAR Glossary [2].

Abbreviation	Description
GTM	Global Time Master
BswM	BSW Mode Manager module
<Bus>TSyn	Bus specific Time Synchronization module
CAN FD	Controller Area Network (CAN) - Flexible Data Rate
CanTSyn	Time Synchronization over CAN module
CRC	Cyclic Redundancy Checksum
Debounce Time	Minimum gap between two TX messages with the same PDU
Det	Default Error Tracer module
DLC	Data Length Code
CanIf	CAN interface module
FUP message	Follow-Up message
OFNS message	Offset adjustment message
OFS message	Offset Synchronization message
OVS	Overflow Seconds value (field in FUP message)
SC	Sequence Counter in Time Synchronization messages
SGW	"Synchronized to Gateway" state of Time Synchronization
StbM	Synchronized Time-Base Manager
SYNC message	Time Synchronization message
Timesync	Time Synchronization

3 Related documentation

3.1 Input documents & related standards and norms

[1] Requirements on Time Synchronization
AUTOSAR_RS_TimeSync

[2] Glossary
AUTOSAR_TR_Glossary

[3] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral

[4] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral

[5] Specification of Synchronized Time-Base Manager

AUTOSAR_SWS_SynchronizedTimeBaseManager

[6] Specification of CRC Routines
AUTOSAR_SWS_CRCLibrary

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [3, SWS BSW General], which is also valid for CanTSyn.

Thus, the General Specification on Basic Software (SWS BSW General) shall be considered additionally and as required specification for CanTSyn.

4 Constraints and assumptions

4.1 Limitations

- The current version of CanTSyn does not support hardware timestamping capabilities.
  - The first consequence is that the Time Synchronization is less accurate due to RX-/TX-ISR latencies and execution time until the Virtual Local Time is retrieved.
  - The second consequence is the need of not nested interrupts in the CAN driver for the Global Time PDUs (i.e., it is strongly recommended not to invoke the TX confirmation and RX indication functions in polling mode).
- The Time Base in the SYNC and OFS messages is limited to 32 bit, wherefore the maximum supported time value is 4294967295 seconds (2^{32}-1).
- Time Masters, Time Gateways and Time Slaves shall work with a Time Base reference clock with a worst-case accuracy of 2μs.
- "CRC secured" in the context of this document refers to CRC integrity protection mechanism and does not imply that CRC is used as a cybersecurity solution.

4.2 Applicability to car domains

Automotive systems requiring a common Time Base for ECUs regardless of which bus system the ECUs are connected to.

5 Dependencies to other modules

The Time Synchronization over CAN (CanTSyn) has interfaces towards the Synchronized Time-Base Manager (StbM), the CAN Interface (CanIf), the BSW Mode Manager (BswM) and the Default Error Tracer (Det).

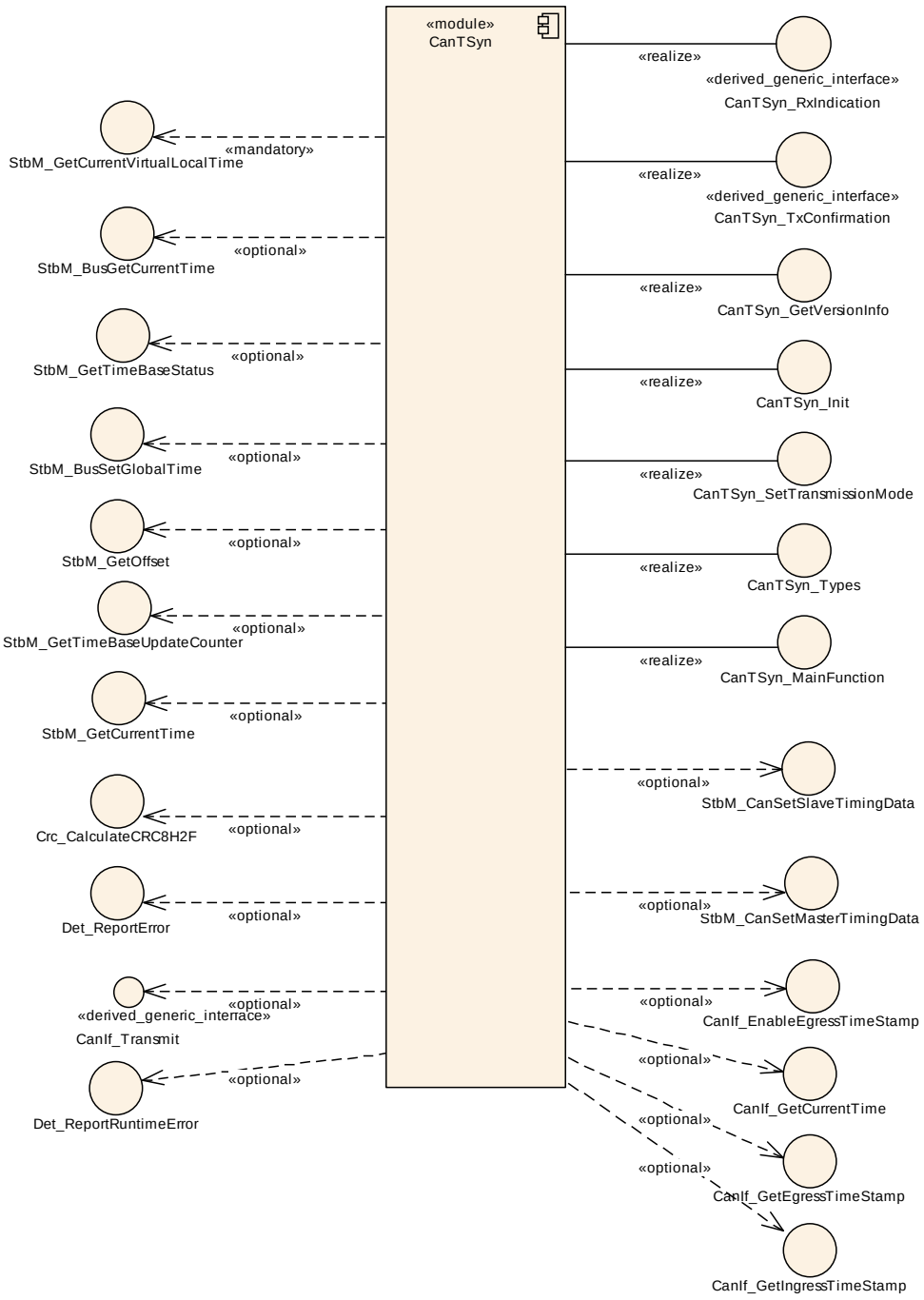

Figure 5.1: Module dependencies of the CanTSyn module

- StbM - Get and set the current time value
- CanIf - Receiving and transmitting messages

- BswM - Coordination of network access (via `CanTSyn_SetTransmissionMode`)
- DET - Reporting of development errors

5.1 File structure

5.1.1 Code file structure

For details, refer to the section 5.1.6 "Code file structure" of the SWS BSW General [3].

5.1.2 Header file structure

For details, refer to the section 5.1.7 "Header file structure" of the SWS BSW General [3].

6 Requirements Tracing

The following tables reference the requirements specified in [1, RS TimeSync] and [4, SRS BSWGeneral] and links to the fulfillment of these. Please note that if column "Satisfied by" is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[RS_TS_00002]	The Implementation of Time Synchronization shall maintain its own Time Base independently of the acting role.	[SWS_CanTSyn_00999]
[RS_TS_00003]	The TS shall initialize the Local Time Base with a configurable startup value	[SWS_CanTSyn_00003]
[RS_TS_00004]	The Implementation of Time Synchronization shall initialize the Global Time Base with a configurable startup value.	[SWS_CanTSyn_00003]
[RS_TS_00005]	The Implementation of Time Synchronization shall allow customers to have access to the Synchronized Time Base	[SWS_CanTSyn_00999]
[RS_TS_00006]	The Implementation of Time Synchronization shall provide time information to TSP modules	[SWS_CanTSyn_00999]

Requirement	Description	Satisfied by
[RS_TS_00007]	The Implementation of Time Synchronization shall synchronize the Time Base of a Time Slave, on reception of a Time Master value	[SWS_CanTSyn_00999]
[RS_TS_00008]	The Implementation of Time Synchronization shall continuously maintain its Time Bases based on a Time Base reference clock	[SWS_CanTSyn_00999]
[RS_TS_00009]	The Implementation of Time Synchronization shall maintain the synchronization status of a Time Base	[SWS_CanTSyn_00999]
[RS_TS_00010]	The Implementation of Time Synchronization shall allow customer on master side to set the Global Time	[SWS_CanTSyn_00999]
[RS_TS_00011]	The Implementation of Time Synchronization shall allow customers on master side to trigger time transmission by the TSP module	[SWS_CanTSyn_00999]
[RS_TS_00012]	The Implementation of Time Synchronization shall allow customers and TSP modules to read the offset value of an Offset Time Base	[SWS_CanTSyn_00999]
[RS_TS_00013]	The Implementation of Time Synchronization shall allow the customers and TSP modules to set the offset value of an Offset Master Time Base	[SWS_CanTSyn_00999]
[RS_TS_00014]	The Implementation of Time Synchronization shall allow customers to read User Data propagated via the TSP modules.	[SWS_CanTSyn_00999]
[RS_TS_00015]	The Implementation of Time Synchronization shall allow customers to set User Data propagated via the TSP modules.	[SWS_CanTSyn_00999]
[RS_TS_00016]	The Implementation of Time Synchronization shall notify customers about status events	[SWS_CanTSyn_00999]
[RS_TS_00017]	The Implementation of Time Synchronization shall notify customers about elapsed pre-defined time span.	[SWS_CanTSyn_00999]
[RS_TS_00018]	The Implementation of Time Synchronization shall support rate correction	[SWS_CanTSyn_00999]

Requirement	Description	Satisfied by
[RS_TS_00019]	The Implementation of Time Synchronization shall support damping offset correction	[SWS_CanTSyn_00999]
[RS_TS_00021]	The Implementation of Time Synchronization shall provide interfaces to query the synchronization status	[SWS_CanTSyn_00999]
[RS_TS_00024]	The Implementation of Time Synchronization shall support storage of the Time Base value at shutdown if configured as Time Master	[SWS_CanTSyn_00999]
[RS_TS_00025]	The Implementation of Time Synchronization shall provide fault detection mechanisms	[SWS_CanTSyn_00999]
[RS_TS_00026]	The Implementation of Time Synchronization shall provide to the customers a specific API per type of Time Base Resource	[SWS_CanTSyn_00999]
[RS_TS_00027]	The TS shall provide a bus independent customer interface	[SWS_CanTSyn_00999]
[RS_TS_00029]	The configuration of the Time Synchronization implementation shall allow the implementation to behave as a (vehicle wide) Time Master	[SWS_CanTSyn_00999]
[RS_TS_00030]	The configuration of the Time Synchronization implementation shall allow the implementation to behave as a Time Slave	[SWS_CanTSyn_00999]
[RS_TS_00031]	The configuration of the Time Synchronization implementation shall allow the implementation to behave as a Time Gateway	[SWS_CanTSyn_00999]
[RS_TS_00032]	The Implementation of Time Synchronization shall trigger registered customers	[SWS_CanTSyn_00999]
[RS_TS_00033]	The Implementation of Time Synchronization shall use a time format with a resolution of 1 ns	[SWS_CanTSyn_00999]
[RS_TS_00034]	The Implementation of Time Synchronization shall provide measurement data to the application	[SWS_CanTSyn_00137] [SWS_CanTSyn_00138] [SWS_CanTSyn_00139] [SWS_CanTSyn_00140] [SWS_CanTSyn_00141] [SWS_CanTSyn_00142]
[RS_TS_00035]	The Implementation of Time Synchronization shall provide a system service interface to applications	[SWS_CanTSyn_00999]
[RS_TS_00036]	The Implementation of Time Synchronization shall provide a bus independent customer interface	[SWS_CanTSyn_00999]

Requirement	Description	Satisfied by
[RS_TS_00037]	The configuration of the Time Synchronization implementation shall allow the interaction with different types of customers	[SWS_CanTSyn_00999]
[RS_TS_00038]	The Implementation of Time Synchronization shall copy Time Base information upon user request	[SWS_CanTSyn_00999]
[RS_TS_20031]	The Timesync over CAN module shall trigger Time Base Synchronization transmission	[SWS_CanTSyn_00025] [SWS_CanTSyn_00026] [SWS_CanTSyn_00028] [SWS_CanTSyn_00032] [SWS_CanTSyn_00035] [SWS_CanTSyn_00036] [SWS_CanTSyn_00038] [SWS_CanTSyn_00043] [SWS_CanTSyn_00044] [SWS_CanTSyn_00117] [SWS_CanTSyn_00118] [SWS_CanTSyn_00119] [SWS_CanTSyn_00120] [SWS_CanTSyn_00121] [SWS_CanTSyn_00122] [SWS_CanTSyn_00123] [SWS_CanTSyn_00124] [SWS_CanTSyn_00125] [SWS_CanTSyn_00136]
[RS_TS_20032]	The Timesync over CAN module shall provide the Time Base after reception of a valid Timesync/TS messages	[SWS_CanTSyn_00064] [SWS_CanTSyn_00072] [SWS_CanTSyn_00133] [SWS_CanTSyn_00135]
[RS_TS_20033]	The Timesync over CAN module shall support means to protect the Time synchronization protocol	[SWS_CanTSyn_00007] [SWS_CanTSyn_00015] [SWS_CanTSyn_00016] [SWS_CanTSyn_00017] [SWS_CanTSyn_00018] [SWS_CanTSyn_00031] [SWS_CanTSyn_00041] [SWS_CanTSyn_00048] [SWS_CanTSyn_00049] [SWS_CanTSyn_00050] [SWS_CanTSyn_00054] [SWS_CanTSyn_00055] [SWS_CanTSyn_00056] [SWS_CanTSyn_00111] [SWS_CanTSyn_00112] [SWS_CanTSyn_00126] [SWS_CanTSyn_00127] [SWS_CanTSyn_00128] [SWS_CanTSyn_00129]

Requirement	Description	Satisfied by
[RS_TS_20034]	The Timesync over CAN module shall detect and handle timeout and integrity errors in the Time Synchronization protocol	[SWS_CanTSyn_00027] [SWS_CanTSyn_00033] [SWS_CanTSyn_00037] [SWS_CanTSyn_00042] [SWS_CanTSyn_00057] [SWS_CanTSyn_00060] [SWS_CanTSyn_00061] [SWS_CanTSyn_00062] [SWS_CanTSyn_00063] [SWS_CanTSyn_00064] [SWS_CanTSyn_00065] [SWS_CanTSyn_00068] [SWS_CanTSyn_00071] [SWS_CanTSyn_00072] [SWS_CanTSyn_00076] [SWS_CanTSyn_00077] [SWS_CanTSyn_00078] [SWS_CanTSyn_00079] [SWS_CanTSyn_00080] [SWS_CanTSyn_00084] [SWS_CanTSyn_00085] [SWS_CanTSyn_00087] [SWS_CanTSyn_00088] [SWS_CanTSyn_00109] [SWS_CanTSyn_00110] [SWS_CanTSyn_00113] [SWS_CanTSyn_00114] [SWS_CanTSyn_00115] [SWS_CanTSyn_00116] [SWS_CanTSyn_00133] [SWS_CanTSyn_00143]
[RS_TS_20035]	The Timesync over CAN module shall support a protocol for precise time measurement and synchronization over CAN	[SWS_CanTSyn_00008] [SWS_CanTSyn_00010] [SWS_CanTSyn_00011] [SWS_CanTSyn_00015] [SWS_CanTSyn_00016] [SWS_CanTSyn_00017] [SWS_CanTSyn_00018] [SWS_CanTSyn_00025] [SWS_CanTSyn_00026] [SWS_CanTSyn_00027] [SWS_CanTSyn_00028] [SWS_CanTSyn_00029] [SWS_CanTSyn_00030] [SWS_CanTSyn_00031] [SWS_CanTSyn_00032] [SWS_CanTSyn_00033] [SWS_CanTSyn_00043] [SWS_CanTSyn_00044] [SWS_CanTSyn_00045] [SWS_CanTSyn_00047] [SWS_CanTSyn_00048] [SWS_CanTSyn_00049] [SWS_CanTSyn_00050] [SWS_CanTSyn_00054]

Requirement	Description	Satisfied by
		[SWS_CanTSyn_00055] [SWS_CanTSyn_00056] [SWS_CanTSyn_00057] [SWS_CanTSyn_00058] [SWS_CanTSyn_00059] [SWS_CanTSyn_00060] [SWS_CanTSyn_00061] [SWS_CanTSyn_00062] [SWS_CanTSyn_00063] [SWS_CanTSyn_00073] [SWS_CanTSyn_00075] [SWS_CanTSyn_00076] [SWS_CanTSyn_00078] [SWS_CanTSyn_00079] [SWS_CanTSyn_00080] [SWS_CanTSyn_00084] [SWS_CanTSyn_00085] [SWS_CanTSyn_00086] [SWS_CanTSyn_00087] [SWS_CanTSyn_00090] [SWS_CanTSyn_00091] [SWS_CanTSyn_00092] [SWS_CanTSyn_00093] [SWS_CanTSyn_00094] [SWS_CanTSyn_00095] [SWS_CanTSyn_00096] [SWS_CanTSyn_00099] [SWS_CanTSyn_00102] [SWS_CanTSyn_00103] [SWS_CanTSyn_00105] [SWS_CanTSyn_00106] [SWS_CanTSyn_00109] [SWS_CanTSyn_00110] [SWS_CanTSyn_00144] [SWS_CanTSyn_00145] [SWS_CanTSyn_00146] [SWS_CanTSyn_00147] [SWS_CanTSyn_00148] [SWS_CanTSyn_00149] [SWS_CanTSyn_00150] [SWS_CanTSyn_00151] [SWS_CanTSyn_00152] [SWS_CanTSyn_00153] [SWS_CanTSyn_00154]

Requirement	Description	Satisfied by
[RS_TS_20036]	The Timesync over CAN module shall use the time measurement and synchronization protocol to transmit and receive an offset value	[SWS_CanTSyn_00030] [SWS_CanTSyn_00035] [SWS_CanTSyn_00036] [SWS_CanTSyn_00037] [SWS_CanTSyn_00038] [SWS_CanTSyn_00039] [SWS_CanTSyn_00040] [SWS_CanTSyn_00041] [SWS_CanTSyn_00042] [SWS_CanTSyn_00043] [SWS_CanTSyn_00044] [SWS_CanTSyn_00046] [SWS_CanTSyn_00048] [SWS_CanTSyn_00049] [SWS_CanTSyn_00050] [SWS_CanTSyn_00054] [SWS_CanTSyn_00055] [SWS_CanTSyn_00056] [SWS_CanTSyn_00065] [SWS_CanTSyn_00066] [SWS_CanTSyn_00067] [SWS_CanTSyn_00068] [SWS_CanTSyn_00069] [SWS_CanTSyn_00070] [SWS_CanTSyn_00071] [SWS_CanTSyn_00074] [SWS_CanTSyn_00077] [SWS_CanTSyn_00078] [SWS_CanTSyn_00079] [SWS_CanTSyn_00080] [SWS_CanTSyn_00085] [SWS_CanTSyn_00086] [SWS_CanTSyn_00087] [SWS_CanTSyn_00111] [SWS_CanTSyn_00112] [SWS_CanTSyn_00113] [SWS_CanTSyn_00114] [SWS_CanTSyn_00126] [SWS_CanTSyn_00127] [SWS_CanTSyn_00128] [SWS_CanTSyn_00129]
[RS_TS_20037]	The Timesync over CAN module shall support user specific data within the time measurement and synchronization protocol	[SWS_CanTSyn_00011] [SWS_CanTSyn_00012] [SWS_CanTSyn_00013] [SWS_CanTSyn_00014]
[RS_TS_20038]	The Timesync over CAN module configuration shall allow the Implementation of Time Synchronization for CAN to support different roles for a Time Base	[SWS_CanTSyn_00108] [SWS_CanTSyn_00135]
[RS_TS_20039]	The Timesync over FlexRay module shall trigger Time Base Synchronization transmission	[SWS_CanTSyn_00999]

Requirement	Description	Satisfied by
[RS_TS_20040]	The Timesync over FlexRay module shall provide a Time Base after reception of a valid protocol information	[SWS_CanTSyn_00999]
[RS_TS_20041]	The Timesync over FlexRay module shall support means to protect the Time Synchronization protocol	[SWS_CanTSyn_00999]
[RS_TS_20042]	The Timesync over FlexRay module shall detect and handle timeout and integrity errors in the Time Synchronization protocol	[SWS_CanTSyn_00999]
[RS_TS_20043]	The Timesync over FlexRay module shall support a protocol for precise time measurement and synchronization over Flex Ray	[SWS_CanTSyn_00999]
[RS_TS_20044]	The Timesync over FlexRay module shall use the time measurement and synchronization protocol to transmit and receive an offset value	[SWS_CanTSyn_00999]
[RS_TS_20045]	The Timesync over FlexRay module shall support user specific data within the time measurement and synchronization protocol	[SWS_CanTSyn_00999]
[RS_TS_20046]	The configuration for Time synchronization over FlexRay shall allow the FlexRay Time Synchronization module to support different roles for a Time Base	[SWS_CanTSyn_00999]
[RS_TS_20047]	The Timesync over Ethernet module shall trigger Time Base Synchronization transmission	[SWS_CanTSyn_00999]
[RS_TS_20048]	The Timesync over Ethernet module shall support IEEE 802.1AS as well as AUTOSAR extensions	[SWS_CanTSyn_00999]
[RS_TS_20051]	The Timesync over Ethernet module shall detect and handle errors in synchronization protocol / communication	[SWS_CanTSyn_00999]
[RS_TS_20052]	The configuration of the Time Synchronization over Ethernet module shall allow the module to work as a Time Master	[SWS_CanTSyn_00999]
[RS_TS_20053]	The configuration of the Time Synchronization over Ethernet module shall allow the module to work as a Time Slave	[SWS_CanTSyn_00999]

Requirement	Description	Satisfied by
[RS_TS_20054]	The Implementation of the Time Synchronization shall evaluate and propagate Time Gateway relevant information	[SWS_CanTSyn_00999]
[RS_TS_20058]	The Timesync over Ethernet module shall provide the precision of Synchronized Time Bases	[SWS_CanTSyn_00999]
[RS_TS_20059]	The Timesync over Ethernet module shall access all communication ports belonging to Time Synchronization	[SWS_CanTSyn_00999]
[RS_TS_20060]	The Timesync over Ethernet module shall provide a Time Base after reception of a valid protocol information	[SWS_CanTSyn_00999]
[RS_TS_20061]	The Timesync over Ethernet module shall support means to protect the Time Synchronization protocol	[SWS_CanTSyn_00999]
[RS_TS_20062]	The Timesync over Ethernet module shall support user specific data within the time measurement and synchronization protocol	[SWS_CanTSyn_00999]
[RS_TS_20063]	The Timesync over Ethernet module shall use the Time Synchronization protocol for Synchronized Time Bases to transmit and receive Offset Time Bases	[SWS_CanTSyn_00999]
[RS_TS_20066]	The Timesync over Ethernet module shall support a static (pre)configuration of IEEE 802.1AS Pdelay	[SWS_CanTSyn_00999]
[RS_TS_20068]	The Timesync over CAN module shall support classic CAN and CAN FD	[SWS_CanTSyn_00010] [SWS_CanTSyn_00015] [SWS_CanTSyn_00016] [SWS_CanTSyn_00017] [SWS_CanTSyn_00018] [SWS_CanTSyn_00036] [SWS_CanTSyn_00041] [SWS_CanTSyn_00055] [SWS_CanTSyn_00071] [SWS_CanTSyn_00072] [SWS_CanTSyn_00077] [SWS_CanTSyn_00085] [SWS_CanTSyn_00111] [SWS_CanTSyn_00112] [SWS_CanTSyn_00130] [SWS_CanTSyn_00131] [SWS_CanTSyn_00132]

Requirement	Description	Satisfied by
[RS_TS_20069]	The TimeSync over Ethernet module shall provide read / write access to bus protocol specific parameters	[SWS_CanTSyn_00999]
[RS_TS_20070]	The Timesync over CAN module shall support hardware and software timestamping	[SWS_CanTSyn_00144] [SWS_CanTSyn_00147] [SWS_CanTSyn_00148] [SWS_CanTSyn_00150] [SWS_CanTSyn_00152] [SWS_CanTSyn_00153]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_CanTSyn_00088] [SWS_CanTSyn_00097] [SWS_CanTSyn_00100] [SWS_CanTSyn_00134]
[SRS_BSW_00337]	Classification of development errors	[SWS_CanTSyn_00097] [SWS_CanTSyn_00100] [SWS_CanTSyn_00134]
[SRS_BSW_00385]	List possible error notifications	[SWS_CanTSyn_00089]

7 Functional specification

This chapter defines the behavior of the Time Synchronization over CAN. The API of the module is defined in chapter 8, while the configuration is defined in chapter 10.

7.1 Overview

The Time Synchronization over CAN is responsible to realize the CAN specific Time Synchronization protocol.

Time Synchronization principles and common wording is described in the SWS Synchronized Time-Base Manager [5] and RS Time Synchronization [1].

7.2 Module Handling

This section contains description of auxiliary functionality of the Time Synchronization over CAN.

[SWS_CanTSyn_00135] ⌈If `CanTSyn` calls an API of the `StbM`, it shall use the Time Base ID of the Time Base referenced via the parameter `CanTSynSynchronizedTimeBaseRef` of the corresponding Time Domain.⌋*(RS_TS_20032, RS_TS_20038)*

7.2.1 Interrupt Handling

When transmitting or receiving a SYNC message, the current value of the Virtual Local Time needs to be captured in the RX indication / TX confirmation callbacks

- either in interrupt mode in context of the RX / TX interrupt
- or in polling mode in the main function (Note: it is strongly recommended not to use polling mode for Time Slaves).

Any delay between the occurrence of the interrupt itself and the determination of the current Virtual Local Time worsens the precision of either the transmitted or received Time Base.

Therefore, it is inevitable that these RX indication / TX confirmation callbacks establish a protection against interruptions immediately after being called (if called in context of the RX / TX interrupt with interrupt nesting disabled, this is implicitly ensured by the controller).

Thereafter only the necessary checks shall be made to determine that the message is a SYNC message (and to determine the Time Base ID if necessary). Once the Time Base ID and the SYNC message type are confirmed the current value of the Virtual Local Time is obtained from a function call to the StbM (still in the context of locked interrupts). Afterwards the interruption protection can be removed without having a negative impact on the precision.

As a consequence it might be possible that a snapshot of the Virtual Local Time is taken although the subsequent frame checks (e.g., CRC validation, SC validation) might fail and thus the snapshot becomes superfluous.

7.2.2 Initialization

The Time Synchronization over CAN is initialized via CanTSyn_Init. Except for CanTSyn_GetVersionInfo and CanTSyn_Init, the API functions of the Time Synchronization over CAN may only be called when the module has been properly initialized.

[SWS_CanTSyn_00003] ⌈A call to CanTSyn_Init initializes all internal variables and sets the Time Synchronization over CAN to the initialized state.⌋ *(RS_TS_00003, RS_TS_00004)*

[SWS_CanTSyn_00007] ⌈The Sequence Counter (SC) shall be initialized with 0.⌋ *(RS_TS_20033)*

7.2.3 Error Handling

[SWS_CanTSyn_00088] ⌈On errors and exceptions, the CanTSyn module shall not modify its current module state but shall simply report the error event.⌋(*RS_TS_20034, SRS_BSW_00323*)

7.3 Message Format

SYNC, FUP, OFS and OFNS messages are assigned to a dedicated message type "TimeSync".

SYNC, FUP, OFS and OFNS messages of the same Time Domain share the same CAN ID by using a multiplexed signal group. For different Time Domains the same CAN ID may be used if Timesync messages are sent by the same Time Master or Time Gateway. For different Time Domains different CAN IDs shall be used if Timesync messages are sent by different Time Masters or Time Gateways. The multiplexer is located at Byte 0, named as Type.

The usage of a CRC is optional. To ensure a great variability between several time observing units, the configuration decides of how to handle CRC secured Timesync messages if the receiver does not support the CRC calculation. Hence it might be possible, that a receiver is just using the given Time Base value without evaluating the CRC.

[SWS_CanTSyn_00008] ⌈The byte order for time value signals in Time Synchronization messages is "Big Endian".⌋(*RS_TS_20035*)

[SWS_CanTSyn_00010] ⌈The DLC of SYNC, FUP, OFS and OFNS messages is 8 for classic CAN.

The DLC of SYNC, FUP, OFS and OFNS messages is 16 for CAN FD if CanTSynUseExtendedMsgFormat is TRUE.⌋(*RS_TS_20035, RS_TS_20068*)

[SWS_CanTSyn_00011] ⌈Depending on its type Time Synchronization messages may contain User Data according to the given message format.
⌋(*RS_TS_20035, RS_TS_20037*)

[SWS_CanTSyn_00012] ⌈User Data shall be read consistently from incoming Time Synchronization messages that contain User Data Fields.⌋(*RS_TS_20037*)

[SWS_CanTSyn_00013] ⌈User Data shall be written consistently to outgoing Time Synchronization messages that contain User Data Fields.

If the number of User Data Fields in a Time Synchronization message is greater than the number of User Data Bytes provided by the StbM, the remaining User Data Fields shall be set to 0 (default value).⌋(*RS_TS_20037*)

[SWS_CanTSyn_00014] ⌈User Data shall be mapped to the `StbM_UserDataType`, where the byte number given in the message and by the `StbM_UserDataType` shall match (User Byte 0 mapped to `StbM_UserDataType.userByte0`, etc.).

`StbM_UserDataType.userDataLength` shall be set to the Time Synchronization message type specific number of User Bytes.⌋*(RS_TS_20037)*

7.3.1 SYNC and FUP Message

[SWS_CanTSyn_00015] ⌈SYNC not CRC secured message format:

Byte	Bit Position	Field Name	Field Value Range	Description
0		`Type`	`0x10`	
1		User Byte 1	default: 0	
2	7..4	`D`	0..15	Time Domain Id
	3..0	`SC`	0..15	Sequence Counter
3		User Byte 0	default: 0	
4-7		`SyncTimeSec`		32 bit LSB of the 48 bits seconds part of the time
If `CanTSynUseExtendedMsgFormat` = `TRUE`:				
8-15		reserved	always 0	

Table 7.1: SYNC not CRC secured message format

⌋*(RS_TS_20033, RS_TS_20035, RS_TS_20068)*

[SWS_CanTSyn_00016] ⌈FUP not CRC secured message format:

Byte	Bit Position	Field Name	Field Value Range	Description
0		`Type`	`0x18`	
1		User Byte 2	default: 0	
2	7..4	`D`	0..15	Time Domain Id
	3..0	`SC`	0..15	Sequence Counter
3	7..3	reserved	default: 0	
	2	`SGW`	`SyncToGTM` = 0 `SyncToSubDomain` = 1	
	1..0	`OVS`		Overflow of seconds
4-7		`SyncTimeNSec`		32 Bit time value in nanoseconds
If `CanTSynUseExtendedMsgFormat` = `TRUE`:				
8-15		reserved	always 0	

Table 7.2: FUP not CRC secured message format

⌋*(RS_TS_20033, RS_TS_20035, RS_TS_20068)*

[SWS_CanTSyn_00017] ⌈SYNC CRC secured message format:

Byte	Bit Position	Field Name	Field Value Range	Description
0		`Type`	`0x20`	
1		`CRC`		

2	7..4	D	0..15	Time Domain Id
	3..0	SC	0..15	Sequence Counter
3		User Byte 0	default: 0	
4-7		SyncTimeSec		32 bit LSB of the 48 bits seconds part of the time
If CanTSynUseExtendedMsgFormat = TRUE:				
8-15		reserved	always 0	

Table 7.3: SYNC CRC secured message format

⌋(*RS_TS_20033*, *RS_TS_20035*, *RS_TS_20068*)

[SWS_CanTSyn_00018] ⌈FUP CRC secured message format:

Byte	Bit Position	Field Name	Field Value Range	Description
0		Type	0x28	
1		CRC		
2	7..4	D	0..15	Time Domain Id
	3..0	SC	0..15	Sequence Counter
3	7..3	reserved	default: 0	
	2	SGW	SyncToGTM = 0 SyncToSubDomain = 1	
	1..0	OVS		Overflow of seconds
4-7		SyncTimeNSec		32 Bit time value in nanoseconds
If CanTSynUseExtendedMsgFormat = TRUE:				
8-15		reserved	always 0	

Table 7.4: FUP CRC secured message format

⌋(*RS_TS_20033*, *RS_TS_20035*, *RS_TS_20068*)

7.3.2 Offset Messages

Offset messages can be multiplexed with the Time Synchronization messages (using the same PDU, etc.).

For Classic CAN (CAN 2.0) two different Offset messages are used, OFS and OFNS. For both of them there are variants with and without a CRC field.

For CAN FD, if CanTSynUseExtendedMsgFormat is TRUE, the content of OFS and OFNS is merged into a single Extended OFS message (variants with and without a CRC field exist as well).

[SWS_CanTSyn_00132] ⌈CanTSynUseExtendedMsgFormat shall always be FALSE for CAN 2.0 buses.⌋(*RS_TS_20068*)

[SWS_CanTSyn_00130] ⌈If CanTSynUseExtendedMsgFormat is FALSE, then the Normal Offset Message Format shall be used, i.e., Offset Messages with message

Type 0x34, 0x44, 0x3C and 0x4C.
⌋(*RS_TS_20068*)

Note: For Normal Offset Message Format refer to chapter 7.3.2.1

[SWS_CanTSyn_00131] ⌈If CanTSynUseExtendedMsgFormat is TRUE, then the Extended Offset Message Format shall be used, i.e., Offset Messages with message Type 0x54 and 0x64.
⌋(*RS_TS_20068*)

Note: For Extended Offset Message Format refer to chapter 7.3.2.2

7.3.2.1 Normal Offset Messages

[SWS_CanTSyn_00126] ⌈OFS not CRC secured message format:

Byte	Bit Position	Field Name	Field Value Range	Description
0		Type	0x34	
1		User Byte 1	default: 0	
2	7..4	D	16..31	Time Domain Id
	3..0	SC		Sequence Counter
3		User Byte 0	default: 0	
4-7		OfsTimeSec		32 Bit offset time value in seconds

Table 7.5: OFS not CRC secured message format

⌋(*RS_TS_20033, RS_TS_20036*)

[SWS_CanTSyn_00127] ⌈OFNS not CRC secured message format:

Byte	Bit Position	Field Name	Field Value Range	Description
0		Type	0x3C	
1		User Byte 2	default: 0	
2	7..4	D	16..31	Time Domain Id
	3..0	SC	0..15	Sequence Counter
3	7..1	reserved	default: 0	
	0	SGW	SyncToGTM = 0 SyncToSubDomain = 1	
4-7		OfsTimeNSec		32 Bit offset time value in nanoseconds

Table 7.6: OFNS not CRC secured message format

⌋(*RS_TS_20033, RS_TS_20036*)

[SWS_CanTSyn_00128] ⌈OFS CRC secured message format:

Byte	Bit Position	Field Name	Field Value Range	Description
0		Type	0x44	
1		CRC		

2	7..4	D	16..31	Time Domain Id
	3..0	SC		Sequence Counter
3		User Byte 0	default: 0	
4-7		OfsTimeSec		32 Bit offset time value in seconds

Table 7.7: OFS CRC secured message format

⌋(*RS_TS_20033*, *RS_TS_20036*)

[SWS_CanTSyn_00129] ⌈OFNS CRC secured message format:

Byte	Bit Position	Field Name	Field Value Range	Description
0		Type	0x4C	
1		CRC		
2	7..4	D	16..31	Time Domain Id
	3..0	SC		Sequence Counter
3	7..1	reserved	default: 0	
	0	SGW	SyncToGTM = 0 SyncToSubDomain = 1	
4-7		OfsTimeNSec		32 Bit offset time value in nanoseconds

Table 7.8: OFNS CRC secured message format

⌋(*RS_TS_20033*, *RS_TS_20036*)

7.3.2.2 Extended Offset messages

If CanTSynUseExtendedMsgFormat is TRUE, the message layout of the Extended OFS message is as follows. A separate OFNS message is not required.

[SWS_CanTSyn_00111] ⌈OFS not CRC secured message format for CAN FD PDUs:

Byte	Bit Position	Field Name	Field Value Range	Description
0		Type	0x54	
1		User Byte 2	default: 0	
2	7..4	D	16..31	Time Domain Id
	3..0	SC		Sequence Counter
3	7..1	reserved	default: 0	
	0	SGW	SyncToGTM = 0 SyncToSubDomain = 1	
4		User Byte 0	default: 0	
5		User Byte 1	default: 0	
6		reserved	default: 0	
7		reserved	default: 0	
8-11		OfsTimeSec		32 Bit offset time value in seconds
12-15		OfsTimeNSec		32 Bit offset time value in nanoseconds

Table 7.9: OFS not CRC secured message format for CAN FD

⌋(*RS_TS_20033*, *RS_TS_20036*, *RS_TS_20068*)

[SWS_CanTSyn_00112] ⌈OFS CRC secured message format for CAN FD PDUs:

Byte	Bit Position	Field Name	Field Value Range	Description
0		Type	0x64	
1		CRC		
2	7..4	D	16..31	Time Domain
	3..0	SC		Sequence Counter
3	7..1	reserved	default: 0	
	0	SGW	SyncToGTM = 0 SyncToSubDomain = 1	
4		User Byte 0	default: 0	
5		User Byte 1	default: 0	
6		reserved	default: 0	
7		reserved	default: 0	
8-11		OfsTimeSec		32 Bit offset time value in seconds
12-15		OfsTimeNSec		32 Bit offset time value in nanoseconds

Table 7.10: OFS CRC secured message format for CAN FD

⌋(*RS_TS_20033*, *RS_TS_20036*, *RS_TS_20068*)

7.4 Acting as Time Master

A Time Master is an entity which is the master for a certain Time Base and which propagates this Time Base to a set of Time Slaves within a certain segment of a communication network, being a source for this Time Base.

If a Time Master is also the owner of the Global Time Base, the Time Base from which all further Time Bases are derived from, then it is the Global Time Master (refer to Figure 7.1). A Time Gateway typically consists of one Time Master port which is connected to one or more Time Slaves. When mapping time entities to real ECUs it has to be noted, that an ECU could be Time Master (or even Global Time Master) for one Time Base and Time Slave for another Time Base.

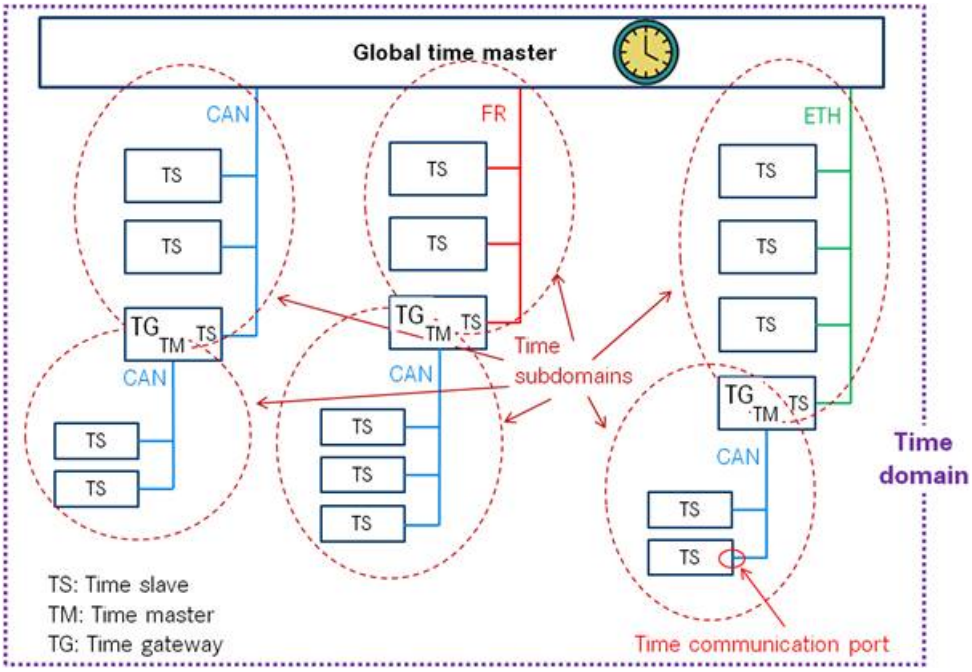

Figure 7.1: Terminology Example

[SWS_CanTSyn_00136] ⌈A master shall transmit SYNC, FUP, OFS and OFNS messages by calling `CanIf_Transmit` with the PduId derived via `CanTSynGlobalTimePduRef` of the corresponding Time Domain.⌋*(RS_TS_20031)*

7.4.1 SYNC and FUP message processing

[SWS_CanTSyn_00025] ⌈A Time Master shall start each Time Synchronization sequence for a Synchronized Time Base with a SYNC message.
⌋*(RS_TS_20031, RS_TS_20035)*

[SWS_CanTSyn_00026] ⌈A Time Master shall finish each Time Synchronization sequence for a Synchronized Time Base with a FUP message.
⌋*(RS_TS_20031, RS_TS_20035)*

[SWS_CanTSyn_00027] ⌈If a transmission of a SYNC or FUP message fails (`CanTSyn_TxConfirmation` is called with `E_NOT_OK`), `CanTSyn` shall reset the state machine to start with a new SYNC transmission again once it is due.⌋*(RS_TS_20034, RS_TS_20035)*

Note: No FUP message will be sent, if the SYNC message transmission fails.

[SWS_CanTSyn_00028] ⌈If configured as Time Master of a Synchronized Time Domain (refer to `CanTSynGlobalTimeDomain`) the `CanTSyn` module shall periodically transmit SYNC messages with the cycle `CanTSynGlobalTimeTxPeriod` if

- the `GLOBAL_TIME_BASE` bit within the timeBaseStatus is set
- and `CanTSynGlobalTimeTxPeriod` is unequal to 0

- and if the associated cyclicMsgResumeCounter is not running.

The cyclic transmission shall be started in the earliest possible CanTSyn_MainFunction call once the requirements above are fulfilled.⌋(*RS_TS_20031*, *RS_TS_20035*)

Note: "earliest possible" means:

- In the next CanTSyn_MainFunction, because GLOBAL_TIME_BASE is set outside the CanTSyn_MainFunction.
- In the current CanTSyn_MainFunction, when switching from immediate to cyclic transmission (because this decision is made inside the CanTSyn_MainFunction).

[SWS_CanTSyn_00029] ⌈The SYNC and FUP sequence shall not be interrupted, neither by Time Synchronization messages of the same Time Domain nor by Time Synchronization messages of other Time Domains if the same CAN ID is used for the Time Synchronization messages.⌋(*RS_TS_20035*)

[SWS_CanTSyn_00031] ⌈Depending on CanTSynGlobalTimeTxCrcSecured the SYNC / FUP message shall be of type:

CanTSynGlobalTimeTxCrcSecured Value	SYNC Message Type	FUP Message Type
CRC_NOT_SUPPORTED	0x10 SYNC not CRC secured message	0x18 FUP not CRC secured message
CRC_SUPPORTED	0x20 SYNC CRC secured message	0x28 FUP CRC secured message

Table 7.11

⌋(*RS_TS_20033*, *RS_TS_20035*)

[SWS_CanTSyn_00032] ⌈A transmitter of FUP messages (Time Master) is using as trigger condition for SYNC to FUP that the debounceCounter value reaches 0.⌋(*RS_TS_20031*, *RS_TS_20035*)

Note: Refer to chapter 7.4.4 for the use of the debounceCounter.

[SWS_CanTSyn_00033] ⌈Each transmission request of a SYNC message shall be monitored for a transmit confirmation timeout.

If CanTSyn_TxConfirmation is not called within 3 sec after transmission request, CanTSyn shall

- wait until CanTSyn_TxConfirmation is called (with E_OK or E_NOT_OK) and
- send no FUP message and
- instead reset the state machine to start with a new SYNC transmission once it is due.

⌋(*RS_TS_20034*, *RS_TS_20035*)

Note: A timeout of 3 sec is used to avoid an overflow of the SyncTimeNSec value in the FUP message (value range: 0 .. 2^{32} - 1 ns), if CanTSyn_TxConfirmation is called late

7.4.2 OFS message processing

[SWS_CanTSyn_00035] ⌈A Time Master shall start each Time Synchronization sequence for an Offset Time Base with an OFS message.
⌋*(RS_TS_20031, RS_TS_20036)*

[SWS_CanTSyn_00036] ⌈If CanTSynUseExtendedMsgFormat is FALSE, a Time Master shall finish each Time Synchronization sequence for an Offset Time Base with an OFNS message.⌋*(RS_TS_20031, RS_TS_20036, RS_TS_20068)*

Note: If CanTSynUseExtendedMsgFormat is TRUE, OFNS messages are not required.

[SWS_CanTSyn_00037] ⌈If the transmission of an OFS or an OFNS message fails (i.e., CanTSyn_TxConfirmation for the corresponding PDU is called with parameter result set to E_NOT_OK), the state machine shall be reset to start with a new OFS transmission again (once it is due).⌋*(RS_TS_20034, RS_TS_20036)*

Note: No OFNS message will be sent, if the OFS message transmission fails

[SWS_CanTSyn_00038] ⌈If configured as Time Master of an Offset Time Domain (refer to CanTSynGlobalTimeDomain) the CanTSyn module shall periodically transmit OFS messages with the cycle CanTSynGlobalTimeTxPeriod if

- the GLOBAL_TIME_BASE bit within the timeBaseStatus of the referenced Time Base CanTSynSynchronizedTimeBaseRef is set
- and CanTSynGlobalTimeTxPeriod is unequal to 0
- and if the associated cyclicMsgResumeCounter is not running.

The cyclic transmission shall be started in the earliest possible CanTSyn_MainFunction call once the requirements above are fulfilled.⌋*(RS_TS_20031, RS_TS_20036)*

Note: "earliest possible" means:

- In the next CanTSyn_MainFunction, because GLOBAL_TIME_BASE is set outside the CanTSyn_MainFunction.
- In the current CanTSyn_MainFunction, when switching from immediate to cyclic transmission (because this decision is made inside the CanTSyn_MainFunction).

[SWS_CanTSyn_00039] ⌈The OFS and OFNS sequence shall not be interrupted, neither by Time Synchronization messages of the same Time Domain nor by Time Synchronization messages of other Time Domains if the same CAN ID is used for the Time Synchronization messages.⌋*(RS_TS_20036)*

[SWS_CanTSyn_00040] ⌈A transmitter of OFNS messages (Time Master) is using as trigger condition for OFS to OFNS that the `debounceCounter` value reaches 0.⌋ *(RS_TS_20036)*

Note: Refer to chapter 7.4.4 for the use of the `debounceCounter`.

[SWS_CanTSyn_00041] ⌈Depending on `CanTSynGlobalTimeTxCrcSecured` the OFS / OFNS message shall be of type:

Bus Type	Value of Parameter `CanTSynGlobalTimeTxCrcSecured`	OFS Message Type	OFNS Message Type
CAN	`CRC_NOT_SUPPORTED`	`0x34` OFS not CRC secured message	`0x3C` OFNS not CRC secured message
	`CRC_SUPPORTED`	`0x44` OFS CRC secured message	`0x4C` OFNS CRC secured message
CAN FD (`CanTSyn-UseExtended-MsgFormat = TRUE`)	`CRC_NOT_SUPPORTED`	`0x54` OFS not CRC secured message	Not Available
	`CRC_SUPPORTED`	`0x64` OFS CRC secured message	

Table 7.12

⌋*(RS_TS_20033, RS_TS_20036, RS_TS_20068)*

[SWS_CanTSyn_00042] ⌈Each transmission request of an OFS message shall be monitored for a transmit confirmation timeout.

If `CanTSyn_TxConfirmation` is not called within 3 sec after transmission request, `CanTSyn` shall

- wait until `CanTSyn_TxConfirmation` is called (with `E_OK` or `E_NOT_OK`) and
- send no OFNS message and
- instead reset the state machine to start with a new OFS transmission once it is due.

⌋*(RS_TS_20034, RS_TS_20036)*

Note: A reset of the state machine in the event of a timeout avoids, that a possibly outdated Offset Time is sent. Instead the latest Offset Time via `StbM_GetOffset` is retrieved.

7.4.3 Transmission mode

[SWS_CanTSyn_00043] ⌈If `CanTSyn_SetTransmissionMode(Controller, Mode)` is called and parameter Mode equals `CANTSYN_TX_OFF`, all transmit requests

from CanTSyn shall be omitted on this CAN channel.⌋(*RS_TS_20031*, *RS_TS_20035*, *RS_TS_20036*)

[SWS_CanTSyn_00044] ⌈If CanTSyn_SetTransmissionMode(Controller, Mode) is called and parameter Mode equals CANTSYN_TX_ON, all transmit requests from CanTSyn on this CAN channel shall be able to be transmitted.⌋(*RS_TS_20031*, *RS_TS_20035*, *RS_TS_20036*)

7.4.4 Debounce Time

The debounce time shall inhibit transmission bursts of a specific CAN PDU. Inhibiting transmission bursts of Timesync messages on a specific CAN bus is not possible if multiple PDUs are used for multiple Time Domains since there is no inter-PDU debounce time configurable within the CanTSyn module.

[SWS_CanTSyn_00123] ⌈If CanTSynGlobalTimeDebounceTime is greater than 0 for a Time Base, CanTSyn shall always do debouncing for the corresponding Timesync PDUs as described below, otherwise CanTSyn shall not do any debouncing.
⌋(*RS_TS_20031*)

[SWS_CanTSyn_00124] ⌈CanTSynGlobalTimeDebounceTime represents the debounce value of a PDU specific debounceCounter that shall be started after the Timesync PDU has been successfully sent (i.e., CanTSyn_TxConfirmation for the corresponding PDU is called with parameter result set to E_OK).

CanTSyn shall decrement the debounceCounter value on each invocation of CanTSyn_MainFunction⌋(*RS_TS_20031*)

[SWS_CanTSyn_00125] ⌈A new Timesync PDU shall only be sent if the corresponding debounceCounter has a value equal or less than 0.⌋(*RS_TS_20031*)

Note: Since the decrement of the debounceCounter takes place in the CanTSyn_MainFunction call but the start of the counter takes place when the Timesync PDU has been sent (either in the subsequent CanTSyn_MainFunction call or in the transmit confirmation callback function) the effective debounce time will be equal or larger than CanTSynGlobalTimeDebounceTime. The extension of the debounce time shall be limited to the value of CanTSynMainFunctionPeriod

7.4.5 Immediate Time Synchronization

In addition to the cyclic Timesync message transmission, an immediate message transmission might be required.

Depending on configuration, the CanTSyn module checks on each CanTSyn_MainFunction call the necessity for a Timesync message transmission for each Time Base, where a Master Port belongs to.

[SWS_CanTSyn_00117] ⌈If CanTSynImmediateTimeSync is set to TRUE for a Time Base, CanTSyn shall check on each CanTSyn_MainFunction call by calling StbM_GetTimeBaseUpdateCounter, if the timeBaseUpdateCounter of the corresponding Time Base has changed.⌋*(RS_TS_20031)*

[SWS_CanTSyn_00118] ⌈If

- CanTSynImmediateTimeSync is set to TRUE and
- the timeBaseUpdateCounter of a Time Base has changed and
- the GLOBAL_TIME_BASE bit of the timeBaseStatus is set and
- the debounceCounter is 0 and
- no transmission of the corresponding PDU is pending (CanTSyn_TxConfirmation has been called with E_OK or E_NOT_OK),

CanTSyn shall trigger an immediate transmission of Time Synchronization messages for the corresponding Time Base.⌋*(RS_TS_20031)*

Note: timeBaseStatus can be obtained by StbM_GetTimeBaseStatus, StbM_BusGetCurrentTime or StbM_GetCurrentTime.

[SWS_CanTSyn_00119] ⌈If CanTSynImmediateTimeSync is set to TRUE, cyclicMsgResumeCounter and CanTSynCyclicMsgResumeTime shall be considered.⌋*(RS_TS_20031)*

[SWS_CanTSyn_00120] ⌈CanTSynCyclicMsgResumeTime represents the timeout value of a cyclicMsgResumeCounter that shall be started after an immediate transmission of a SYNC or an OFS message has been successfully completed (i.e., CanTSyn_TxConfirmation for the corresponding PDU is called with parameter result set to E_OK), asynchronously to the cyclic Timesync message transmission.

cyclicMsgResumeCounter shall be decremented on each invocation of CanTSyn_MainFunction, if no Timesync PDU is transmitted asynchronously.⌋*(RS_TS_20031)*

[SWS_CanTSyn_00121] ⌈If the cyclicMsgResumeCounter has reached a value equal or less than zero, CanTSyn shall resume cyclic Timesync message transmission by sending either a SYNC or OFS message.⌋*(RS_TS_20031)*

[SWS_CanTSyn_00122] ⌈If the cyclicMsgResumeCounter is started CanTSyn shall stop cyclic Timesync message transmission.⌋*(RS_TS_20031)*

7.4.6 Calculation and Assembling of Time Synchronization Messages

This chapter describes the workflow, how the items of a Time Synchronization message will be calculated (1st step) and how the message will be assembled (2nd step).

7.4.6.1 Global Time Calculation

In addition to the message fields (refer to chapter 7.3)

- SyncTimeSec
- OVS and
- SyncTimeNSec,

which are actually transmitted on the bus by the Time Master, this chapter defines and uses the following internal variables for calculation of the Global Time to be transmitted on the bus for Synchronized Time Domains:

- $T0_{SYNC}$: Global Time retrieved from StbM
- $T0_{SYNC\_ns}$: Nanosecond part of $T0_{SYNC}$
- $T0_{VLT}$: Virtual Local Time that corresponds to $T0_{SYNC}$. Retrieved together with $T0_{SYNC}$ from StbM
- $T1_{VLT}$: Egress timestamp of SYNC message relative to Virtual Local Time in StbM
- $T1_{CAN}$: Egress timestamp of SYNC message as captured by CAN controller HW
- T4: Correction value for $T0_{SYNC}$, which accounts for the delay between retrieving the time tuple [$T0_{SYNC}$;$T0_{VLT}$] from StbM and actually transmitting the SYNC message on the bus.
- $T_{currentTime\_CAN}$: Current local time as read from CAN controller HW when TX confirmation interrupt for SYNC message is processed in CanTSyn

Refer to Figure 1.1 and to sequence diagram Figure 9.2 for a better understanding of all steps of the Global Time Calculation sequence of the Time Master as specified in the requirements below.

[SWS_CanTSyn_00045]{OBSOLETE} ⌈The transmitter of a Synchronized Time Base (Time Master) shall perform the following steps to distribute the Synchronized Time Base exactly:

1. On transmission of SYNC message
   (a) Get current Synchronized Time Base's Time Tuple as [$T0_{SYNC}$;$T0_{VLT}$] via StbM_BusGetCurrentTime and write second portion of $T0_{SYNC}$ to SyncTimeSec
2. On SYNC message TX confirmation
   (a) Immediately establish a protection against interruptions and run the next step:
   (b) Retrieve current Virtual Local Time value as $T1_{VLT}$ via StbM_GetCurrentVirtualLocalTime

(c) The protection against interruptions may be released

(d) Calculate T4 for FUP message as T4 = $T0_{SYNC\_ns}$ + ($T1_{VLT}$ - $T0_{VLT}$) with $T0_{SYNC\_ns}$ as nanosecond portion of $T0_{SYNC}$

3. On transmission of FUP message

(a) Write seconds portion of T4 (T4 >= 1s) to OVS

(b) Write nanoseconds portion of T4 to SyncTimeNSec

⌋(*RS_TS_20035*)

[SWS_CanTSyn_00149]{DRAFT} ⌈If for a Synchronized Time Domain a cyclic or immediate transmission of a SYNC message is requested, the Time Master shall

1. get current Synchronized Time Base's Time Tuple as [$T0_{SYNC}$;$T0_{VLT}$] via StbM_BusGetCurrentTime and
2. call CanIf_Transmit with the seconds portion of $T0_{SYNC}$ written to SyncTimeSec field of the message data.

⌋(*RS_TS_20035*)

After a successful transmission of the SYNC message the CanTSyn captures the egress timestamp of the SYNC message.

[SWS_CanTSyn_00150]{DRAFT} ⌈Upon successful SYNC message TX confirmation for a Synchronized Time Domain and if no TX confirmation timeout has occurred (refer to [SWS_CanTSyn_00033]) the Time Master shall within the TX confirmation routine (CanTSyn_TxConfirmation)

- if HW timestamping is enabled,
  - Retrieve $T1_{CAN}$ as egress timestamp from CAN controller HW value via CanIf_GetEgressTimestamp
- else
  - Retrieve $T1_{VLT}$ as egress timestamp by reading current Virtual Local Time value via StbM_GetCurrentVirtualLocalTime

⌋(*RS_TS_20035, RS_TS_20070*)

Note: If SW timestamping is used, SW should immediately establish a protection against interruptions in the TX confirmation callback - unless interrupt nesting is disabled (when this is typically done implicitly by the controller). Any delay of StbM_GetCurrentVirtualLocalTime would impair precision.

Based on the egress timestamps $T1_{CAN}$ and $T1_{VLT}$, respectively, CanTSyn can calculate the delay between reading the tuple [$T0_{SYNC}$;$T0_{VLT}$] from StbM via StbM_BusGetCurrentTime and actual transmission of $T0_{SYNC}$ in the SYNC message on the bus.

T4, which accounts for that delay, is calculated in 3 different ways depending on

- whether HW timestamping is enabled or not and
- whether the StbM is using for internal time measurement the same time source as the CanTSyn for Virtual Local Time

This can be done either in the TX confirmation routine (CanTSyn_TxConfirmation) or in the subsequent CanTSyn_MainFunction invocation.

[SWS_CanTSyn_00151]{DRAFT} ⌈
If

- HW timestamping is disabled,

CanTSyn shall after successful capture of the egress timestamp (refer to [SWS_CanTSyn_00150]):

- calculate T4 = $T0_{SYNC\_ns}$ + ($T1_{VLT}$ - $T0_{VLT}$)

⌋(*RS_TS_20035*)

[SWS_CanTSyn_00152]{DRAFT} ⌈
If

- HW timestamping is enabled and
- CanTSyn is using for internal time measurement the same time source as the StbM for Virtual Local Time,

CanTSyn shall after successful capture of the egress timestamp (refer to [SWS_CanTSyn_00150])

- calculate T4 = $T0_{SYNC\_ns}$ + $T1_{VLT}$ - $T0_{VLT}$,
  with $T1_{VLT}$ = $T1_{CAN}$

⌋(*RS_TS_20035, RS_TS_20070*)

Note: In case CanTSyn uses for internal time measurement the same time source as the StbM for Virtual Local Time $T1_{CAN}$ equals $T1_{VLT}$.

[SWS_CanTSyn_00153]{DRAFT} ⌈
If

- HW timestamping is enabled and
- CanTSyn is using for internal time measurement a different time source as the StbM for Virtual Local Time,

CanTSyn shall after successful capture of the egress timestamp (refer to [SWS_CanTSyn_00150]):

1. establish a protection against interruptions
2. read $T_{currentTime\_CAN}$ via CanIf_GetCurrentTime,
3. read $T1_{VLT}$ via StbM_GetCurrentVirtualLocalTime,

4. release the protection against interruptions and
5. calculate T4 = $T0_{SYNC\_ns}$ + ($T1_{VLT}$ - $T0_{VLT}$) - ($T_{currentTime\_CAN}$ - $T1_{CAN}$)

⌋(*RS_TS_20035*, *RS_TS_20070*)

Note: In the above sequence protection against interruptions is important, because any interruption of the sequence of step 2 and step 3 would worsen the precision of T4 and hence the Global Time.

Note: The term $T_{currentTime\_CAN}$ - $T1_{CAN}$ compensates the interrupt delay from egress timestamping in HW until $T1_{VLT}$ can be sampled in `CanTSyn_TxConfirmation` via `StbM_GetCurrentVirtualLocalTime`.

[SWS_CanTSyn_00154]{DRAFT} ⌈If for a Synchronized Time Domain a FUP message is due, the Time Master shall

1. call `CanIf_Transmit` and
2. write the following data to the message:
   (a) seconds portion of T4 (T4 >= 1s) to the `OVS` field and
   (b) nanoseconds portion of T4 to the `SyncTimeNSec` field

⌋(*RS_TS_20035*)

[SWS_CanTSyn_00046] ⌈The transmitter of an Offset Time Base (Time Master) shall perform the following steps to distribute the Offset Time Base exactly:

- Retrieve current Offset Time via `StbM_GetOffset`
- Write seconds portion of the Offset Time to the `OfsTimeSec` field
- Write nanoseconds portion of the Offset Time to the `OfsTimeNSec` field

⌋(*RS_TS_20036*)

Note: OFS and OFNS messages are not time stamped.

7.4.6.2 OVS Calculation

[SWS_CanTSyn_00047] ⌈`OVS` shall be set within FUP messages if the transmitter detects a nanosecond overflow greater than the defined range of `StbM_TimeStampType.nanoseconds` (refer to [SWS_CanTSyn_00154]). The leftover part of seconds which does not fit into `StbM_TimeStampType.nanoseconds` shall be written to `OVS`.⌋ (*RS_TS_20035*)

7.4.6.3 SGW Calculation

[SWS_CanTSyn_00030] ⌈The SGW value (Time Gateway synchronization status) shall be retrieved from the Time Base synchronization status. If the SYNC_TO_GATEWAY bit within timeBaseStatus is not set the SGW value shall be SyncToGTM. Otherwise the SGW value shall be set to SyncToSubDomain.⌋(*RS_TS_20035*, *RS_TS_20036*)

7.4.6.4 Sequence Counter Calculation

[SWS_CanTSyn_00048] ⌈A Sequence Counter (SC) of 4 bit is representing numbers from 0 to 15 per Time Domain. The Sequence Counter shall be independent between SYNC and OFS messages and shall be incremented by 1 continuously on every transmission request of a SYNC or OFS message. It shall wrap around at 15 to 0 again.⌋ (*RS_TS_20033*, *RS_TS_20035*, *RS_TS_20036*)

[SWS_CanTSyn_00049] ⌈The Sequence Counter (SC) value for a FUP message shall be set to the SC value of the corresponding SYNC message. The SC value for an OFNS message shall be set to the SC value of the corresponding OFS message.⌋ (*RS_TS_20033*, *RS_TS_20035*, *RS_TS_20036*)

7.4.6.5 CRC Calculation

[SWS_CanTSyn_00050] ⌈The function Crc_CalculateCRC8H2F as defined in [6] shall be used to calculate the CRC if configured.
⌋(*RS_TS_20033*, *RS_TS_20035*, *RS_TS_20036*)

[SWS_CanTSyn_00054] ⌈The DataID shall be calculated as DataID = DataIDList [SC], where DataIDList is given by configuration for each message type (refer to configuration containers CanTSynGlobalTimeSyncDataIDList, CanTSynGlobalTimeFupDataIDList, CanTSynGlobalTimeOfsDataIDList and CanTSynGlobalTimeOfnsDataIDList).⌋(*RS_TS_20033*, *RS_TS_20035*, *RS_TS_20036*)

Note: A specific DataID out of a predefined DataIDList ensures the identification of data elements of Time Synchronization messages.

[SWS_CanTSyn_00055] ⌈If CanTSynUseExtendedMsgFormat is FALSE, the CRC shall be calculated over Time Synchronization message Byte 2 to Byte 7 and DataID, where Byte 2 is applied first, followed by the other bytes in ascending order, and Data ID last.

If CanTSynUseExtendedMsgFormat is TRUE, the CRC shall be calculated over Time Synchronization message Byte 2 to Byte 15 and DataID for Extended Timesync message formats, where Byte 2 is applied first, followed by the other bytes in ascending order, and DataID last.
⌋(*RS_TS_20033*, *RS_TS_20035*, *RS_TS_20036*, *RS_TS_20068*)

7.4.6.6 Message Assembling

[SWS_CanTSyn_00056] ⌈For each transmission of a Time Synchronization message the CanTSyn module shall assemble the message as follows:

1. Calculate OVS (FUP only)
2. Calculate SGW (FUP, OFNS and Extended OFS)
3. Calculate SC
4. Copy all data to the appropriate position within the related message
5. Calculate CRC (configuration dependent)

⌋(*RS_TS_20033*, *RS_TS_20035*, *RS_TS_20036*)

7.5 Acting as Time Slave

A Time Slave is an entity, which is the recipient for a certain Time Base within a certain segment of a communication network, being a consumer for this Time Base.

7.5.1 SYNC and FUP message processing

[SWS_CanTSyn_00057] ⌈The CanTSyn shall only accept a SYNC message with Type equal to 0x20 and a correct CRC value if CanTSynRxCrcValidated is configured to CRC_VALIDATED.⌋(*RS_TS_20034*, *RS_TS_20035*)

[SWS_CanTSyn_00058] ⌈The CanTSyn shall only accept a SYNC message with Type equal to 0x10 if CanTSynRxCrcValidated is configured to CRC_NOT_VALIDATED.⌋(*RS_TS_20035*)

[SWS_CanTSyn_00059] ⌈The CanTSyn shall only accept a SYNC message with Type equal to 0x10 or 0x20 if CanTSynRxCrcValidated is configured to CRC_IGNORED.⌋(*RS_TS_20035*)

[SWS_CanTSyn_00109] ⌈The CanTSyn shall only accept a SYNC message with Type equal to 0x10 or a SYNC message with Type equal to 0x20 and a correct CRC value if CanTSynRxCrcValidated is configured to CRC_OPTIONAL.
⌋(*RS_TS_20034*, *RS_TS_20035*)

[SWS_CanTSyn_00060] ⌈The CanTSyn shall only accept a FUP message with an identical Sequence Counter to the value of the corresponding SYNC message and Type equal to 0x28 and a correct CRC value if CanTSynRxCrcValidated is configured to CRC_VALIDATED.⌋(*RS_TS_20034*, *RS_TS_20035*)

[SWS_CanTSyn_00061] ⌈The CanTSyn shall only accept a FUP message with an identical Sequence Counter to the value of the corresponding SYNC message and

Type equal to 0x18 if CanTSynRxCrcValidated is configured to CRC_NOT_VALIDATED.⌋(*RS_TS_20034*, *RS_TS_20035*)

[SWS_CanTSyn_00062] ⌈The CanTSyn shall only accept a FUP message with an identical Sequence Counter to the value of the corresponding SYNC message and Type equal to 0x18 or 0x28 if CanTSynRxCrcValidated is configured to CRC_IGNORED.⌋(*RS_TS_20034*, *RS_TS_20035*)

[SWS_CanTSyn_00110] ⌈The CanTSyn shall only accept a FUP message with an identical Sequence Counter to the value of the corresponding SYNC message and Type equal to 0x18 or a FUP message with an identical sequence counter to the value of the corresponding SYNC message and Type equal to 0x28 and a correct CRC value if CanTSynRxCrcValidated is configured to CRC_OPTIONAL.
⌋(*RS_TS_20034*, *RS_TS_20035*)

[SWS_CanTSyn_00063] ⌈For each configured Time Slave (refer to CanTSynGlobalTimeSlave) the CanTSyn module shall observe the reception timeout CanTSynGlobalTimeFollowUpTimeout between the SYNC and its FUP message. If the reception timeout occurs the sequence shall be reset (i.e., waiting for a new SYNC message).⌋(*RS_TS_20034*, *RS_TS_20035*)

Note: The general timeout monitoring for the Time Base update is located in the StbM and not in the Timesync modules.

[SWS_CanTSyn_00064] ⌈For a valid pair of SYNC and FUP messages with successfully validated set of values SyncTimeSec, OVS and SyncTimeNSec a new Synclocal Time Tuple [TL_{Sync};$T3_{VLT}$] (refer to [5]), consisting of the Global Time value and the associated value of the Virtual Local Time, shall be calculated (refer to [SWS_CanTSyn_00146], [SWS_CanTSyn_00147], [SWS_CanTSyn_00148]) and forwarded to the StbM module via StbM_BusSetGlobalTime.⌋(*RS_TS_20032*, *RS_TS_20034*)

Note: For the detailed sequence of actions to derive a new Synclocal Time Tuple refer to Figure 9.4

7.5.2 OFS and OFNS message processing

[SWS_CanTSyn_00065] ⌈The CanTSyn shall only accept an OFS message with Type equal to 0x44 or 0x64 and a correct CRC value if CanTSynRxCrcValidated is configured to CRC_VALIDATED.⌋(*RS_TS_20034*, *RS_TS_20036*)

[SWS_CanTSyn_00066] ⌈The CanTSyn shall only accept an OFS message with Type equal to 0x34 or 0x54 if CanTSynRxCrcValidated is configured to CRC_NOT_VALIDATED.⌋(*RS_TS_20036*)

[SWS_CanTSyn_00067] ⌈The CanTSyn shall only accept an OFS message with Type equal to 0x34, 0x44, 0x54 or 0x64 if CanTSynRxCrcValidated is configured to CRC_IGNORED.⌋(*RS_TS_20036*)

[SWS_CanTSyn_00113] ⌈The CanTSynn shall only accept an OFS message with Type equal to 0x34 or 0x54 or an OFS message with Type equal to 0x44 or 0x64 and a correct CRC value if CanTSynRxCrcValidated is configured to CRC_OPTIONAL.⌋ *(RS_TS_20034, RS_TS_20036)*

[SWS_CanTSyn_00068] ⌈The CanTSyn shall only accept an OFNS message with an identical Sequence Counter to the value of the corresponding OFS message and Type equal to 0x4C and a correct CRC value if CanTSynRxCrcValidated is configured to CRC_VALIDATED.⌋ *(RS_TS_20034, RS_TS_20036)*

[SWS_CanTSyn_00069] ⌈The CanTSyn shall only accept an OFNS message with an identical Sequence Counter to the value of the corresponding OFS message and Type equal to 0x3C if CanTSynRxCrcValidated is configured to CRC_VALIDATED.⌋ *(RS_TS_20036)*

[SWS_CanTSyn_00070] ⌈The CanTSyn shall only accept an OFNS message with an identical Sequence Counter to the value of the corresponding OFS message and Type equal to 0x3C or 0x4C if CanTSynRxCrcValidated is configured to CRC_IGNORED.⌋ *(RS_TS_20036)*

[SWS_CanTSyn_00114] ⌈The CanTSyn shall only accept an OFNS message with an identical Sequence Counter to the value of the corresponding OFS message and Type equal to 0x3C or an OFNS message with an identical Sequence Counter to the value of the corresponding OFS message and Type equal to 0x4C and a correct CRC value if CanTSynRxCrcValidated is configured to CRC_OPTIONAL.
⌋*(RS_TS_20034, RS_TS_20036)*

[SWS_CanTSyn_00071] ⌈If CanTSynUseExtendedMsgFormat is FALSE, the CanTSyn shall observe for each configured Time Slave (CanTSynGlobalTimeSlave) the reception timeout CanTSynGlobalTimeFollowUpTimeout between the OFS and its OFNS message. If the reception timeout occurs the sequence shall be reset (i.e. waiting for a new OFS message).⌋ *(RS_TS_20034, RS_TS_20036, RS_TS_20068)*

Note: The general timeout monitoring for the Time Base update is located in the StbM and not in the Timesync modules.

[SWS_CanTSyn_00072] ⌈For a valid pair of OFS and OFNS messages and if CanTSynUseExtendedMsgFormat is FALSE, the CanTSyn shall calculate a new Time Tuple, consisting of the Offset Time value and the associated value of the Virtual Local Time, (according to [SWS_CanTSyn_00074]) and forward it to the StbM module via StbM_BusSetGlobalTime.

If CanTSynUseExtendedMsgFormat is TRUE, the CanTSyn shall calculate a new Time Tuple, consisting of the Offset Time value and the associated value of the Virtual Local Time, (according to [SWS_CanTSyn_00074]) after receiving a valid OFS message and forward it to the StbM module via StbM_BusSetGlobalTime.
⌋*(RS_TS_20032, RS_TS_20034, RS_TS_20068)*

[SWS_CanTSyn_00116] ⌈On an invocation of `StbM_BusSetGlobalTime` the parameter `pathDelay` of the `measureDataPtr` structure shall be set to 0.⌋*(RS_TS_-20034)*

7.5.3 Validation and Disassembling of Time Synchronization Messages

This chapter describes the workflow, how the items of a Time Synchronization message will be validated (1st step) and how the message will be disassembled (2nd step).

7.5.3.1 Global Time Calculation

In addition to the message fields (refer to chapter section 7.3)

- `SyncTimeSec`
- `OVS` and
- `SyncTimeNSec`,

which are actually received from the bus by the Time Master, this chapter defines and uses the following internal variables for calculation of the Global Time to be transmitted on the bus for Synchronized Time Domains:

- T0: Global Time (seconds portion) received from Time Master in SYNC message
- TL_{Sync}: Local Instance of Global Time calculated by Time Slave
- $T2_{VLT}$: Ingress timestamp of SYNC message relative to Virtual Local Time in `StbM`
- $T2_{CAN}$: Ingress timestamp of SYNC message as captured by CAN controller HW.
- $T3_{VLT}$: Current time relative to Virtual Local Time when FUP message is processed
- $T3_{CAN}$: Current time read from CAN controller HW when FUP message is processed
- T4: Correction value for T0 as received from the Time Master. It is calculated from values of `OVS` and `SyncTimeNSec` field in the FUP message.

Refer to Figure 1.1 and to sequence diagram Figure 9.4 for a better understanding of all steps of the Global Time Calculation sequence of the Time Slave as specified in the requirements below.

[SWS_CanTSyn_00073]{OBSOLETE} ⌈The receiver of a Synchronized Time Base shall perform the following steps to retrieve the Synchronized Time Base exactly:

1. On SYNC message RX indication, which delivers Synchronized Time Base part T0:

(a) Immediately establish a protection against interruptions and run the next step directly afterwards:

(b) Retrieve the current Virtual Local Time value as $T2_{VLT}$ via `StbM_GetCurrentVirtualLocalTime`

(c) The protection against interruptions may be released

2. On FUP message reception (either in RX indication or in the subsequent Main Function invocation), which delivers Synchronized Time Base part T4 = (`OVS` + `SyncTimeNSec`), retrieve current Virtual Local Time value as $T5_{VLT}$ via `StbM_GetCurrentVirtualLocalTime`
3. Calculate the Time Tuple [T5;$T5_{VLT}$] to update the Time Slave's Local Time Base: T5 = T0 + T4 + ($T5_{VLT}$ - $T2_{VLT}$).

⌋(*RS_TS_20035*)

[SWS_CanTSyn_00144]{DRAFT} ⌈Upon SYNC message RX indication for a Synchronized Time Domain, the Time Slave shall within the RX indication routine (`CanTSyn_RxIndication`)

1. - If HW timestamping is enabled,
     - Retrieve $T2_{CAN}$ as ingress timestamp from CAN controller HW value via `CanIf_GetIngressTimestamp`
   - else
     - Retrieve $T2_{VLT}$ as ingress timestamp by reading current Virtual Local Time value via `StbM_GetCurrentVirtualLocalTime`
2. Retrieve T0 from the SYNC message data

⌋(*RS_TS_20035, RS_TS_20070*)

Note: If SW timestamping is used, SW should immediately establish a protection against interruptions in the RX indication callback until $T2_{VLT}$ is retrieved (if called in context of the RX interrupt with interrupt nesting disabled, interrupt protection is typically implicitly ensured by the controller). Immediately protecting against interruptions means that there shall be no frame checks before. Once the interrupts are locked, it is ok to check whether the received message is a SYNC message for which a snapshot of the Virtual Local Time shall be taken, but no other frame checks (e.g., CRC validation, SC validation, etc.) shall be done before taking $T2_{VLT}$. Once $T2_{VLT}$ has been sampled it is ok to remove the protection against interruptions and to make the necessary validations. This means that $T2_{VLT}$ will be taken even if the succeeding validations fail and thus making the snapshot superfluous.

[SWS_CanTSyn_00145]{DRAFT} ⌈Upon reception of a FUP message, `CanTSyn` shall

1. retrieve the following data from the FUP message

- the OVS field and
- the SyncTimeNSec field

2. calculate T4 = OVS + SyncTimeNSec

in the RX indication routine (CanTSyn_RxIndication).⌋(*RS_TS_20035*)

Based on the ingress timestamp $T2_{VLT}$ (or $T2_{CAN}$ respectively), CanTSyn can determine the value for the local instance of the Global Time for the Time Slave, TL_{Sync}.

TL_{Sync} is calculated in 3 different ways depending on

- whether HW timestamping is enabled or not and
- whether StbM is using for internal time measurement the same time source as the CanTSyn for Virtual Local Time

This can be done either in the TX confirmation routine (CanTSyn_TxConfirmation) or in the subsequent CanTSyn_MainFunction invocation.

[SWS_CanTSyn_00146]{DRAFT} ⌈If

- HW timestamping is disabled,

CanTSyn shall after successful capture of T4 (refer to [SWS_CanTSyn_00145]):

1. capture the current Virtual Local Time via StbM_GetCurrentVirtualLocal-Time from StbM as $T3_{VLT}$ and
2. calculate $TL_{Sync} = (T0 + T4) + (T3_{VLT} - T2_{VLT})$

⌋(*RS_TS_20035*)

[SWS_CanTSyn_00147]{DRAFT} ⌈If

- HW timestamping is enabled and
- CanTSyn is using for internal time measurement the same time source as the StbM for Virtual Local Time,

CanTSyn shall after successful capture of T4 (refer to [SWS_CanTSyn_00145]):

1. retrieve current time from CAN controller HW value via CanIf_GetCurrent-Time as $T3_{CAN}$,
2. set $T2_{VLT} = T2_{CAN}$,
3. set $T3_{VLT} = T3_{CAN}$ and
4. calculate $TL_{Sync} = (T0 + T4) + (T3_{VLT} - T2_{VLT})$

⌋(*RS_TS_20035, RS_TS_20070*)

Note: In case CanTSyn uses for internal time measurement the same time source as the StbM for Virtual Local Time, i.e., Virtual Local Time is read from CAN controller HW, $T3_{VLT}$ and $T2_{VLT}$ equal $T3_{CAN}$ and $T2_{CAN}$, respectively.

[SWS_CanTSyn_00148]{DRAFT} ⌈If

- HW timestamping is enabled and
- CanTSyn is using for internal time measurement a different time source as the StbM for Virtual Local Time,

CanTSyn shall after successful capture of T4 (refer to [SWS_CanTSyn_00145]):

1. establish a protection against interruptions
2. retrieve current time from CAN controller HW value via CanIf_GetCurrentTime as $T3_{CAN}$,
3. capture the current Virtual Local Time via StbM_GetCurrentVirtualLocalTime from StbM as $T3_{VLT}$
4. release the protection against interruptions
5. calculate $T2_{VLT} = T3_{VLT} - (T3_{CAN} - T2_{CAN})$,
6. calculate $TL_{Sync} = (T0 + T4) + (T3_{CAN} - T2_{CAN})$

⌋*(RS_TS_20035, RS_TS_20070)*

Note: In the above sequence protection against interruptions is important, because any interruption of the sequence of step 2 and step 3 would worsen the precision of local instance of the Global Time, which depends on time tuple $[TL_{Sync};T3_{VLT}]$.

[SWS_CanTSyn_00074] ⌈The receiver of an Offset Time Base shall perform the following steps to assemble the Offset Time:

1. Get seconds portion of the Offset Time out of OfsTimeSec
2. Get nanoseconds portion of the Offset Time out of OfsTimeNSec
3. Retrieve current Virtual Local Time value via StbM_GetCurrentVirtualLocalTime

⌋*(RS_TS_20036)*

Note: OFS and OFNS messages are not time stamped.

7.5.3.2 OVS Consideration

[SWS_CanTSyn_00075] ⌈OVS (FUP only) shall be considered on the receiver side to retrieve the second portion of the received Synchronized Time Base.⌋*(RS_TS_20035)*

7.5.3.3 SGW Calculation

[SWS_CanTSyn_00133] ⌈If the SGW value (FUP, OFNS and Extended OFS) is set to SyncToSubDomain, the SYNC_TO_GATEWAY bit within timeBaseStatus shall be set to TRUE. Otherwise, it shall be set to FALSE.⌋*(RS_TS_20032, RS_TS_20034)*

7.5.3.4 Sequence Counter Validation

[SWS_CanTSyn_00076] ⌈The Sequence Counter of each SYNC message must match to the Sequence Counter of the next incoming FUP message of the same Time Domain. Otherwise, the contents of the already received SYNC message shall be discarded and the received FUP message shall be ignored.
⌋*(RS_TS_20034, RS_TS_20035)*

[SWS_CanTSyn_00077] ⌈If CanTSynUseExtendedMsgFormat is FALSE, the Sequence Counter of each OFS message must match to the Sequence Counter of the next incoming OFNS message of the same Time Domain. If the SCs do not match, the received OFNS message shall be ignored and the contents of the already received OFS message shall be discarded.⌋*(RS_TS_20034, RS_TS_20036, RS_TS_20068)*

[SWS_CanTSyn_00078] ⌈The Sequence Counter Jump Width between two consecutive SYNC or two consecutive OFS messages of the same Time Domain shall be greater than 0 and smaller than or equal to CanTSynGlobalTimeSequenceCounterJumpWidth. Otherwise, a Time Slave shall ignore the respective SYNC / OFS message.

If the CanTSynGlobalTimeSequenceCounterJumpWidth value is set to 0, the Time Slave shall not do Sequence Counter Jump Width checks.⌋*(RS_TS_20034, RS_TS_20035, RS_TS_20036)*

[SWS_CanTSyn_00079] ⌈Upon reception of a SYNC (or OFS) message a Time Slave shall check the Sequence Counter of the received message per Time Domain against the configured value of CanTSynGlobalTimeSequenceCounterJumpWidth (according to [SWS_CanTSyn_00078]), unless it is the first message

- at Startup or
- after a Time Base update timeout has been detected (TIMEOUT bit set in Time Base synchronization status timeBaseStatus).

⌋*(RS_TS_20034, RS_TS_20035, RS_TS_20036)*

Note: There are scenarios when it makes sense to skip the check of the Sequence Counter Jump Width, e.g. at startup (Time Slaves start asynchronously to the Time Master) or after a message timeout to allow for Sequence Counter (re-)synchronization. In case of a timeout the error has been detected already by the timeout monitoring, there is no benefit in generating a subsequent error by the jump width check.

Note: According to [SWS_CanTSyn_00078] the Sequence Counter validation will still discard messages with a Sequence Counter Jump Width being zero (i.e., stuck Sequence Counter) during Time Base update timeout.

[SWS_CanTSyn_00143] ⌈While a Time Base Timeout is present (`TIMEOUT` bit is set in Time Base synchronization status `timeBaseStatus`), `CanTSyn` shall discard SYNC/FUP (or OFS/OFNS) messages until it has successfully validated (refer to [SWS_CanTSyn_00078]) `n` consecutive SYNC/FUP (or OFS/OFNS) message pairs (`n` is given by the parameter `CanTSynGlobalTimeSequenceCounterHysteresis`).⌋ *(RS_TS_20034)*

Note: [SWS_CanTSyn_00143] improves robustness against a scenario with a buggy master implementation or injection of invalid master messages (sequence counter increments greater than `CanTSynGlobalTimeSequenceCounterJumpWidth`. In such a scenario any valid message pair would cause the Time Slave to leave the Timeout state (refer to [SWS_CanTSyn_00079]) although the sequence counter is not incremented correctly. An additional hysteresis avoids this.

7.5.3.5 CRC Validation

[SWS_CanTSyn_00080] ⌈The function `Crc_CalculateCRC8H2F` as defined in [6] shall be used to validate the `CRC` if configured.
⌋*(RS_TS_20034, RS_TS_20035, RS_TS_20036)*

[SWS_CanTSyn_00084] ⌈The `DataID` shall be calculated as `DataID = DataIDList[SC]`, where `DataIDList` is given by configuration for each message Type.⌋ *(RS_TS_20034, RS_TS_20035)*

Note: A specific `DataID` out of a predefined `DataIDList` ensures the identification of data elements of time synchronization messages.

[SWS_CanTSyn_00085] ⌈If `CanTSynUseExtendedMsgFormat` is `FALSE`, the `CRC` shall be calculated over Time Synchronization message Byte 2 to Byte 7 and `DataID`, where Byte 2 is applied first, followed by the other Bytes in ascending order, and `DataID` last.

If `CanTSynUseExtendedMsgFormat` is `TRUE`, the `CRC` shall be calculated over Time Synchronization message Byte 2 to Byte 15 and `DataID` for Extended Timesync message formats, where Byte 2 is applied first, followed by the other bytes in ascending order, and `DataID` last.
⌋*(RS_TS_20034, RS_TS_20035, RS_TS_20036, RS_TS_20068)*

7.5.3.6 Message Disassembling

[SWS_CanTSyn_00086] ⌈For each received Time Synchronization message the `CanTSyn` shall validate the message as follows (all conditions must match):

1. Type matches depending on the CanTSynRxCrcValidated parameter
2. SC value is within the accepted range (refer to [SWS_CanTSyn_00078] and [SWS_CanTSyn_00079])
3. D matches to the defined Time Domain range for each Type
4. D matches to one of the configured Time Domains (given by parameter CanTSynGlobalTimeDomainId)
5. SyncTimeNSec (FUP / OFNS / Extended OFS only) matches the defined range of StbM_TimeStampType.nanoseconds.
6. CRC (including DataID) matches depending on the CanTSynRxCrcValidated parameter

⌋(*RS_TS_20035*, *RS_TS_20036*)

[SWS_CanTSyn_00087] ⌈For each received Time Synchronization message the CanTSyn shall disassemble the message after successful validation (refer to [SWS_CanTSyn_00086]).⌋(*RS_TS_20034*, *RS_TS_20035*, *RS_TS_20036*)

7.6 Time Recording

7.6.1 Global Time Precision Measurement

[SWS_CanTSyn_00115] ⌈On an invocation of StbM_BusSetGlobalTime the parameter pathDelay of the measureDataPtr structure shall be set to 0.
⌋(*RS_TS_20034*)

7.6.2 Time Validation

[SWS_CanTSyn_00137] ⌈The CanTSyn shall support Time Validation, if CanTSynTimeValidationSupport set to TRUE.⌋(*RS_TS_00034*)

[SWS_CanTSyn_00138] ⌈
If

- CanTSynTimeValidationSupport is enabled and
- CanTSynEnableTimeValidation for the Time Domain is enabled

CanTSyn shall do time recording for Time Validation for that Time Domain
⌋(*RS_TS_00034*)

[SWS_CanTSyn_00139] ⌈
If

- time recording for Time Validation is enabled for a Time Domain (refer to [SWS_CanTSyn_00115] and [SWS_CanTSyn_00116]) and
- CanTSyn is configured as Time Slave for that Time Domain,

CanTSyn shall call StbM_CanSetSlaveTimingData upon successful reception of a FUP message.

StbM_CanSetSlaveTimingData shall be called after StbM_BusSetGlobalTime. ⌋(*RS_TS_00034*)

Note: StbM_BusSetGlobalTime shall be called first, because it updates the Synclocal Time Tuple (refer to [5]), which is required by StbM_CanSetSlaveTimingData.

[SWS_CanTSyn_00140] ⌈Upon invocation of StbM_CanSetSlaveTimingData CanTSyn shall pass following values

- the sequence counter value from the transmitter (Time Master),
- the segment id of the physical channel on which the SYNC message has been received (refer to parameter CanTSynGlobalTimeNetworkSegmentId)
- $T2_{VLT}$ as syncIngressTimestamp for the SYNC message (refer to step 1 in [SWS_CanTSyn_00144], [SWS_CanTSyn_00147] and [SWS_CanTSyn_00148]),
- T0 + T4 as preciseOriginTimestamp received from the Time Master (refer to [SWS_CanTSyn_00144] and [SWS_CanTSyn_00145])

to the function by the parameter measureDataPtr.

Struct members

- measureDataPtr→referenceLocalTimestamp and
- measureDataPtr→referenceGlobalTimestamp

shall be passed as 0.
⌋(*RS_TS_00034*)

Note: The CanTSyn passes 0 to avoid undefined values. The structure members referenceLocalTimestamp and referenceGlobalTimestamp will be set by the StbM via StbM_CanSetSlaveTimingData internally (refer to [SWS_StbM_00471] in [5]).

[SWS_CanTSyn_00141] ⌈
If

- time recording for Time Validation is enabled for a Time Domain (refer to [SWS_CanTSyn_00115] and [SWS_CanTSyn_00115]) and
- CanTSyn is configured as Time Master for that Time Domain

CanTSyn shall call StbM_CanSetMasterValidationData upon successful transmission of a SYNC message).
⌋(*RS_TS_00034*)

[SWS_CanTSyn_00142] ⌈Upon invocation of StbM_CanSetMasterValidationData CanTSyn shall pass the following data

- the sequence counter as sent in the SYNC message
- the segment id of the physical channel on which the SYNC message has been sent (refer to parameter CanTSynGlobalTimeNetworkSegmentId)
- $T1_{VLT}$ as the syncEgressTimestamp of SYNC message (refer to [SWS_CanTSyn_00149], [SWS_CanTSyn_00152] and [SWS_CanTSyn_00153]),
- $T0_{SYNC}$ + ($T1_{VLT}$ - $T0_{VLT}$) as precise preciseOriginTimestamp (refer to [SWS_CanTSyn_00149], [SWS_CanTSyn_00151], [SWS_CanTSyn_00152] and [SWS_CanTSyn_00153]),

to the function by the parameter measureDataPtr.
⌋(*RS_TS_00034*)

7.7 Error Classification

Section 7.2 "Error Handling" of the document "General Specification of Basic Software Modules" [3] describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.7.1 Development Errors

[SWS_CanTSyn_00089] ⌈

Type of error	*Related error code*	*Error value*
API service called with wrong PDU or SDU	CANTSYN_E_INVALID_PDUID	0x01
API service used in un-initialized state	CANTSYN_E_UNINIT	0x02
A pointer is NULL	CANTSYN_E_NULL_POINTER	0x03
CanTSyn initialization failed	CANTSYN_E_INIT_FAILED	0x04
API called with invalid parameter	CANTSYN_E_PARAM	0x05
Invalid Controller index	CANTSYN_E_INV_CTRL_IDX	0x06

⌋(*SRS_BSW_00385*)

7.7.2 Runtime Errors

There are no runtime errors.

7.7.3 Transient Faults

There are no transient faults.

7.7.4 Production Errors

There are no production errors.

7.7.5 Extended Production Errors

There are no extended production errors.

8 API specification

8.1 Imported types

In this section all types included from the following files are listed:

[SWS_CanTSyn_00090] ⌈

Module	*Header File*	*Imported Type*
Can	Can_GeneralTypes.h	Can_TimeStampType (draft)
ComStack_Types	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
StbM	Rte_StbM_Type.h	StbM_CanTimeMasterMeasurementType
	Rte_StbM_Type.h	StbM_CanTimeSlaveMeasurementType
	Rte_StbM_Type.h	StbM_SynchronizedTimeBaseType
	Rte_StbM_Type.h	StbM_TimeBaseStatusType
	Rte_StbM_Type.h	StbM_TimeStampShortType
	Rte_StbM_Type.h	StbM_TimeStampType
	Rte_StbM_Type.h	StbM_UserDataType
	StbM.h	StbM_MeasurementType
	StbM.h	StbM_VirtualLocalTimeType
Std	Std_Types.h	Std_ReturnType

▽

△

Module	Header File	Imported Type
	Std_Types.h	Std_VersionInfoType

⌋(*RS_TS_20035*)

8.2 Type definitions

8.2.1 CanTSyn_ConfigType

[SWS_CanTSyn_00091] ⌈

Name	CanTSyn_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	–
Description	This is the base type for the configuration of the Time Synchronization over CAN. A pointer to an instance of this structure will be used in the initialization of the Time Synchronization over CAN. The content of this structure is defined in chapter 10 Configuration specification.	
Available via	CanTSyn.h	

⌋(*RS_TS_20035*)

8.2.2 CanTSyn_TransmissionModeType

[SWS_CanTSyn_00092] ⌈

Name	CanTSyn_TransmissionModeType		
Kind	Enumeration		
Range	CANTSYN_TX_OFF	–	Transmission Disabled
	CANTSYN_TX_ON	–	Transmission Enabled
Description	Handles the enabling and disabling of the transmission mode		
Available via	CanTSyn.h		

⌋(*RS_TS_20035*)

8.3 Function definitions

8.3.1 CanTSyn_Init

[SWS_CanTSyn_00093] ⌈

Service Name	CanTSyn_Init	

Syntax	`void CanTSyn_Init (`
	`  const CanTSyn_ConfigType* configPtr`
	`)`	

Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	configPtr	Pointer to selected configuration structure
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function initializes the Time Synchronization over CAN.	
Available via	CanTSyn.h	

⌋(*RS_TS_20035*)

CANTSYN_E_INIT_FAILED is reported as specified by [SWS_BSW_00050] in [3]. See section 7.2.2 for details.

8.3.2 CanTSyn_GetVersionInfo

[SWS_CanTSyn_00094] ⌈

Service Name	CanTSyn_GetVersionInfo	

Syntax	`void CanTSyn_GetVersionInfo (`
	`  Std_VersionInfoType* versioninfo`
	`)`	

Service ID [hex]	0x02	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	versioninfo	Pointer to where to store the version information of this module.
Return value	None	
Description	Returns the version information of this module.	
Available via	CanTSyn.h	

⌋(*RS_TS_20035*)

8.3.3 CanTSyn_SetTransmissionMode

[SWS_CanTSyn_00095] ⌈

Service Name	CanTSyn_SetTransmissionMode	
Syntax	`void CanTSyn_SetTransmissionMode ( uint8 CtrlIdx, CanTSyn_TransmissionModeType Mode )`	
Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the CAN channel
	Mode	CANTSYN_TX_OFF CANTSYN_TX_ON
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This API is used to turn on and off the TX capabilities of the CanTSyn.	
Available via	CanTSyn.h	

⌋*(RS_TS_20035)*

[SWS_CanTSyn_00134] ⌈The function `CanTSyn_SetTransmissionMode` shall inform the `Det`, if development error detection is enabled (i.e., `CanTSynDevErrorDetect` is set to `TRUE`) and if function call has failed because of the following reasons:

- Invalid CtrlIdx (CANTSYN_E_INV_CTRL_IDX)
- Invalid Mode (CANTSYN_E_PARAM)

⌋*(SRS_BSW_00323, SRS_BSW_00337)*

8.4 Callback notifications

This is a list of functions provided for other modules.

8.4.1 CanTSyn_RxIndication

[SWS_CanTSyn_00096] ⌈

Service Name	CanTSyn_RxIndication
Syntax	`void CanTSyn_RxIndication ( PduIdType RxPduId, const PduInfoType* PduInfoPtr )`

▽

△

Service ID [hex]	0x42	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	RxPduId	ID of the received PDU.
	PduInfoPtr	Contains the length (SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU, and the MetaData related to this PDU.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Indication of a received PDU from a lower layer communication interface module.	
Available via	CanTSyn.h	

⌋(*RS_TS_20035*)

Note: The callback function `CanTSyn_RxIndication` called by the CAN Interface and implemented by the `CanTSyn` module. It is called in case of a receive indication event of the CAN Driver.

[SWS_CanTSyn_00097] ⌈The callback function `CanTSyn_RxIndication` shall inform the `Det`, if development error detection is enabled (`CanTSynDevErrorDetect` is set to TRUE) and if function call has failed because of the following reasons:

- Invalid PDU ID (CANTSYN_E_INVALID_PDUID)
- `PduInfoPtr` or `SduDataPtr` equals `NULL_PTR` (CANTSYN_E_NULL_POINTER)

⌋(*SRS_BSW_00323*, *SRS_BSW_00337*)

Caveats of `CanTSyn_RxIndication`:

- Until this service returns, the CAN Interface will not access `canSduPtr`. The `canSduPtr` is only valid and can be used by upper layers until the indication returns. The CAN Interface guarantees that the number of configured bytes for this `CanTSynRxPduId` is valid. The call context is either on interrupt level (interrupt mode) or on task level (polling mode). This callback service is re-entrant for multiple CAN controller usage.

  Note: Using polling mode as call context significantly increases the latency and thus reduces the precision. It is therefore highly recommended to only use interrupt mode.

- The `CanTSyn` module is initialized correctly.

8.4.2 CanTSyn_TxConfirmation

[SWS_CanTSyn_00099] ⌈

Service Name	CanTSyn_TxConfirmation	
Syntax	`void CanTSyn_TxConfirmation ( PduIdType TxPduId, Std_ReturnType result )`	
Service ID [hex]	0x40	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	ID of the PDU that has been transmitted.
	result	E_OK: The PDU was transmitted. E_NOT_OK: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.	
Available via	CanTSyn.h	

⌋(*RS_TS_20035*)

Note: The callback function `CanTSyn_TxConfirmation` is called by the CAN Interface and implemented by the `CanTSyn` module.

[SWS_CanTSyn_00100] ⌈The callback function `CanTSyn_TxConfirmation` shall inform the `Det`, if development error detection is enabled (`CanTSynDevErrorDetect` is set to TRUE) and if the function call has failed because of the following reason:

- Invalid PDU ID (CANTSYN_E_INVALID_PDUID), i.e., a PDU ID not configured by parameter `CanTSynGlobalTimeMasterConfirmationHandleId`

⌋(*SRS_BSW_00323, SRS_BSW_00337*)

Caveats of `CanTSyn_TxConfirmation`:

- The call context is either on interrupt level (interrupt mode) or on task level (polling mode). This callback service is re-entrant for multiple CAN controller usage.

  Note: Using polling mode as call context significantly increases the latency and thus reduces the precision. It is therefore highly recommended to only use interrupt mode.

- The `CanTSyn` module is initialized correctly.

8.5 Scheduled functions

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

8.5.1 CanTSyn_MainFunction

[SWS_CanTSyn_00102] ⌈

Service Name	CanTSyn_MainFunction
Syntax	`void CanTSyn_MainFunction ( void )`
Service ID [hex]	0x06
Description	Main function for cyclic call / resp. Timesync message transmission
Available via	CanTSyn_SchM.h

⌋(*RS_TS_20035*)

[SWS_CanTSyn_00103] ⌈The frequency of invocations of `CanTSyn_MainFunction` is determined by the configuration parameter `CanTSynMainFunctionPeriod`.⌋(*RS_TS_20035*)

8.6 Expected interfaces

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

8.6.1 Mandatory interfaces

Note: This section defines all interfaces, which are required to fulfill the core functionality of the module.

[SWS_CanTSyn_00105] ⌈

API Function	***Header File***	***Description***
StbM_GetCurrentVirtualLocalTime	StbM.h	Returns the Virtual Local Time of the referenced Time Base.

⌋(*RS_TS_20035*)

8.6.2 Optional interfaces

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

[SWS_CanTSyn_00106] ⌈

API Function	*Header File*	*Description*
CanIf_EnableEgressTimeStamp (draft)	CanIf.h	This service calls the corresponding CAN Driver service to activate egress time stamping on a dedicated message object. **Tags:** atp.Status=draft
CanIf_GetCurrentTime (draft)	CanIf.h	This service calls the corresponding CAN Driver service to retrieve the current time value out of the HW registers. **Tags:** atp.Status=draft
CanIf_GetEgressTimeStamp (draft)	CanIf.h	This service calls the corresponding CAN Driver service to read back the egress time stamp on a dedicated message object. It needs to be called within the TxConfirmation() function. **Tags:** atp.Status=draft
CanIf_GetIngressTimeStamp (draft)	CanIf.h	This service calls the corresponding CAN Driver service to reads back the ingress time stamp on a dedicated message object. It needs to be called within the RxIndication() function. **Tags:** atp.Status=draft
CanIf_Transmit	CanIf.h	Requests transmission of a PDU.
Crc_CalculateCRC8H2F	Crc.h	This service makes a CRC8 calculation with the Polynomial 0x2F on Crc_Length
Det_ReportError	Det.h	Service to report development errors.
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.
StbM_BusGetCurrentTime	StbM.h	Returns the current Time Tuple, status and User Data of the Time Base.
StbM_BusSetGlobalTime	StbM.h	Allows the Time Base Provider Modules to forward a new Global Time tuple (i.e., Rx Time Tuple) to the StbM.
StbM_CanSetMasterTimingData (draft)	StbM_CanTSyn.h	Provides CAN Timesyn module specific data for a Time Master to the StbM. **Tags:** atp.Status=draft
StbM_CanSetSlaveTimingData (draft)	StbM_CanTSyn.h	Allows the CanTSyn Module to forward CAN specific details to the StbM. **Tags:** atp.Status=draft
StbM_GetCurrentTime	StbM.h	Returns a time value (Local Time Base derived from Global Time Base) in standard format. Note: This API shall be called with locked interrupts / within an Exclusive Area to prevent interruption (i.e., the risk that the time stamp is outdated on return of the function call).
StbM_GetOffset	StbM.h	Allows the Timesync Modules to get the current Offset Time and User Data.
StbM_GetTimeBaseStatus	StbM.h	Returns detailed status information for a Synchronized (or Pure Local) Time Base and, if called for an Offset Time Base, for the Offset Time Base and the underlying Synchronized Time Base.
StbM_GetTimeBaseUpdateCounter	StbM.h	Allows the Timesync Modules to detect, whether a Time Base should be transmitted immediately in the subsequent <Bus>TSyn_MainFunction() cycle.

⌋(*RS_TS_20035*)

9 Sequence diagrams

9.1 CAN Time Synchronization (Time Master)

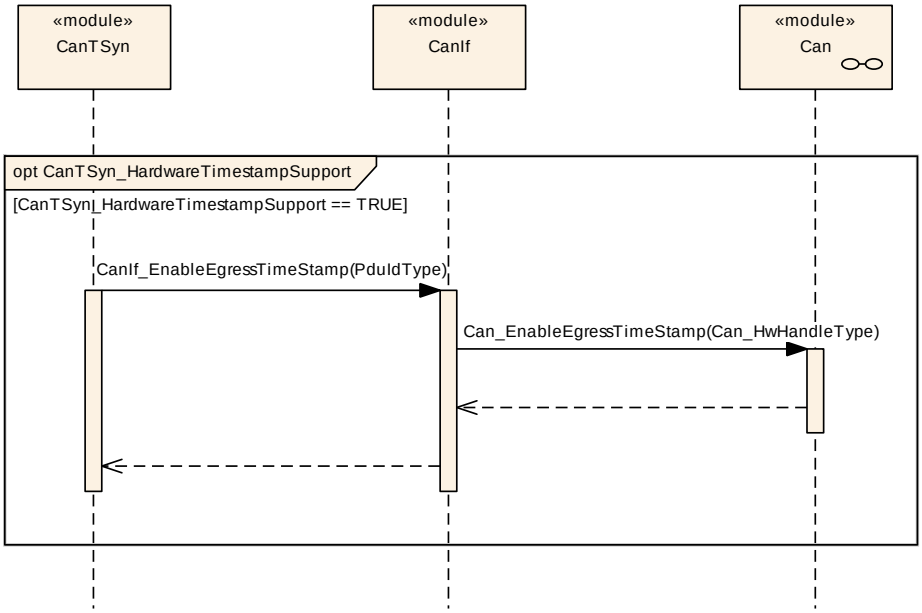

Figure 9.1: CAN Time Synchronization (Time Master)

9.2 CAN Time Synchronization (Time Master)

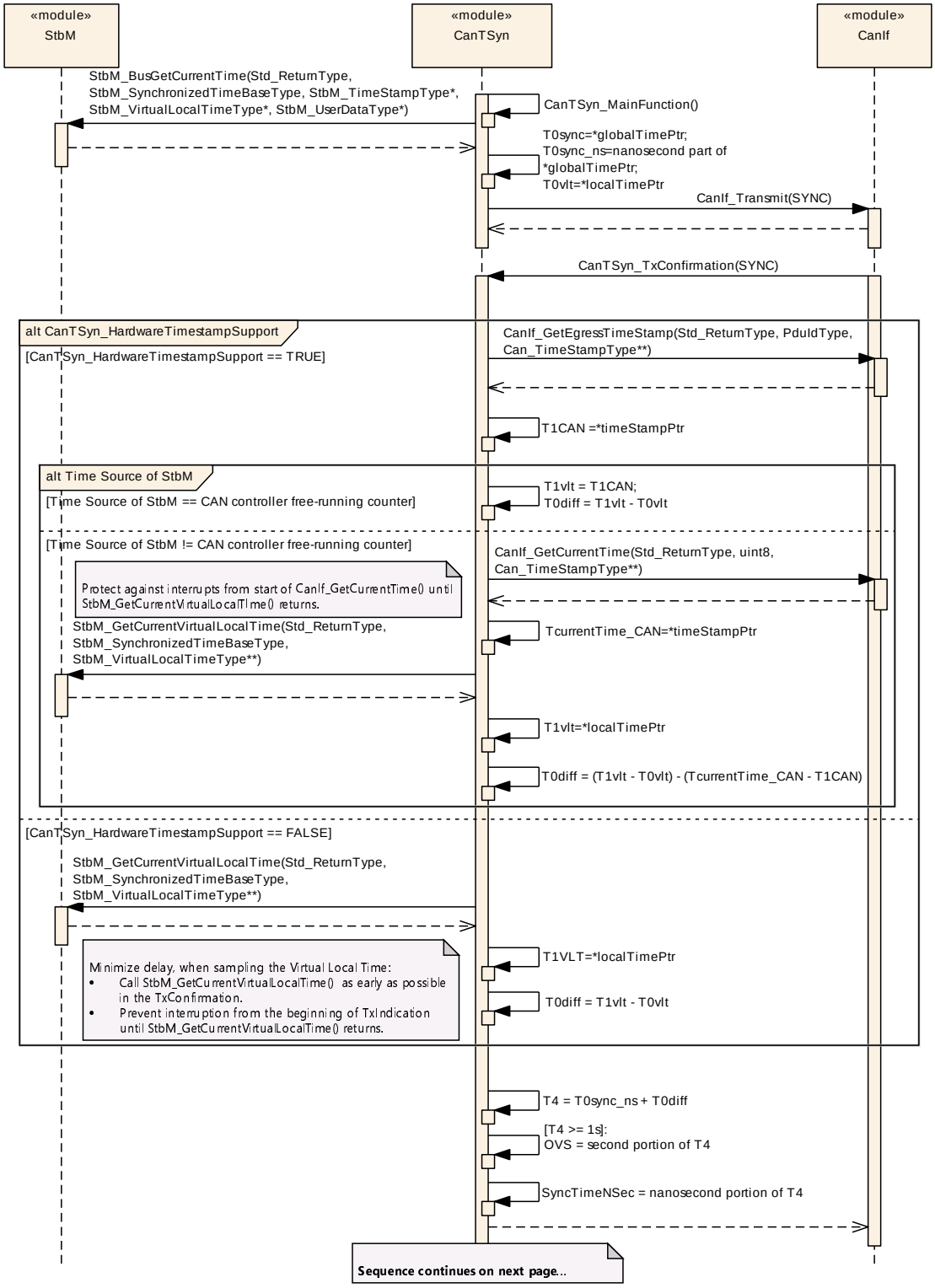

Figure 9.2: CAN Time Synchronization (Time Master), Part 1

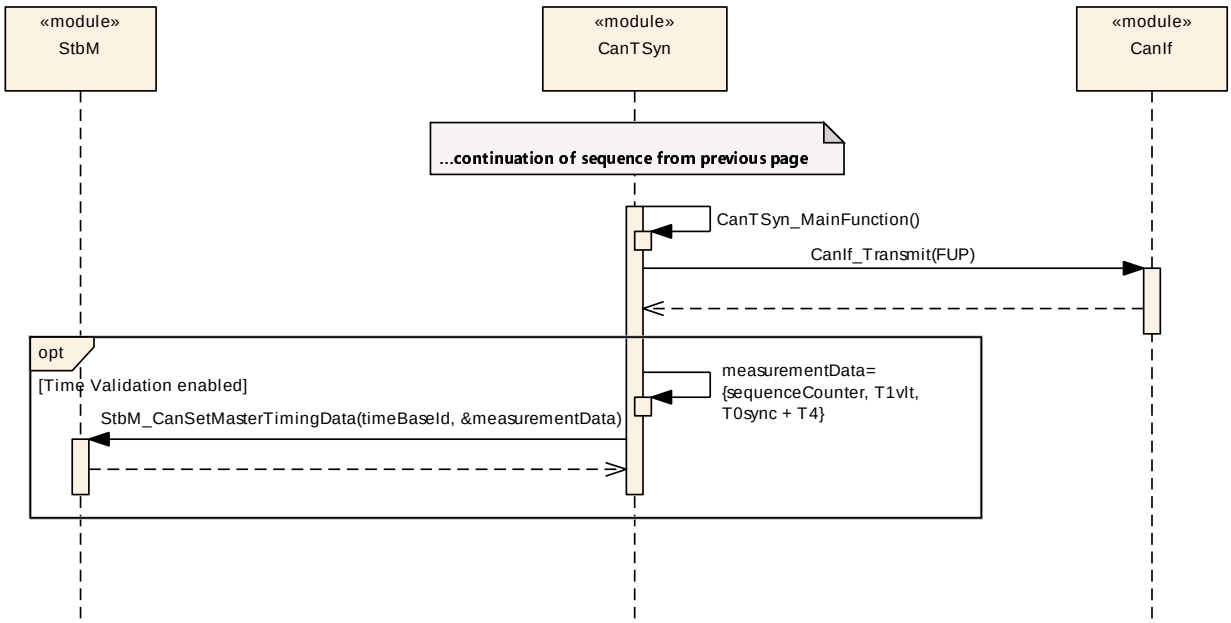

Figure 9.3: CAN Time Synchronization (Time Master), Part 1

9.3 CAN Time Synchronization (Time Slave)

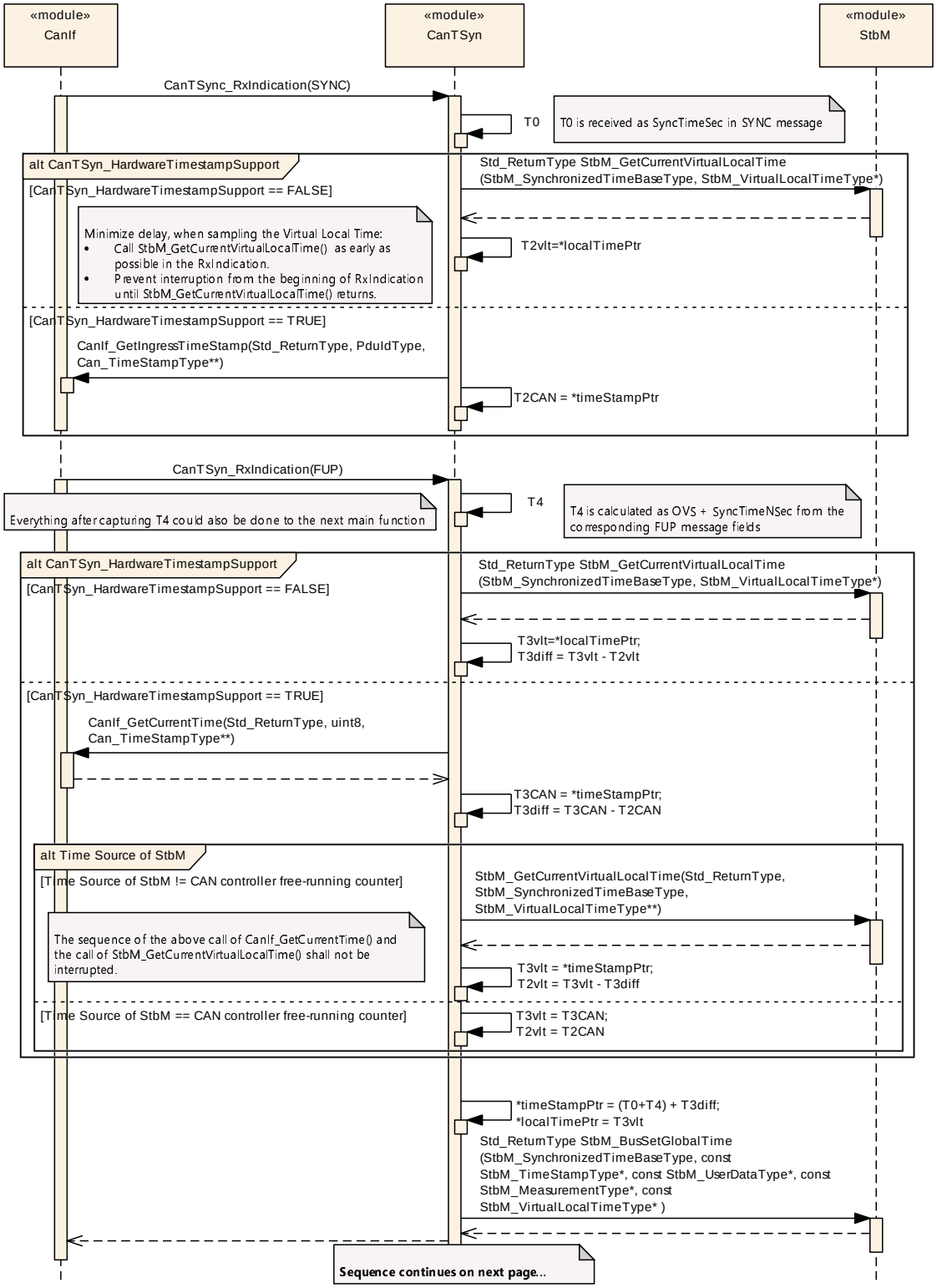

Figure 9.4: CAN Time Synchronization (Time Slave), Part 1

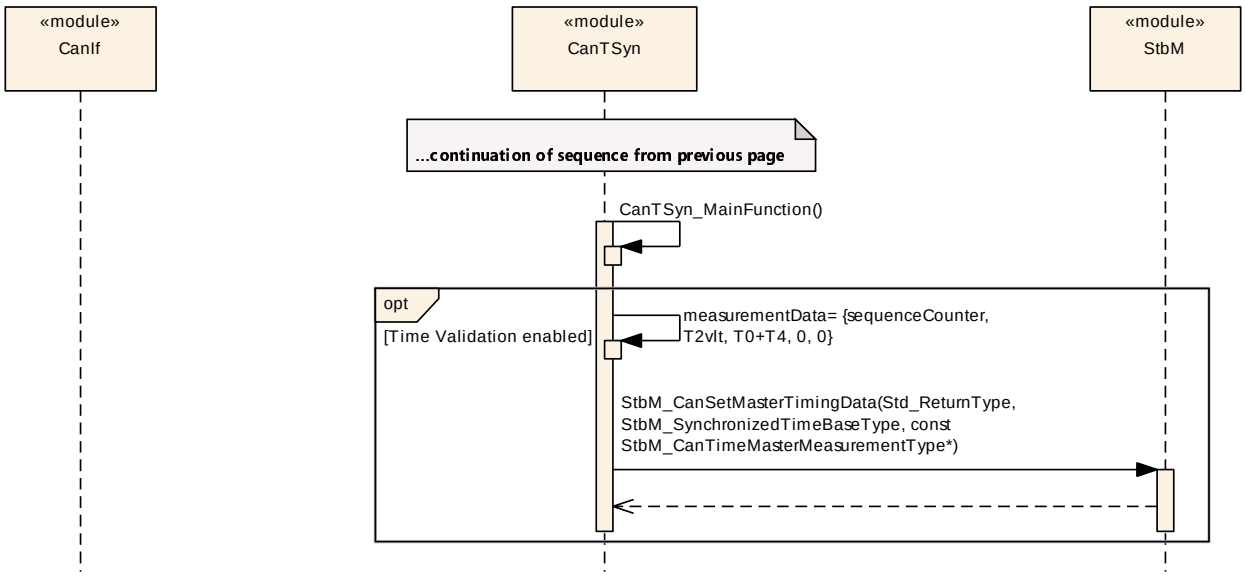

Figure 9.5: CAN Time Synchronization (Time Slave), Part 2

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 `CanTSyn`.

Chapter 10.3 specifies published information of the module `CanTSyn`.

10.1 How to read this chapter

For details refer to the chapter 10.1 "Introduction to configuration specification" in [3].

10.2 Containers and configuration parameters

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

10.2.1 Variants

[SWS_CanTSyn_00108] ⌈The Time Synchronization over CAN shall support the configuration for Time Master, Time Slave and Time Gateway.⌋ *(RS_TS_20038)*

The module supports different post-build variants (previously known as post-build selectable configuration sets), but not post-build loadable configuration.

10.2.2 CanTSyn

Module SWS Item	ECUC_CanTSyn_00001	
Module Name	CanTSyn	
Module Description	Configuration of the Synchronized Time-base Manager (StbM) module with respect to global time handling on CAN.	
Post-Build Variant Support	true	
Supported Config Variants	VARIANT-PRE-COMPILE	
Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
CanTSynGeneral	1	This container holds the general parameters of the CAN-specific Synchronized Time-base Manager
CanTSynGlobalTime Domain	1..*	This represents the existence of a global time domain on CAN. The CanTSyn module can administrate several global time domains at the same time that in itself form a hierarchy of domains and sub-domains. If the CanTSyn exists it is assumed that at least one global time domain exists.

10.2.3 CanTSynGeneral

SWS Item	[ECUC_CanTSyn_00003]
Container Name	CanTSynGeneral
Parent Container	CanTSyn
Description	This container holds the general parameters of the CAN-specific Synchronized Time-base Manager
Configuration Parameters	

Name	CanTSynDevErrorDetect [ECUC_CanTSyn_00002]
Parent Container	CanTSynGeneral
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default Value	false

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

Name	CanTSynHardwareTimestampSupport [ECUC_CanTSyn_00054]		
Parent Container	CanTSynGeneral		
Description	Activate/Deactivate the hardware time stamping functionality of the CAN hardware. True: Timestamp is retrieved from the CAN hardware False: Timestamp is retrieved from the StbM **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
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	CanTSynMainFunctionPeriod [ECUC_CanTSyn_00019]		
Parent Container	CanTSynGeneral		
Description	Schedule period of the main function CanTSyn_MainFunction. Unit: [s].		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
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	CanTSynTimeValidationSupport [ECUC_CanTSyn_00050]		
Parent Container	CanTSynGeneral		
Description	Switches support for Time Validation on or off. • true: Time Validation is enabled. • false: Time Validation is disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynVersionInfoApi [ECUC_CanTSyn_00023]		
Parent Container	CanTSynGeneral		
Description	Activate/Deactivate the version information API (CanTSyn_GetVersionInfo). True: version information API activated False: version information API deactivated.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

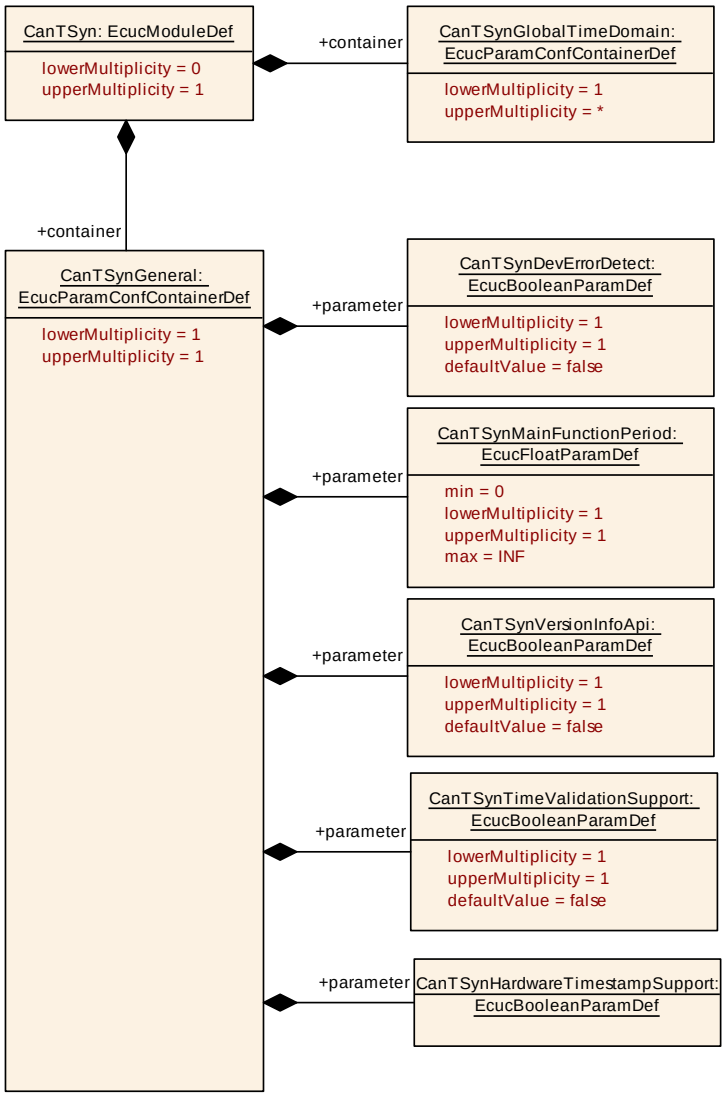

Figure 10.1: CanTSynGeneral

10.2.4 CanTSynGlobalTimeDomain

SWS Item	[ECUC_CanTSyn_00004]
Container Name	CanTSynGlobalTimeDomain
Parent Container	CanTSyn
Description	This represents the existence of a global time domain on CAN. The CanTSyn module can administrate several global time domains at the same time that in itself form a hierarchy of domains and sub-domains. If the CanTSyn exists it is assumed that at least one global time domain exists.
Configuration Parameters	

Name	CanTSynEnableTimeValidation [ECUC_CanTSyn_00051]		
Parent Container	CanTSynGlobalTimeDomain		
Description	Enables/disables time recording for Time Validation for a specific Time Domain.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
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 dependency: Only valid if CanTSynTimeValidationSupport is TRUE. Value set according to parameter StbMEnableTimeValidation of the referenced Time Base in the StbM.		

Name	CanTSynGlobalTimeDomainId [ECUC_CanTSyn_00005]		
Parent Container	CanTSynGlobalTimeDomain		
Description	The global time domain ID.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 31		
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	CanTSynGlobalTimeNetworkSegmentId [ECUC_CanTSyn_00052]		
Parent Container	CanTSynGlobalTimeDomain		
Description	This represents the numerical identifier of the network on system level scope where this Global Time has been communicated on.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynGlobalTimeSecureTmacLength [ECUC_CanTSyn_00046]		
Parent Container	CanTSynGlobalTimeDomain		
Description	Represents the number of bytes for the used Truncated Message Authentication Code (TMAC). If 0, no message authentication will be used. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 16		
Default Value	0		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynUseExtendedMsgFormat [ECUC_CanTSyn_00042]		
Parent Container	CanTSynGlobalTimeDomain		
Description	Switches support for 16 Byte Timesync messages on or off (for CAN FD only) • true: CAN FD support is active: use at least 16 byte for Timesync messages (depending on configuration) • false: Classic CAN support is active: use always 8 byte for Timesync messages		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynSynchronizedTimeBaseRef [ECUC_CanTSyn_00022]		
Parent Container	CanTSynGlobalTimeDomain		
Description	Mandatory reference to the required synchronized time-base.		
Multiplicity	1		
Type	Symbolic name reference to StbMSynchronizedTimeBase		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
CanTSynGlobalTimeFup DataIDList	0..1	The DataIDList for FUP messages ensures the identification of data elements due to CRC calculation and message authentication process.
CanTSynGlobalTime Master	0..1	Configuration of the global time master. Each global time domain is required to have exactly one global time master. This master may or may not exist on the configured ECU.
CanTSynGlobalTime OfnsDataIDList	0..1	The DataIDList for OFNS messages ensures the identification of data elements due to CRC calculation and message authentication process.
CanTSynGlobalTimeOfs DataIDList	0..1	The DataIDList for OFS messages ensures the identification of data elements due to CRC calculation and message authentication process.
CanTSynGlobalTime Slave	0..1	Configuration of a global time slave. Each global time domain is required to have at least one time slave. The configured ECU may or may not represent a time slave.
CanTSynGlobalTime SyncDataIDList	0..1	The DataIDList for SYNC messages ensures the identification of data elements due to CRC calculation and message authentication process.

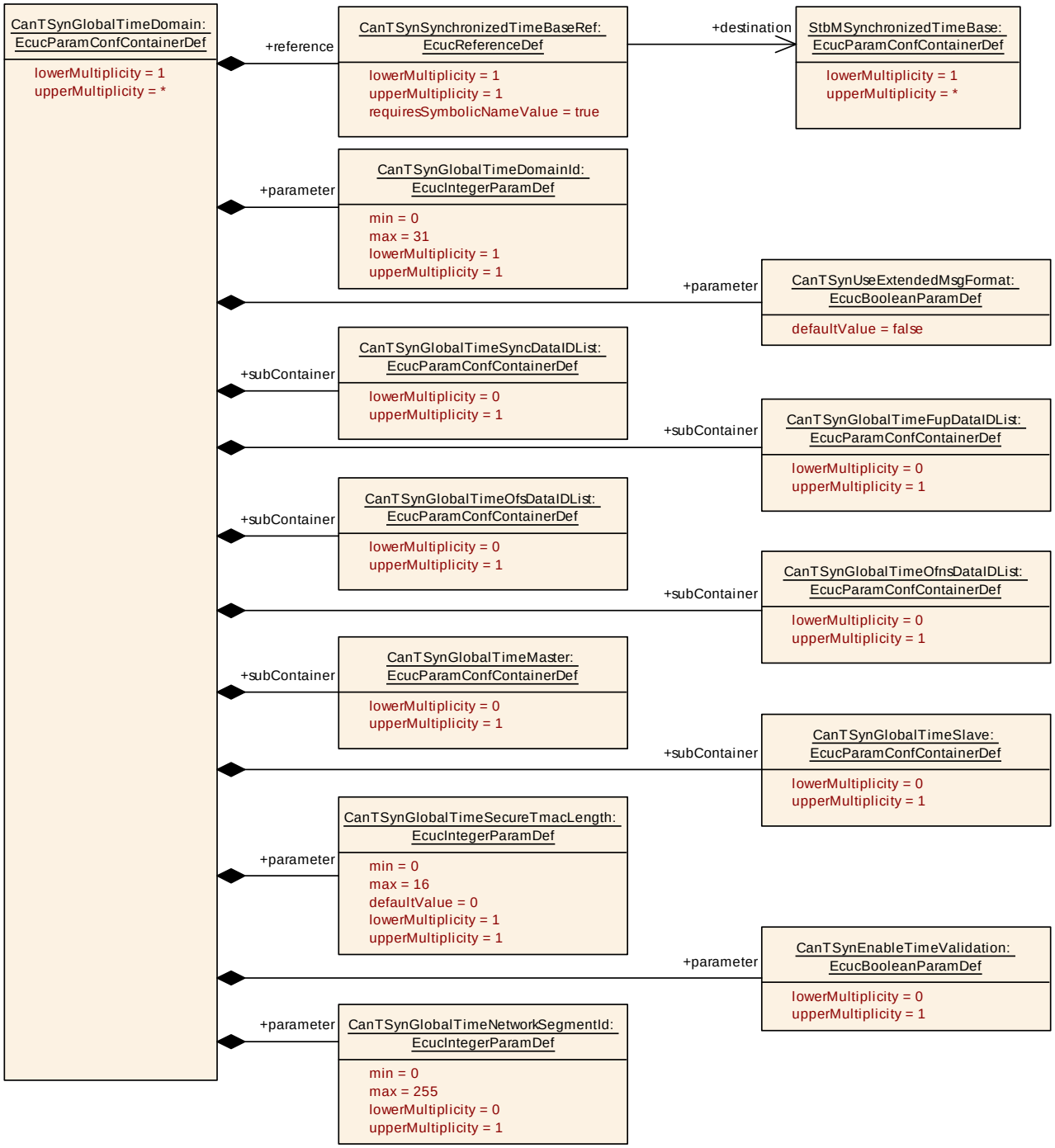

Figure 10.2: CanTSynGlobalTimeDomain

10.2.5 CanTSynGlobalTimeSyncDataIDList

SWS Item	[ECUC_CanTSyn_00024]
Container Name	CanTSynGlobalTimeSyncDataIDList
Parent Container	CanTSynGlobalTimeDomain

Description	The DataIDList for SYNC messages ensures the identification of data elements due to CRC calculation and message authentication process.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
CanTSynGlobalTime SyncDataIDListElement	16	Element of the DataIDList for SYNC messages ensures the identification of data elements due to CRC calculation and message authentication process.

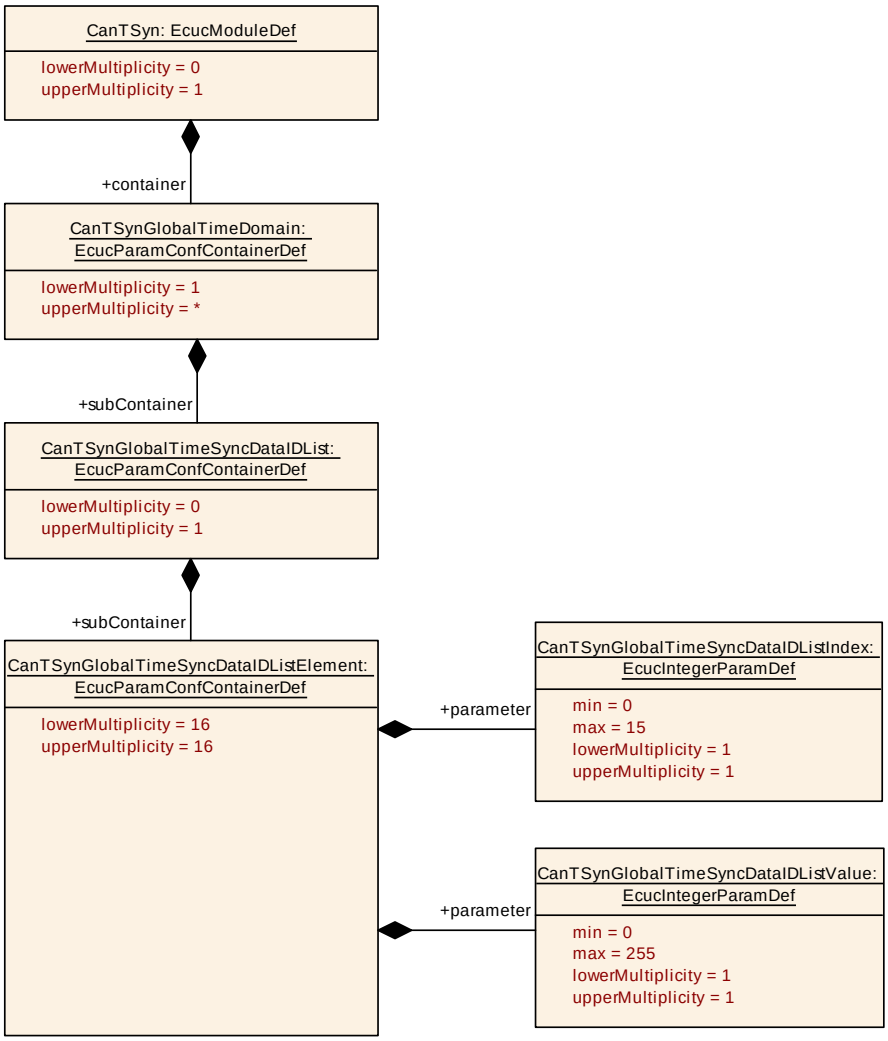

Figure 10.3: CanTSynGlobalTimeSyncDataIDList

10.2.6 CanTSynGlobalTimeSyncDataIDListElement

SWS Item	[ECUC_CanTSyn_00028]
Container Name	CanTSynGlobalTimeSyncDataIDListElement
Parent Container	CanTSynGlobalTimeSyncDataIDList
Description	Element of the DataIDList for SYNC messages ensures the identification of data elements due to CRC calculation and message authentication process.
Configuration Parameters	

Name	CanTSynGlobalTimeSyncDataIDListIndex [ECUC_CanTSyn_00029]		
Parent Container	CanTSynGlobalTimeSyncDataIDListElement		
Description	Index for the DataIDList for SYNC messages ensures the identification of data elements due to CRC calculation and message authentication process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 15		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynGlobalTimeSyncDataIDListValue [ECUC_CanTSyn_00030]		
Parent Container	CanTSynGlobalTimeSyncDataIDListElement		
Description	Value of the DataIDList for SYNC messages ensures the identification of data elements due to CRC calculation and message authentication process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.2.7 CanTSynGlobalTimeFupDataIDList

SWS Item	[ECUC_CanTSyn_00025]		
Container Name	CanTSynGlobalTimeFupDataIDList		
Parent Container	CanTSynGlobalTimeDomain		
Description	The DataIDList for FUP messages ensures the identification of data elements due to CRC calculation and message authentication process.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Containers		
Container Name	Multiplicity	Scope / Dependency
CanTSynGlobalTimeFup DataIDListElement	16	Element of the DataIDList for FUP messages ensures the identification of data elements due to CRC calculation and message authentication process.

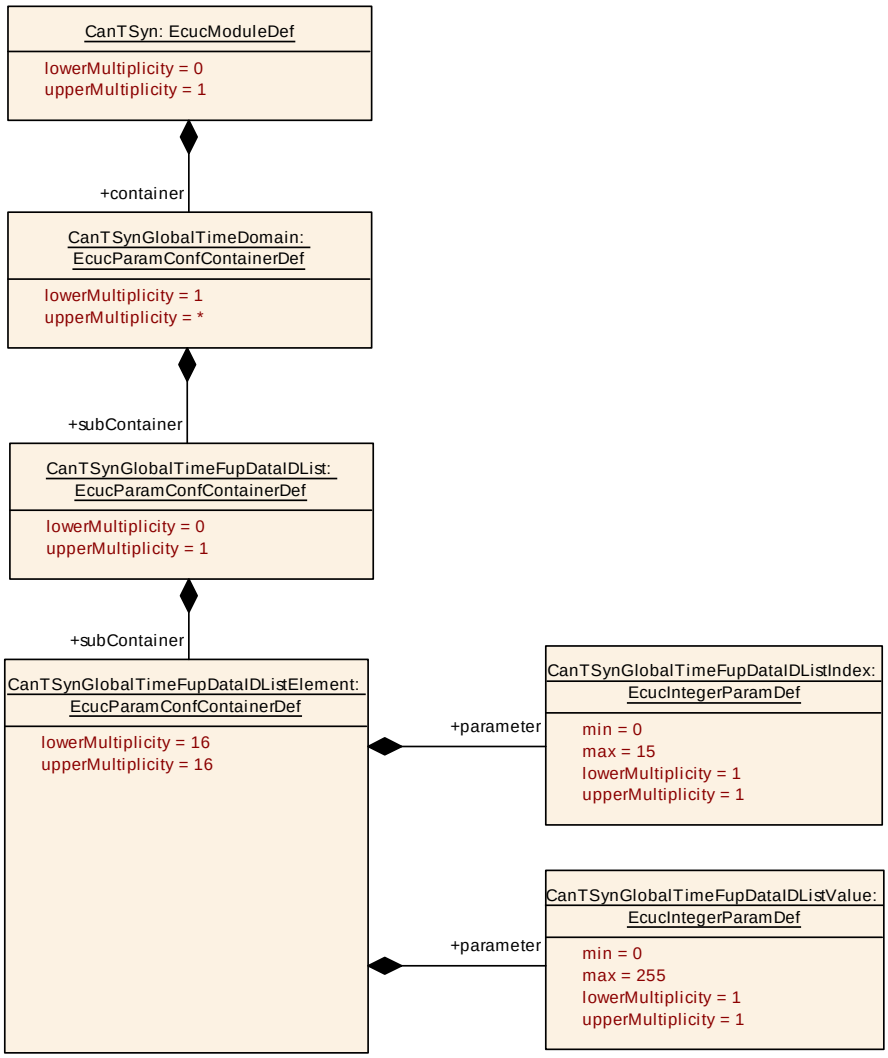

Figure 10.4: CanTSynGlobalTimeFupDataIDList

10.2.8 CanTSynGlobalTimeFupDataIDListElement

SWS Item	[ECUC_CanTSyn_00031]
Container Name	CanTSynGlobalTimeFupDataIDListElement
Parent Container	CanTSynGlobalTimeFupDataIDList
Description	Element of the DataIDList for FUP messages ensures the identification of data elements due to CRC calculation and message authentication process.
Configuration Parameters	

Name	CanTSynGlobalTimeFupDataIDListIndex [ECUC_CanTSyn_00032]		
Parent Container	CanTSynGlobalTimeFupDataIDListElement		
Description	Index of the DataIDList for FUP messages ensures the identification of data elements due to CRC calculation and message authentication process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 15		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynGlobalTimeFupDataIDListValue [ECUC_CanTSyn_00033]		
Parent Container	CanTSynGlobalTimeFupDataIDListElement		
Description	Value of the DataIDList for FUP messages ensures the identification of data elements due to CRC calculation and message authentication process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.2.9 CanTSynGlobalTimeOfsDataIDList

SWS Item	[ECUC_CanTSyn_00026]		
Container Name	CanTSynGlobalTimeOfsDataIDList		
Parent Container	CanTSynGlobalTimeDomain		
Description	The DataIDList for OFS messages ensures the identification of data elements due to CRC calculation and message authentication process.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Configuration Parameters

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
CanTSynGlobalTimeOfs DataIDListElement	16	Element of the DataIDList for OFS messages ensures the identification of data elements due to CRC calculation and message authentication process.

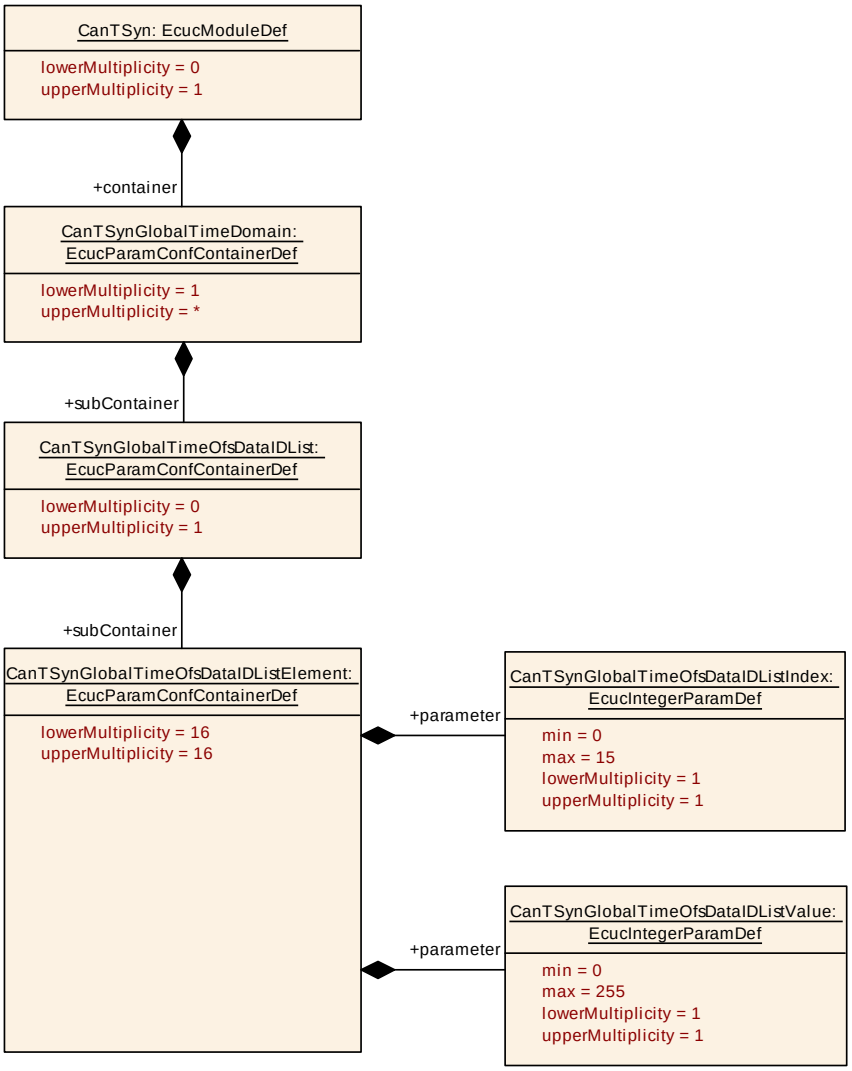

Figure 10.5: CanTSynGlobalTimeOfsDataIDList

10.2.10 CanTSynGlobalTimeOfsDataIDListElement

SWS Item	[ECUC_CanTSyn_00034]
Container Name	CanTSynGlobalTimeOfsDataIDListElement
Parent Container	CanTSynGlobalTimeOfsDataIDList

Description	Element of the DataIDList for OFS messages ensures the identification of data elements due to CRC calculation and message authentication process.
Configuration Parameters	

Name	CanTSynGlobalTimeOfsDataIDListIndex [ECUC_CanTSyn_00035]		
Parent Container	CanTSynGlobalTimeOfsDataIDListElement		
Description	Index of the DataIDList for OFS messages ensures the identification of data elements due to CRC calculation and message authentication process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 15		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynGlobalTimeOfsDataIDListValue [ECUC_CanTSyn_00036]		
Parent Container	CanTSynGlobalTimeOfsDataIDListElement		
Description	Value of the DataIDList for OFS messages ensures the identification of data elements due to CRC calculation and message authentication process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.2.11 CanTSynGlobalTimeOfnsDataIDList

SWS Item	[ECUC_CanTSyn_00041]
Container Name	CanTSynGlobalTimeOfnsDataIDList
Parent Container	CanTSynGlobalTimeDomain

Description	The DataIDList for OFNS messages ensures the identification of data elements due to CRC calculation and message authentication process.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
CanTSynGlobalTime OfnsDataIDListElement	16	Element of the DataIDList for OFNS messages ensures the identification of data elements due to CRC calculation and message authentication process.

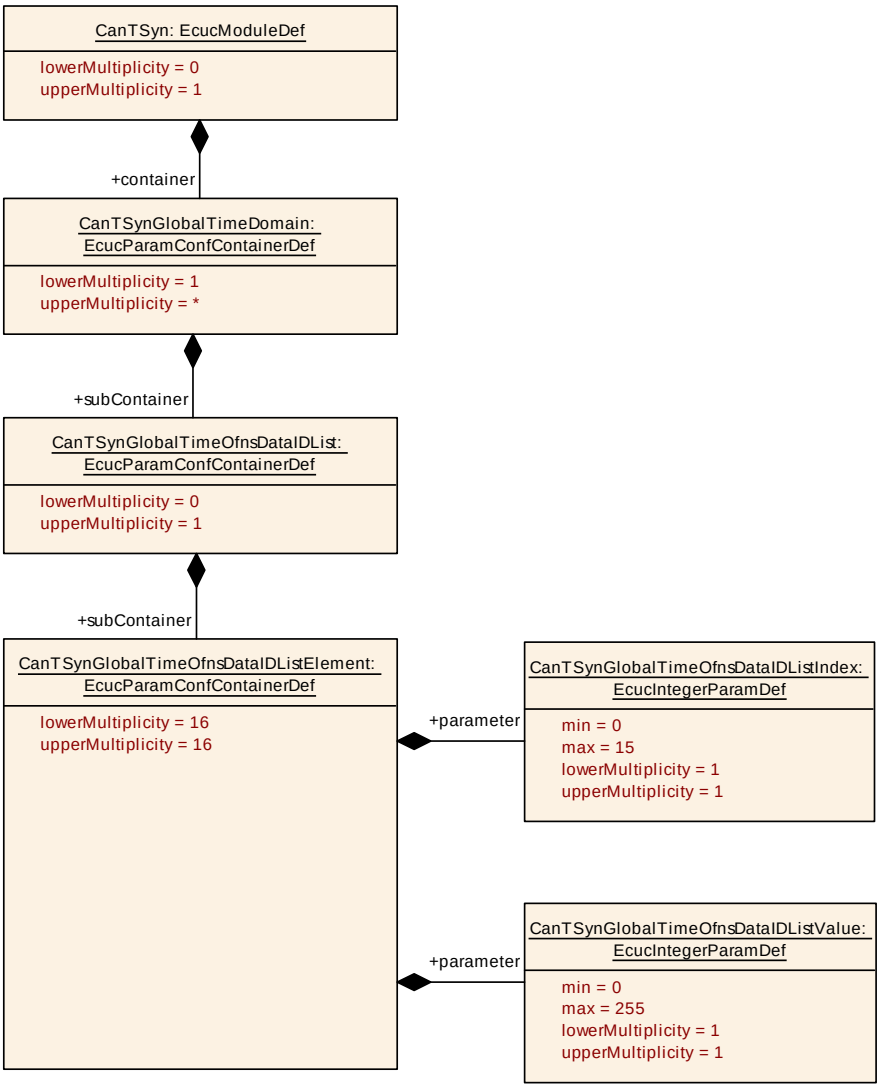

Figure 10.6: CanTSynGlobalTimeOfnsDataIDList

10.2.12 CanTSynGlobalTimeOfnsDataIDListElement

SWS Item	[ECUC_CanTSyn_00037]
Container Name	CanTSynGlobalTimeOfnsDataIDListElement
Parent Container	CanTSynGlobalTimeOfnsDataIDList
Description	Element of the DataIDList for OFNS messages ensures the identification of data elements due to CRC calculation and message authentication process.
Configuration Parameters	

Name	CanTSynGlobalTimeOfnsDataIDListIndex [ECUC_CanTSyn_00038]		
Parent Container	CanTSynGlobalTimeOfnsDataIDListElement		
Description	Index of the DataIDList for OFNS messages ensures the identification of data elements due to CRC calculation and message authentication process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 15		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynGlobalTimeOfnsDataIDListValue [ECUC_CanTSyn_00039]		
Parent Container	CanTSynGlobalTimeOfnsDataIDListElement		
Description	Value of the DataIDList for OFNS messages ensures the identification of data elements due to CRC calculation and message authentication process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.2.13 CanTSynGlobalTimeMaster

SWS Item	[ECUC_CanTSyn_00007]		
Container Name	CanTSynGlobalTimeMaster		
Parent Container	CanTSynGlobalTimeDomain		
Description	Configuration of the global time master. Each global time domain is required to have exactly one global time master. This master may or may not exist on the configured ECU.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	CanTSynCyclicMsgResumeTime [ECUC_CanTSyn_00044]		
Parent Container	CanTSynGlobalTimeMaster		
Description	Defines the time where the 1st regular cycle time based message transmission takes place, after an immediate transmission before. Unit: seconds		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynGlobalTimeDebounceTime [ECUC_CanTSyn_00045]		
Parent Container	CanTSynGlobalTimeMaster		
Description	This represents the configuration of a TX debounce time for SYNC, FUP, OFS and OFNS messages compared to a message before with the same PDU. Unit: seconds		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynGlobalTimeTxCrcSecured [ECUC_CanTSyn_00015]		
Parent Container	CanTSynGlobalTimeMaster		
Description	This represents the configuration of whether or not CRC is supported.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRC_NOT_SUPPORTED	This represents a configuration where CRC is not supported.	
	CRC_SUPPORTED	This represents a configuration where CRC is supported.	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynGlobalTimeTxPeriod [ECUC_CanTSyn_00017]		
Parent Container	CanTSynGlobalTimeMaster		
Description	This represents configuration of the TX period. Unit: seconds		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynImmediateTimeSync [ECUC_CanTSyn_00043]		
Parent Container	CanTSynGlobalTimeMaster		
Description	Enables/Disables the cyclic polling of StbM_GetTimeBaseUpdateCounter() within CanTSyn_MainFunction().		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynTxTmacCalculated [ECUC_CanTSyn_00047]		
Parent Container	CanTSynGlobalTimeMaster		
Description	This parameter controls whether or not TMAC calculation shall be supported. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	TMAC_CALCULATED	The Timesync module shall calculate the TMAC.	
	TMAC_NOT_CALCULATED	The Timesync module shall not calculate any TMAC.	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
CanTSynGlobalTimeMasterPdu	1	This container encloses the configuration of the PDU that is supposed to contain the global time information.

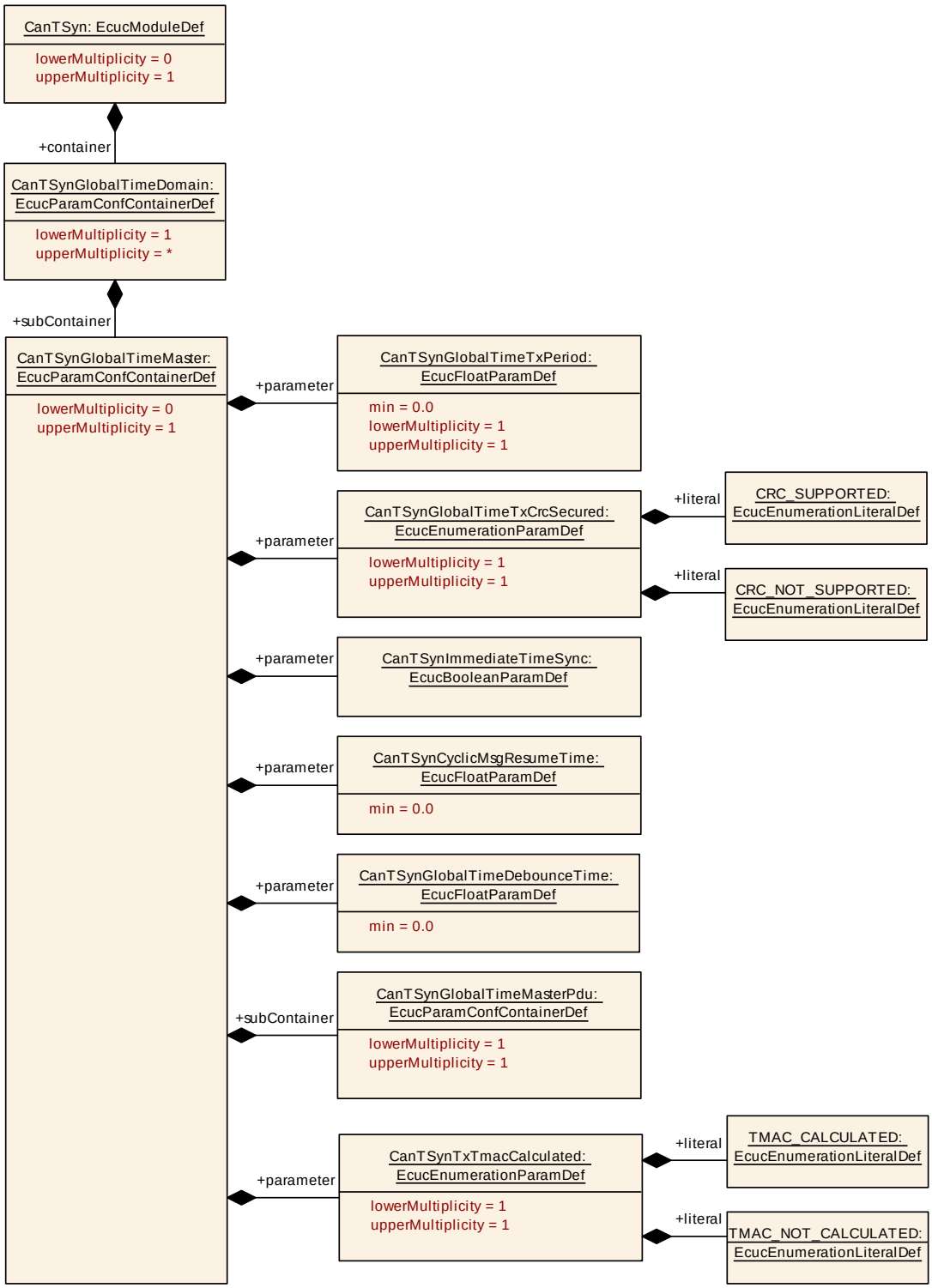

Figure 10.7: CanTSynGlobalTimeMaster

10.2.14 CanTSynGlobalTimeMasterPdu

SWS Item	[ECUC_CanTSyn_00009]

Container Name	CanTSynGlobalTimeMasterPdu
Parent Container	CanTSynGlobalTimeMaster
Description	This container encloses the configuration of the PDU that is supposed to contain the global time information.
Configuration Parameters	

Name	CanTSynGlobalTimeMasterConfirmationHandleId [ECUC_CanTSyn_00008]		
Parent Container	CanTSynGlobalTimeMasterPdu		
Description	This represents the handle ID of the PDU that contains the global time information.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
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	CanTSynGlobalTimePduRef [ECUC_CanTSyn_00027]		
Parent Container	CanTSynGlobalTimeMasterPdu		
Description	This represents the reference to the Pdu taken to transmit the global time information. The global time master of a global time domain acts as the sender of the Pdu while all the time slaves are supposed to receive the Pdu.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

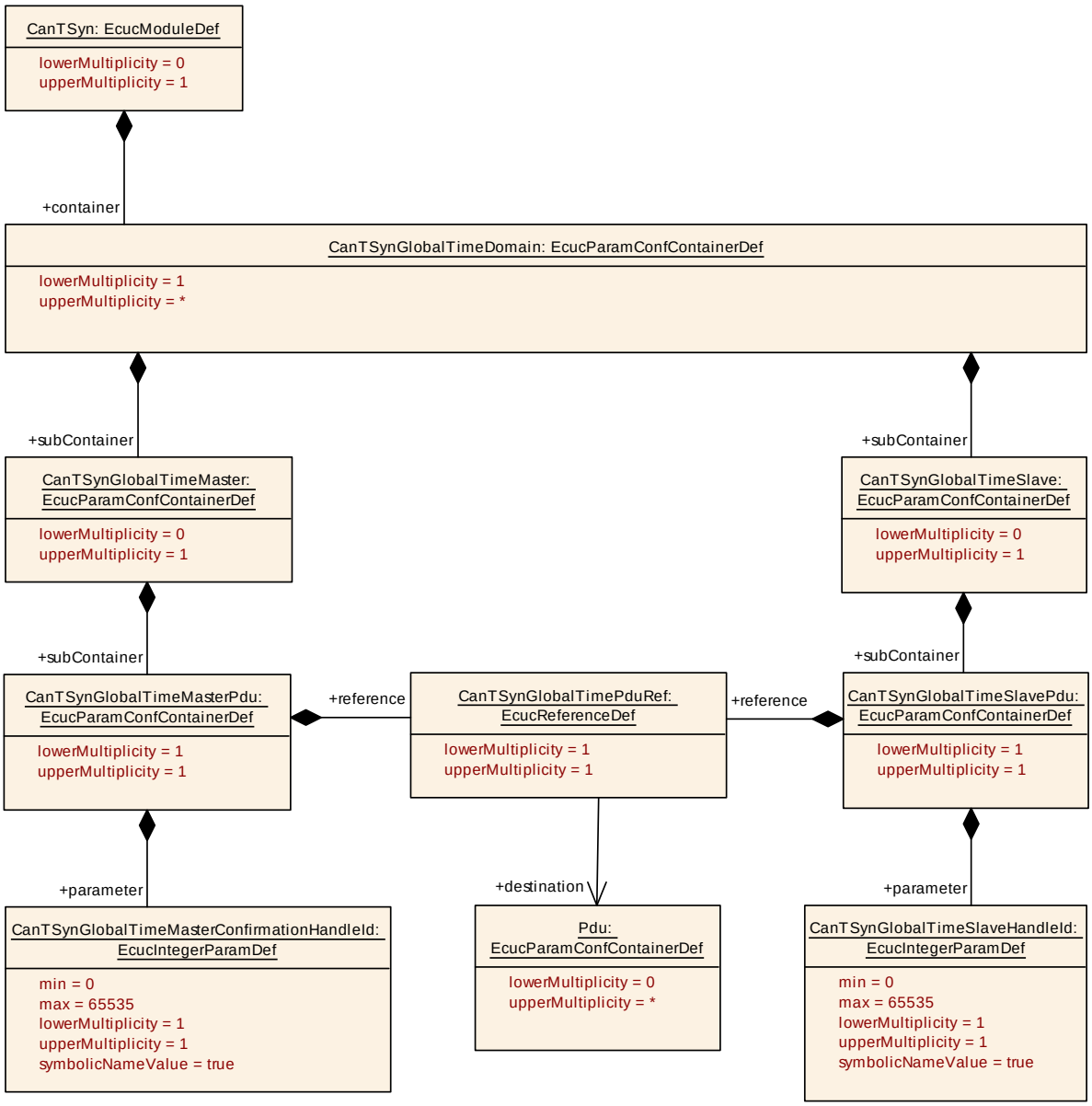

Figure 10.8: CanTSynGlobalTimePdu

10.2.15 CanTSynGlobalTimeSlave

SWS Item	[ECUC_CanTSyn_00012]
Container Name	CanTSynGlobalTimeSlave
Parent Container	CanTSynGlobalTimeDomain
Description	Configuration of a global time slave. Each global time domain is required to have at least one time slave. The configured ECU may or may not represent a time slave.
Post-Build Variant Multiplicity	true

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	CanTSynGlobalTimeFollowUpTimeout [ECUC_CanTSyn_00006]		
Parent Container	CanTSynGlobalTimeSlave		
Description	Rx timeout for the follow-up message. This is only relevant for selected bus systems Unit:seconds		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default Value			
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynGlobalTimeMinMsgGap [ECUC_CanTSyn_00049]		
Parent Container	CanTSynGlobalTimeSlave		
Description	This parameter represents the configuration of a minimum message gap time for received Timesync messages compared to a message before with the same PDU. If PDUs are received more often in between than this parameter allows, they shall be ignored. Unit: seconds **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF[		
Default Value	0		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynGlobalTimeSequenceCounterHysteresis [ECUC_CanTSyn_00053]		
Parent Container	CanTSynGlobalTimeSlave		
Description	CanTSynGlobalTimeSequenceCounterHysteresis specifies the number of consecutive valid message pairs that are required by the Time Slave while being in Timeout state until a Time Tuple is forwarded to the StbM.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 15		
Default Value	0		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynGlobalTimeSequenceCounterJumpWidth [ECUC_CanTSyn_00011]		
Parent Container	CanTSynGlobalTimeSlave		
Description	The SequenceCounterJumpWidth specifies the maximum allowed gap of the Sequence Counter between two SYNC resp. two OFS messages.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 15		
Default Value	0		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynRxCrcValidated [ECUC_CanTSyn_00021]	
Parent Container	CanTSynGlobalTimeSlave	
Description	Definition of whether or not validation of the CRC is supported.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	CRC_IGNORED	The Timesync module accepts Time Synchronization messages, which are CRC secured (without actually validating the CRC) and those, which are not CRC secured. That means, the Timesync module ignores the CRC.

	CRC_NOT_VALIDATED		The Timesync module accepts only Time Synchronization messages, which are not CRC secured. All other Time Synchronization messages are ignored.
	CRC_OPTIONAL		The Timesync module accepts only Time Synchronization messages which are not CRC secured and Time Synchronization messages which are CRC secured and have the correct CRC. All other Time Synchronization messages are ignored.
	CRC_VALIDATED		The Timesync module accepts only Time Synchronization messages, which are CRC secured and have the correct CRC. All other Time Synchronization messages are ignored.
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	CanTSynRxTmacValidated [ECUC_CanTSyn_00048]		
Parent Container	CanTSynGlobalTimeSlave		
Description	This parameter controls whether or not TMAC validation shall be supported. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	TMAC_NOT_VALIDATED		The Timesync module shall not validate the TMAC.
	TMAC_VALIDATED		The Timesync module shall validate the TMAC.
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
CanTSynGlobalTimeSlavePdu	1	This container encloses the configuration of the PDU that is supposed to contain the global time information.

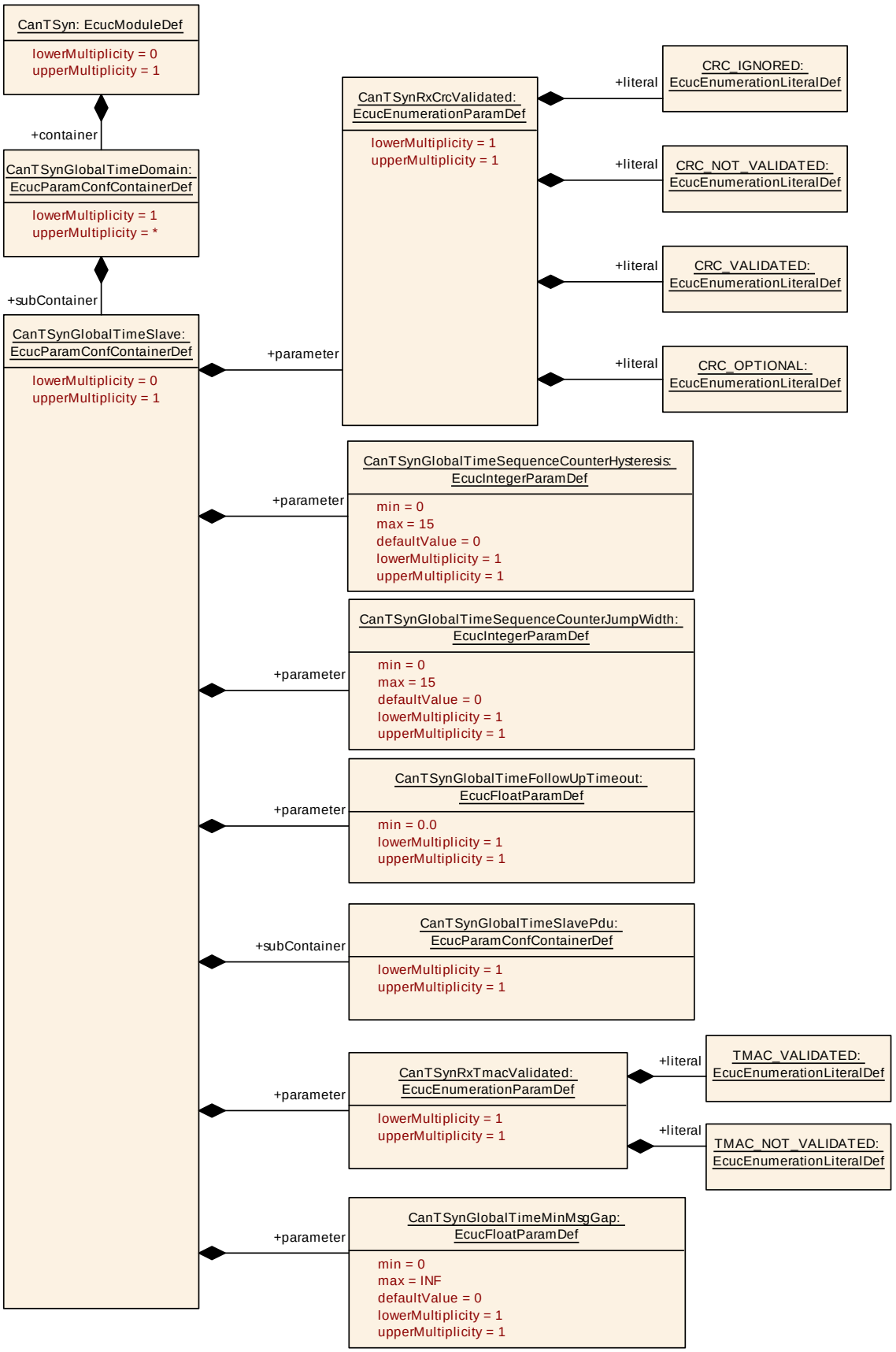

Figure 10.9: CanTSynGlobalTimeSlave

10.2.16 CanTSynGlobalTimeSlavePdu

SWS Item	[ECUC_CanTSyn_00014]
Container Name	CanTSynGlobalTimeSlavePdu
Parent Container	CanTSynGlobalTimeSlave
Description	This container encloses the configuration of the PDU that is supposed to contain the global time information.
Configuration Parameters	

Name	CanTSynGlobalTimeSlaveHandleId [ECUC_CanTSyn_00013]		
Parent Container	CanTSynGlobalTimeSlavePdu		
Description	This represents the handle ID of the PDU that contains the global time information.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
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	CanTSynGlobalTimePduRef [ECUC_CanTSyn_00040]		
Parent Container	CanTSynGlobalTimeSlavePdu		
Description	This represents the reference to the Pdu taken to transmit the global time information. The global time master of a global time domain acts as the sender of the Pdu while all the time slaves are supposed to receive the Pdu.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.3 Published Information

For details, refer to the chapter 10.3 "Published Information" in [3].

A Not applicable requirements

[SWS_CanTSyn_00999] ⌈These requirements on Time Synchronization from the RS Time Synchronization [1] are not applicable to `CanTSyn`, because they refer either to network types other than CAN or to the Time Base Manager module.⌋*(RS_TS_00002, RS_TS_00005, RS_TS_00006, RS_TS_00007, RS_TS_00008, RS_TS_00009, RS_TS_00010, RS_TS_00011, RS_TS_00012, RS_TS_00013, RS_TS_00014, RS_TS_00015, RS_TS_00016, RS_TS_00017, RS_TS_00018, RS_TS_00019, RS_TS_00021, RS_TS_00024, RS_TS_00025, RS_TS_00026, RS_TS_00027, RS_TS_00029, RS_TS_00030, RS_TS_00031, RS_TS_00032, RS_TS_00033, RS_TS_00035, RS_TS_00036, RS_TS_00037, RS_TS_00038, RS_TS_20039, RS_TS_20040, RS_TS_20041, RS_TS_20042, RS_TS_20043, RS_TS_20044, RS_TS_20045, RS_TS_20046, RS_TS_20047, RS_TS_20048, RS_TS_20051, RS_TS_20052, RS_TS_20053, RS_TS_20054, RS_TS_20058, RS_TS_20059, RS_TS_20060, RS_TS_20061, RS_TS_20062, RS_TS_20063, RS_TS_20066, RS_TS_20069)*