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

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.4.0

Document Change History

Date	Release	Changed by	Change Description
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

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

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 5

2 Acronyms, Abbreviations and Definitions 6

3 Related documentation 7

3.1 Input documents 7
3.2 Related specification 7

4 Constraints and assumptions 8

4.1 Limitations 8
4.2 Applicability to car domains 8

5 Dependencies to other modules 9

5.1 File structure 10
5.1.1 Code file structure 10
5.1.2 Header file structure 10

6 Requirements traceability 11

7 Functional specification 14

7.1 Overview 14
7.2 Module Handling 14
7.2.1 Interrupt Handling 14
7.2.2 Initialization 15
7.3 Message Format 15
7.3.1 SYNC and FUP Message 16
7.3.2 Offset Messages 17
7.4 Acting as Time Master 21
7.4.1 SYNC and FUP message processing 22
7.4.2 OFS message processing 23
7.4.3 Transmission mode 24
7.4.4 Debounce Time 25
7.4.5 Immediate Time Synchronization 25
7.4.6 Calculation and Assembling of Time Synchronization Messages 27
7.5 Acting as Time Slave 30
7.5.1 SYNC and FUP message processing 30
7.5.2 OFS and OFNS message processing 31
7.5.3 Validation and Disassembling of Time Synchronization Messages 33
7.6 Error Classification 37
7.6.1 Development Errors 37
7.6.2 Runtime Errors 37
7.6.3 Transient Faults 37
7.6.4 Production Errors 37
7.6.5 Extended Production Errors 38

8 API specification 39

8.1 API 39

8.1.1 Imported types 39
8.1.2 Type definitions 39
8.1.3 Function definitions 40
8.1.4 Call-back notifications 41
8.1.5 Scheduled functions 43
8.1.6 Expected Interfaces 45
9 Sequence diagrams 46
9.1 CAN Time Synchronization (Time Master) 46
9.2 CAN Time Synchronization (Time Slave) 47
10 Configuration specification 48
10.1 How to read this chapter 48
10.2 Containers and configuration parameters 49
10.2.1 Variants 49
10.2.2 CanTSyn 49
10.2.3 CanTSynGeneral 50
10.2.4 CanTSynGlobalTimeDomain 51
10.2.5 CanTSynGlobalTimeSyncDataIDList 55
10.2.6 CanTSynGlobalTimeSyncDataIDListElement 56
10.2.7 CanTSynGlobalTimeFupDataIDList 57
10.2.8 CanTSynGlobalTimeFupDataIDListElement 58
10.2.9 CanTSynGlobalTimeOfsDataIDList 59
10.2.10 CanTSynGlobalTimeOfsDataIDListElement 60
10.2.11 CanTSynGlobalTimeOfnsDataIDList 61
10.2.12 CanTSynGlobalTimeOfnsDataIDListElement 63
10.2.13 CanTSynGlobalTimeMaster 63
10.2.14 CanTSynGlobalTimeMasterPdu 68
10.2.15 CanTSynGlobalTimeSlave 69
10.2.16 CanTSynGlobalTimeSlavePdu 73
10.3 Published Information 73

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.

The following Figure shows the CAN Time Synchronization mechanism.

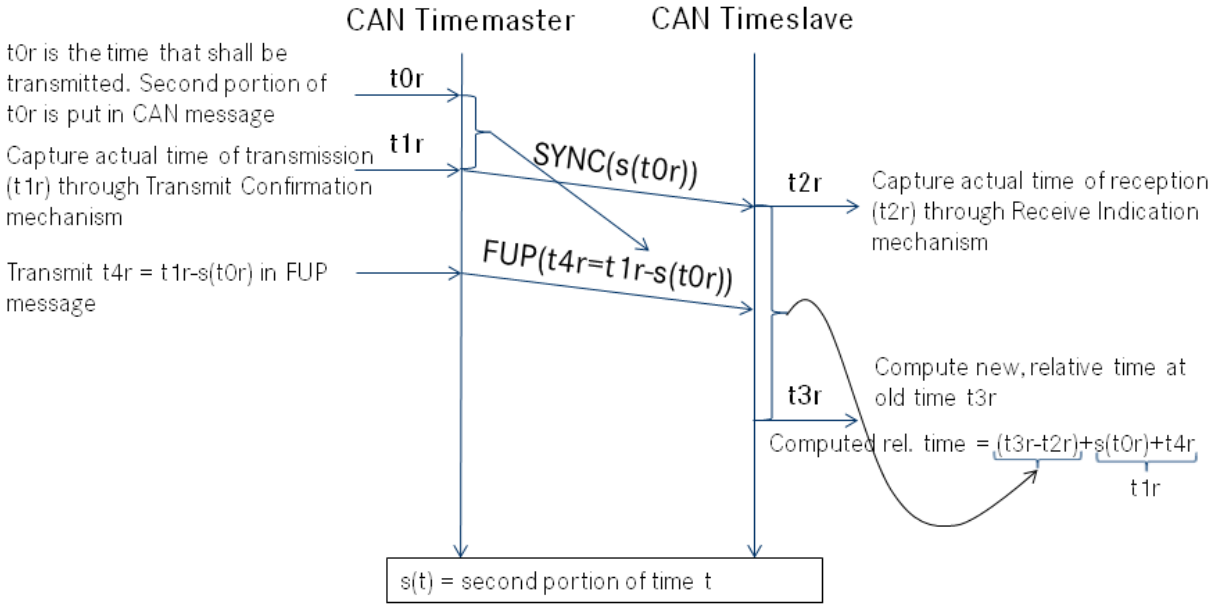


Figure 1: CAN Time Synchronization mechanism

2 Acronyms, Abbreviations and Definitions

This section lists module local Abbreviations and Definitions. For a complete set of Synchronized Time Base related Abbreviations and Definitions refer to the corresponding chapter in [4].

Abbreviation / Acronym:	Description
(G)TD	(Global) Time Domain
(G)TM	(Global)Time Master
<Bus>TSyn	A bus specific Time Synchronization module
CAN	Controller Area Network
CanTSyn	Time Synchronization module for CAN
CRC	Cyclic Redundancy Checksum
Debounce Time	Minimum gap between two Tx messages with the same PDU
DEM	Diagnostic Event Manager
DET	Default Error Tracer
DLC	Data Length Code
FUP message	Follow-Up message
OFNS message	Offset adjustment message
OFS message	Offset Synchronization message
StbM	Synchronized Time-Base Manager
SYNC message	Time Synchronization message
TG	Time Gateway
Timesync	Time Synchronization
TS	Time Slave
TSD	Time Sub-domain

3 Related documentation

3.1 Input documents

[1] Requirements on Synchronized Time-Base Manager
AUTOSAR_SRS_SynchronizedTimeBaseManager.pdf

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[3] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

[4] Specification of Synchronized Time-Base Manager
AUTOSAR_SWS_SynchronizedTimeBaseManager.pdf

[5] Specification of CRC Routines
AUTOSAR_SWS_CRCLibrary.pdf

[6] Specification of CAN Interface
AUTOSAR_SWS_CANInterface.pdf

[7] Specification of Default Error Tracer
AUTOSAR_SWS_DefaultErrorTracer.pdf

[8] Specification of Basic Software Mode Manager
AUTOSAR_SWS_BSWModeManager.pdf

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software (SWS BSW General [3]) 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.

4.2 Applicability to car domains

Systems requiring a common Time Base to ECUs independent to which bus system the ECU is connected.

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 Basic Software Mode Manager (BswM) and the Default Error Tracer (DET).

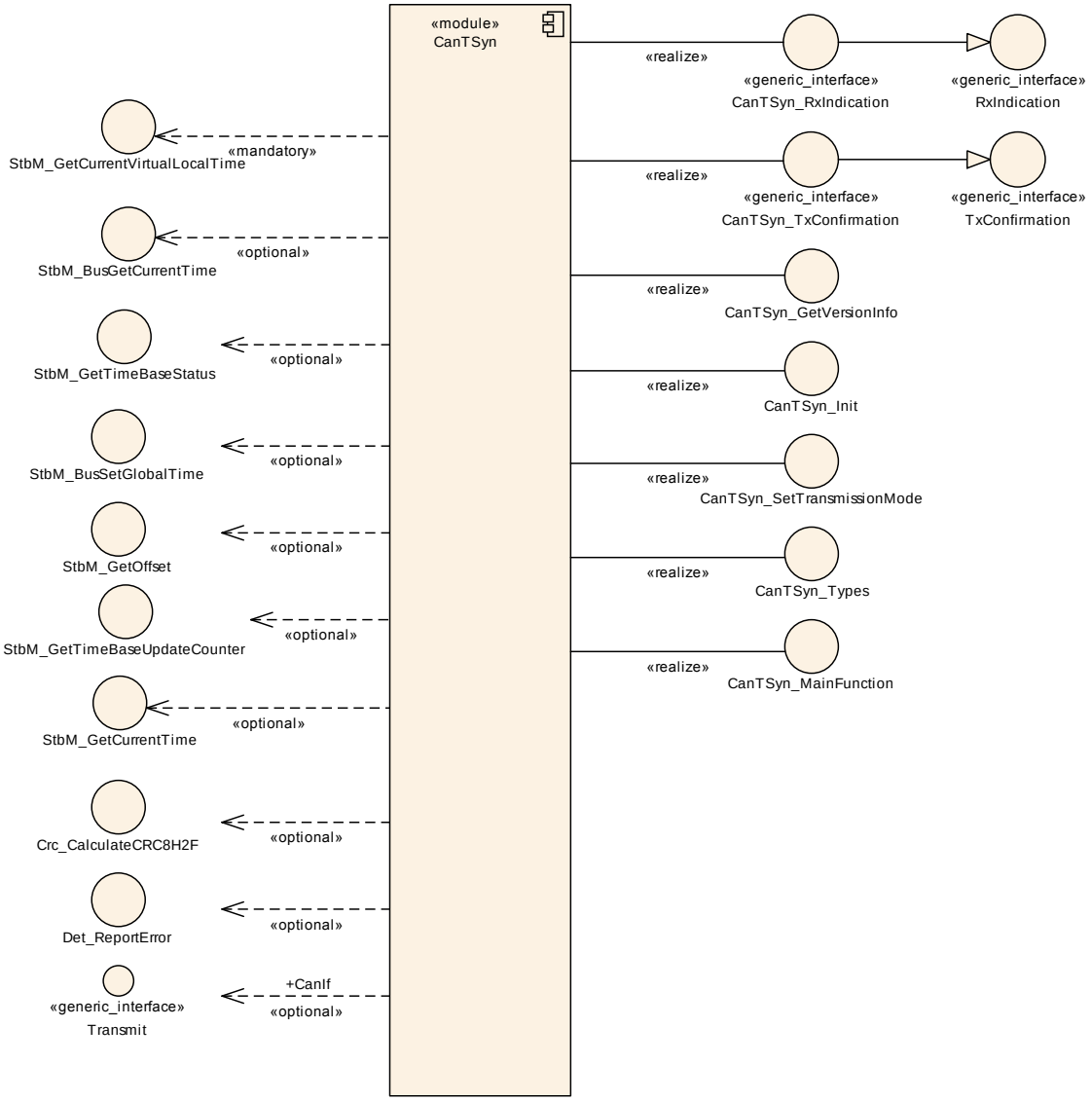


Figure 2: Module dependencies of the CanTSyn module

- 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 traceability

Requirement	Description	Satisfied by
RS_TS_00003	The Implementation of Time Synchronization shall initialize the Local Time Base with zero at startup	SWS_CanTSyn_00003, SWS_CanTSyn_00006
RS_TS_00004	The Implementation of Time Synchronization shall initialize the Global Time Base with a configurable startup value.	SWS_CanTSyn_00003, SWS_CanTSyn_00006
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
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
RS_TS_20035	The Timesync over CAN module shall support a	SWS_CanTSyn_00008, SWS_CanTSyn_00010, SWS_CanTSyn_00011, SWS_CanTSyn_00015,

	protocol for precise time measurement and synchronization over CAN	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, 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
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	SWS_CanTSyn_00108, SWS_CanTSyn_00135

	roles for a Time Base	
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
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 [4].

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 GTS).

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_00006][
When `CanTSyn_Init()` is called in initialized state, the Time Synchronization over CAN shall re-initialize its internal variables.
](RS_TS_00003, RS_TS_00004)

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

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.
](RS_TS_20037)

[SWS_CanTSyn_00014][
User Data shall be mapped to the `StbM_UserDataType`, whereas the byte number given in the message and by the `StbM_UserDataType` shall match (User Byte 0 mapped to `StbM_UserDataType.userByte0` etc.). Afterwards `StbM_UserDataType.userDataLength` shall be set accordingly.
](RS_TS_20037)

7.3.1 SYNC and FUP Message

[SWS_CanTSyn_00015][
SYNC not CRC secured message format:
Byte 0: *Type* = 0x10
Byte 1: User Byte 1, default: 0
Byte 2: *D* = Time Domain 0 to 15 (Bit 7 to Bit 4)
SC = Sequence Counter (Bit 3 to Bit 0)
Byte 3: User Byte 0, default: 0
Byte 4-7: *SyncTimeSec* = 32 bit LSB of the 48 bits seconds part of the time
If `CanTSynUseExtendedMsgFormat` = `TRUE`:
Byte 8-15: reserved, always 0
](RS_TS_20033, RS_TS_20035, RS_TS_20068)

[SWS_CanTSyn_00016][
FUP not CRC secured message format:
Byte 0: *Type* = 0x18
Byte 1: User Byte 2, default: 0

Byte 2: *D* = Time Domain 0 to 15 (Bit 7 to Bit 4)
SC = Sequence Counter (Bit 3 to Bit 0)
Byte 3: reserved (Bit 7 to Bit 3), default: 0
SGW (Bit 2)
SyncToGTM = 0
SyncToSubDomain = 1
OVS = Overflow of seconds (Bit 1 to Bit 0)
Byte 4-7: *SyncTimeNSec* = 32 Bit time value in nanoseconds
If `CanTSynUseExtendedMsgFormat` = `TRUE`:
Byte 8-15: reserved, always 0
](RS_TS_20033, RS_TS_20035, RS_TS_20068)

[SWS_CanTSyn_00017][
SYNC CRC secured message format:
Byte 0: *Type* = 0x20
Byte 1: *CRC*
Byte 2: *D* = Time Domain 0 to 15 (Bit 7 to Bit 4)
SC = Sequence Counter (Bit 3 to Bit 0)
Byte 3: User Byte 0, default: 0
Byte 4-7: *SyncTimeSec* = 32 bit LSB of the 48 bits seconds part of the time
If `CanTSynUseExtendedMsgFormat` = `TRUE`:
Byte 8-15: reserved, always 0
](RS_TS_20033, RS_TS_20035, RS_TS_20068)

[SWS_CanTSyn_00018][
FUP CRC secured message format:
Byte 0: *Type* = 0x28
Byte 1: *CRC*
Byte 2: *D* = Time Domain 0 to 15 (Bit 7 to Bit 4)
SC = Sequence Counter (Bit 3 to Bit 0)
Byte 3: reserved (Bit 7 to Bit 3), default: 0
SGW (Bit 2)
SyncToGTM = 0
SyncToSubDomain = 1
OVS = Overflow of seconds (Bit 1 to Bit 0)
Byte 4-7: *SyncTimeNSec* = 32 Bit time value in nanoseconds
If `CanTSynUseExtendedMsgFormat` = `TRUE`:
Byte 8-15: reserved, always 0
](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 as specified in section 7.3.2.1.
](RS_TS_20068)

[SWS_CanTSyn_00131][
If `CanTSynUseExtendedMsgFormat` is `TRUE`, then the Extended Offset Message Format shall be used as specified in section 7.3.2.2.
](RS_TS_20068)

7.3.2.1 Normal Offset Messages

[SWS_CanTSyn_00126][
OFS not CRC secured message format:
Byte 0: *Type* = 0x34
Byte 1: User Byte 1, default: 0
Byte 2: *D* = Time Domain 16 to 31 (Bit 7 to Bit 4)
SC = Sequence Counter (Bit 3 to Bit 0)
Byte 3: User Byte 0, default: 0
Byte 4-7: *OfsTimeSec* = 32 Bit offset time value in seconds
](RS_TS_20033, RS_TS_20036)

[SWS_CanTSyn_00127][
OFNS not CRC secured message format:
Byte 0: *Type* = 0x3C
Byte 1: User Byte 2, default: 0
Byte 2: *D* = Time Domain 16 to 31 (Bit 7 to Bit 4)
SC = Sequence Counter (Bit 3 to Bit 0)
Byte 3: reserved (Bit 7 to Bit 1), default: 0
SGW (Bit 0)
SyncToGTM = 0
SyncToSubDomain = 1
Byte 4-7: *OfsTimeNSec* = 32 Bit offset time value in nanoseconds
](RS_TS_20033, RS_TS_20036)

[SWS_CanTSyn_00128][
OFS CRC secured message format:
Byte 0: *Type* = 0x44
Byte 1: *CRC*
Byte 2: *D* = Time Domain 16 to 31 (Bit 7 to Bit 4)
SC = Sequence Counter (Bit 3 to Bit 0)
Byte 3: User Byte 0, default: 0
Byte 4-7: *OfsTimeSec*= 32 Bit offset time value in seconds
](RS_TS_20033, RS_TS_20036)

[SWS_CanTSyn_00129][
OFNS CRC secured message format:
Byte 0: *Type* = 0x4C
Byte 1: *CRC*
Byte 2: *D* = Time Domain 16 to 31 (Bit 7 to Bit 4)
SC = Sequence Counter (Bit 3 to Bit 0)
Byte 3: reserved (Bit 7 to Bit 1), default: 0
SGW (Bit 0)
SyncToGTM = 0
SyncToSubDomain = 1
Byte 4-7: *OfsTimeNSec* = 32 Bit offset time value in nanoseconds
](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 0: *Type* = 0x54
Byte 1: User Byte 2, default: 0
Byte 2: *D* = Time Domain 16 to 31 (Bit 7 to Bit 4)
SC = Sequence Counter (Bit 3 to Bit 0)
Byte 3: reserved (Bit 7 to Bit 1), default: 0
SGW (Bit 0)
SyncToGTM = 0
SyncToSubDomain = 1
Byte 4: User Byte 0, default: 0
Byte 5: User Byte 1, default: 0
Byte 6: reserved, default: 0
Byte 7: reserved, default: 0
Byte 8-11: *OfsTimeSec* = 32 Bit offset time value in seconds
Byte 12-15: *OfsTimeNSec* = 32 Bit offset time value in nanoseconds
](RS_TS_20033, RS_TS_20036, RS_TS_20068)

[**SWS_CanTSyn_00112**][
OFS CRC secured message format for CAN FD PDUs:
Byte 0: *Type* = 0x64
Byte 1: *CRC*
Byte 2: *D* = Time Domain 16 to 31 (Bit 7 to Bit 4)
SC = Sequence Counter (Bit 3 to Bit 0)
Byte 3: reserved (Bit 7 to Bit 1), default: 0
SGW (Bit 0)
SyncToGTM = 0
SyncToSubDomain = 1
Byte 4: User Byte 0, default: 0
Byte 5: User Byte 1, default: 0
Byte 6: reserved, default: 0

Byte 7: reserved, default: 0
Byte 8-11: *OfsTimeSec* = 32 Bit offset time value in seconds
Byte 12-15: *OfsTimeNSec* = 32 Bit offset time value in nanoseconds
](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. 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.

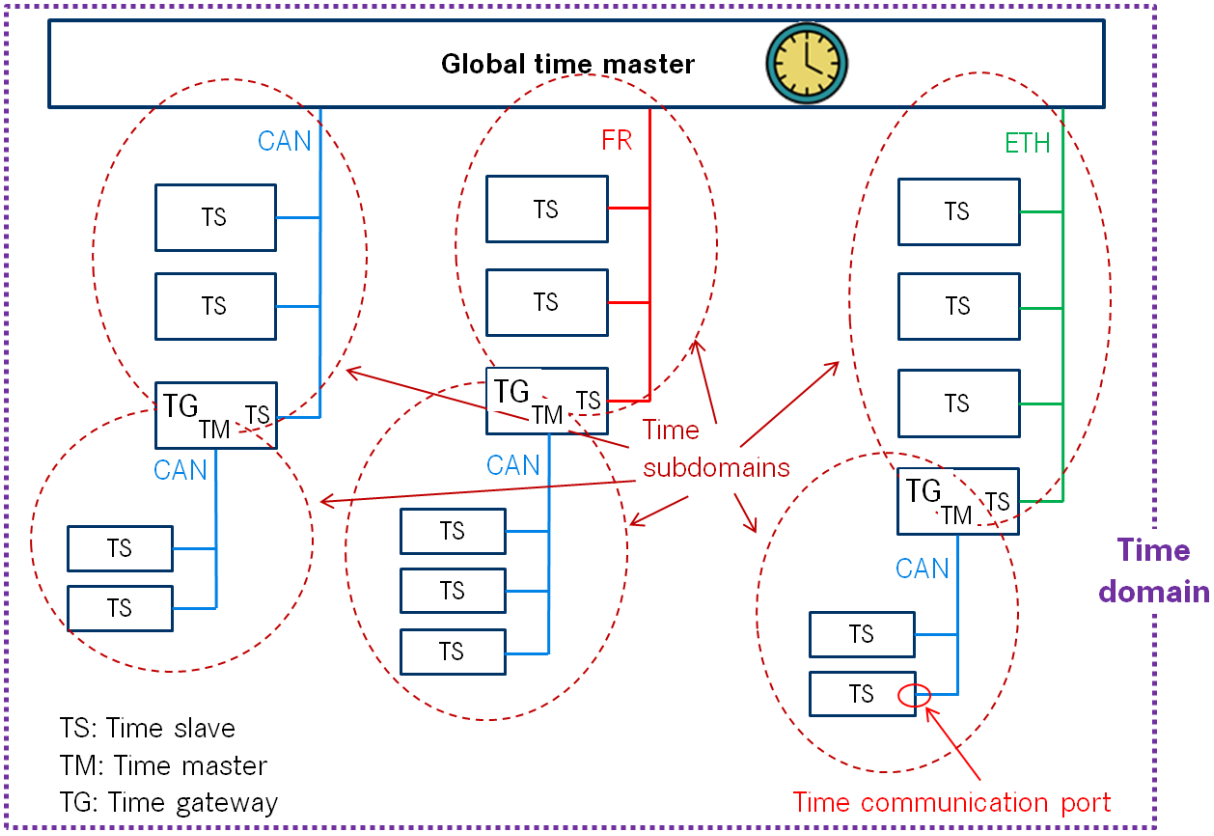


Figure 3: Terminology Example

[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][
Any timeout while waiting for `CanTSyn_TxConfirmation()` function resets the state machine to start with a new SYNC transmission again.
](RS_TS_20034, RS_TS_20035)

[SWS_CanTSyn_00028][
For a Synchronized Time Base a Time Master is using a cyclic transmission of SYNC messages (according to 9.1) with `CanTSynGlobalTimeTxPeriod` **(ECUC_CanTSyn_00017 :)** 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 (see 7.4.5).
](RS_TS_20031, RS_TS_20035)

[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` (**ECUC_CanTSyn_00015 :**) the SYNC / FUP message shall be of type:

`CanTSynGlobalTimeTxCrcSecured`	SYNC	FUP
`CRC_NOT_SUPPORTED`	`0x10` SYNC not CRC secured message	`0x18` FUP not CRC secured message
`CRC_SUPPORTED`	`0x20` SYNC CRC	`0x28` FUP CRC secured

	secured message	message

](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 as described in 7.4.4.
](RS_TS_20031, RS_TS_20035)

[SWS_CanTSyn_00033][
Each transmission request of a SYNC message shall be monitored for a transmit confirmation timeout `CanTSynMasterConfirmationTimeout` (**ECUC_CanTSyn_00020 :**). If the timeout occurs, the transmission request shall be revoked and no FUP message shall be sent.
](RS_TS_20034, RS_TS_20035)

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][
Any Timeout while waiting for `CanTSyn_TxConfirmation()` function resets the state machine to start with a new OFS transmission again.
](RS_TS_20034, RS_TS_20036)

[SWS_CanTSyn_00038][
For an Offset Time Base the Time Master is using a cyclic transmission of OFS messages (`CanTSynGlobalTimeTxPeriod` (refer **ECUC_CanTSyn_00017 :**) 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 (see 7.4.5).
](RS_TS_20031, RS_TS_20036)

[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 as described in 7.4.4.
](RS_TS_20036)

[SWS_CanTSyn_00041][
Depending on `CanTSynGlobalTimeTxCrcSecured` (**ECUC_CanTSyn_00015 :**) the OFS / OFNS message shall be of type:

	`CanTSynGlobalTimeTxCrcSecured`	OFS	OFNS
`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 (CanTSynUseExtendedMsgFormat = TRUE)`	`CRC_NOT_SUPPORTED`	`0x54` OFS not CRC secured message	`Not available`
	`CRC_SUPPORTED`	`0x64` OFS CRC secured message	

](RS_TS_20033, RS_TS_20036, RS_TS_20068)

[SWS_CanTSyn_00042][
Each OFS transmission request shall be monitored for a transmit confirmation timeout `CanTSynMasterConfirmationTimeout` (**ECUC_CanTSyn_00020 :**). If the timeout occurs, the transmission request shall be revoked and no OFNS message shall be sent.
](RS_TS_20034, RS_TS_20036)

7.4.3 Transmission mode

[SWS_CanTSyn_00043][
If `CanTSyn_SetTransmissionMode(Controller, Mode)` is called and parameter `Mode` equals `CANTSYN_TX_OFF`, all transmit request 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 request 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` (**ECUC_CanTSyn_00045 :**) 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` (**ECUC_CanTSyn_00045 :**) represents the debounce value of a PDU specific `debounceCounter` that shall be started after the Timesync PDU has been sent. CanTSyn shall decrement the `debounceCounter` value on each invocation of `CanTSyn_MainFunction()`, if no Timesync PDU is transmitted.
](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 (**ECUC_CanTSyn_00043 :**) 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 (**ECUC_CanTSyn_00043 :**) is set to TRUE and the timeBaseUpdateCounter of a Time Base has changed and the GLOBAL_TIME_BASE bit of the timeBaseStatus is set, 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().

Note: The debounceTimer as described in 7.4.4 shall always be considered.

[SWS_CanTSyn_00119][
If CanTSynImmediateTimeSync (**ECUC_CanTSyn_00043 :**) is set to TRUE, cyclicMsgResumeCounter and CanTSynCyclicMsgResumeTime (**ECUC_CanTSyn_00044 :**) shall be considered.
](RS_TS_20031)

[SWS_CanTSyn_00120][
CanTSynCyclicMsgResumeTime (**ECUC_CanTSyn_00044 :**) represents the timeout value of a cyclicMsgResumeCounter that shall be started when either a SYNC or OFS message has been sent immediately, asynchronous 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

[SWS_CanTSyn_00045][
The transmitter of a Synchronized Time Base (Time Master) shall perform the following steps to distribute the Synchronized Time Base exactly (refer to 9.1):

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 $T4 = T0_{SYNCns} + (T1_{VLT} - T0_{VLT})$ with $T0_{SYNCns}$ as nanosecond portion of $T0_{SYNC}$

3. On transmission of FUP message
   a. Write second portion of T4 (T4 >= 1s) to *OVS*
   b. Write nanosecond portion of T4 to *SyncTimeNSec*

](RS_TS_20035)

With these steps, the Synchronized Time Base value at the transmitter side has been calculated ($T0_{SYNC}$ + T4).

Note:
When using interrupt mode with interrupt nesting disabled, the CanTSyn does not need to explicitly establish a protection against interruptions in the Tx confirmation callback. This is typically done implicitly by the controller.

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

1. Retrieve current Offset Time via `StbM_GetOffset()`
2. Write second portion of the Offset Time to *OfsTimeSec*
3. Write nanosecond portion of the Offset Time to *OfsTimeNSec*

](RS_TS_20036)

Note: OFS and OFNS messages shall not be 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` **[SWS_CanTSyn_00045]** whereas the left over 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 `STBM_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 [5] 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` (**ECUC_CanTSyn_00024 : ECUC_CanTSyn_00025 : ECUC_CanTSyn_00026 : ECUC_CanTSyn_00041 :**) is given by configuration for each message *Type*.
](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 `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_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 (`CanTSynGlobalTimeSlave`) the CanTSyn module shall observe the *reception timeout* `CanTSynGlobalTimeFollowUpTimeout` (**ECUC_CanTSyn_00006 :**) 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 a new Time Tuple, consisting of the Global Time value and the associated value of the Virtual Local Time, shall be calculated and forwarded to the StbM module via `StbM_BusSetGlobalTime()` (according to 9.2).
](RS_TS_20032, RS_TS_20034)

[SWS_CanTSyn_00115][
On an invocation of `StbM_BusSetGlobalTime()` the parameter `PathDelay` of the `measureDataPtr` structure shall be set to 0.
](RS_TS_20034)

7.5.2 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 CanTSyn 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_NOT_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` (**ECUC_CanTSyn_00006 :**) 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

[SWS_CanTSyn_00073][
The receiver of a Synchronized Time Base shall perform the following steps to retrieve the Synchronized Time Base exactly (refer to 9.2):

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 `MainFunction` 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)

Note: Immediately protecting against interruptions means that there shall be no frame checks before. If called in context of the Rx interrupt **with interrupt nesting disabled**, this is typically implicitly ensured by the controller. 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 the snapshot. Once the snapshot has been taken it is ok to remove the protection against interruptions and to make the necessary validations. This means that a snapshot of the Virtual Local Time shall be taken even if the succeeding validations fail and thus making the snapshot superfluous.

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

1. Get second portion of the Offset Time out of *OfsTimeSec*
2. Get nanosecond 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 *SC*s 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.
The `CanTSynGlobalTimeSequenceCounterJumpWidth` value 0 is not allowed.
](RS_TS_20034, RS_TS_20035, RS_TS_20036)

[SWS_CanTSyn_00079][
At Startup or if a Time Base update timeout has been detected (TIMEOUT bit set in Time Base synchronization status `timeBaseStatus`), a Time Slave shall not check the Sequence Counter of the 1st received SYNC (or OFS) message per Time Domain against the defined Sequence Counter Jump Width.
](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.

7.5.3.5 CRC Validation

[SWS_CanTSyn_00080][
The function `Crc_CalculateCRC8H2F()` as defined in [5] 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 Error Classification

This chapter lists and classifies all errors that can be detected by this software module. Each error is classified to relevance (development / production) and the related error code (unique label for the error). For development errors this table also specifies the unique values, which correspond to the error codes.

[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)

7.6.1 Development Errors

The detection of development errors is configurable (see section 10.2, `CanTSynDevErrorDetect`).

[SWS_CanTSyn_00089][
CanTSyn shall use the following errors:

Type or error	*Related error code*	*Value [hex]*
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)

7.6.2 Runtime Errors

No Runtime Errors defined.

7.6.3 Transient Faults

No Transient Faults defined.

7.6.4 Production Errors

No Production Errors defined.

7.6.5 Extended Production Errors

No Extended Production Errors defined.

8 API specification

8.1 API

8.1.1 Imported types

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

[SWS_CanTSyn_00090] [

Module	*Header File*	*Imported Type*
ComStack_Types	ComStackTypes.h	PduIdType
	ComStackTypes.h	PduInfoType
StbM	Rte_StbM_Type.h	StbM_SynchronizedTimeBaseType
	Rte_StbM_Type.h	StbM_TimeBaseStatusType
	Rte_StbM_Type.h	StbM_TimeStampType
	Rte_StbM_Type.h	StbM_UserDataType
	StbM.h	StbM_MeasurementType
	StbM.h	StbM_VirtualLocalTimeType
Std_Types	StandardTypes.h	Std_ReturnType
	StandardTypes.h	Std_VersionInfoType

] (RS_TS_20035)

8.1.2 Type definitions

8.1.2.1 CanTSyn_ConfigType

[SWS_CanTSyn_00091] [

Name:	`CanTSyn_ConfigType`		
Type:	`Structure`		
Element:	`void`	`implementation specific`	--
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)

8.1.2.2 CanTSyn_TransmissionModeType

[SWS_CanTSyn_00092] [

Name:	`CanTSyn_TransmissionModeType`		
Type:	`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)

8.1.3 Function definitions

8.1.3.1 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 in [reference to SWS BSW General] by SWS_BSW_00050.

See section 7.2.1 for details.

8.1.3.2 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)

8.1.3.3 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 (`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)

8.1.4 Call-back notifications

This is a list of functions provided for other modules.

8.1.4.1 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	None	

(inout):	
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.

8.1.4.2 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.

8.1.5 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.

8.1.5.1 CanTSyn_MainFunction

[SWS_CanTSyn_00102] [

Service name:	CanTSyn_MainFunction

Syntax:	`void CanTSyn_MainFunction(`
	`  void`
	`)`

Service ID[hex]:	0x06
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
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` (refer to **ECUC_CanTSyn_00019 :**).
](RS_TS_20035)

8.1.6 Expected Interfaces

In this section, all interfaces required by other modules are listed.

8.1.6.1 Mandatory Interfaces

This section defines all interfaces that are required to fulfill a mandatory 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)

8.1.6.2 Optional Interfaces

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

[SWS_CanTSyn_00106] ⌈

API function	*Header File*	*Description*
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.
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., the Received Time Tuple) to the StbM.
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)

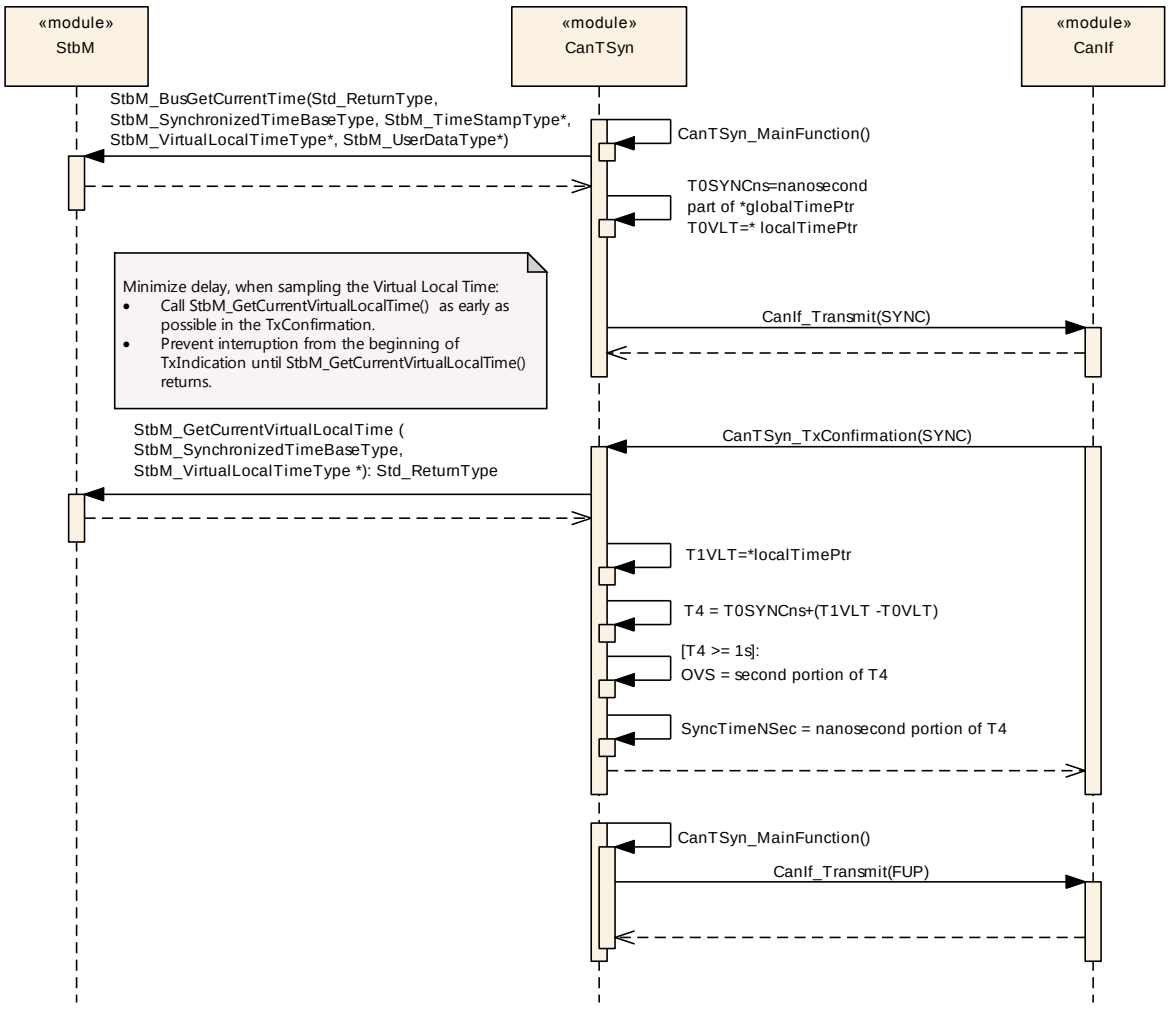


Figure 4: CAN Time Synchronization (Time Master)

9.2 CAN Time Synchronization (Time Slave)

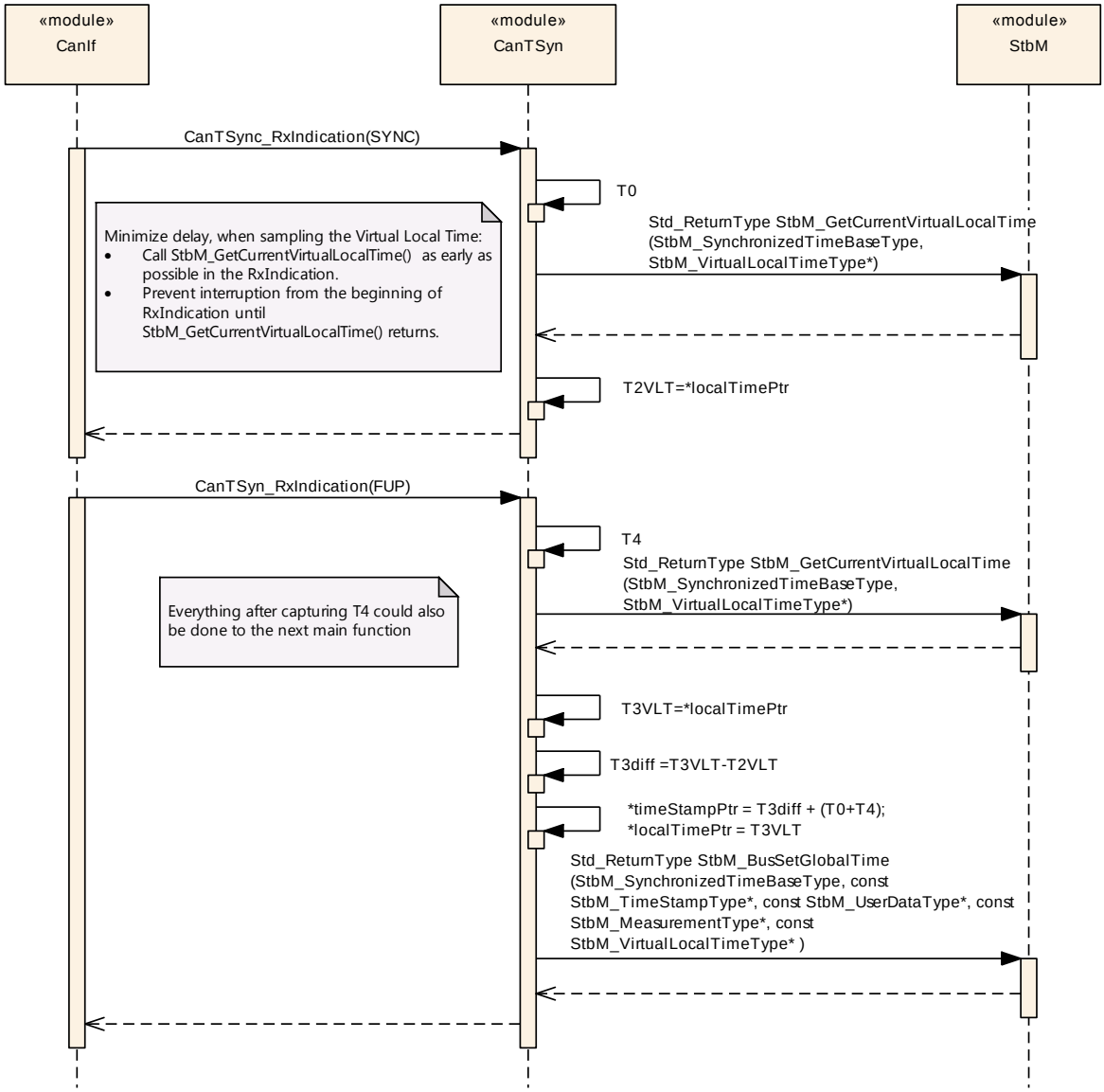


Figure 5: CAN Time Synchronization (Time Slave)

10 Configuration specification

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

Section 10.2 specifies the structure (containers) and the parameters of the Time Synchronization over CAN.

Section 10.2.16 specifies published information of the Time Synchronization over CAN.

10.1 How to read this chapter

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

10.2 Containers and configuration parameters

The following sections summarize all configuration parameters of the Time Synchronization over CAN. The detailed meaning of the parameters is described in chapters 7 and 8.

10.2.1 Variants

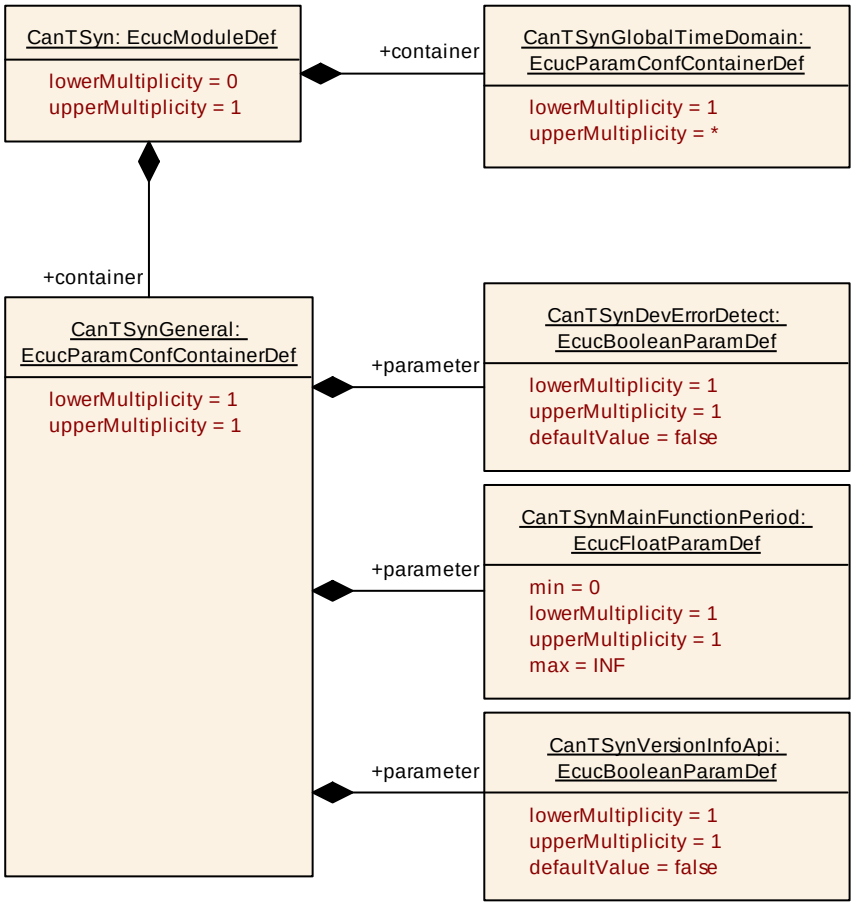
[SWS_CanTSyn_00108][
The Time Synchronization over CAN shall support the configuration for Time Master, Time Slave and Time Gateway.
](RS_TS_20038)

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

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
CanTSynGlobalTimeDomain	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
Description	This container holds the general parameters of the CAN-specific Synchronized Time-base Manager
Configuration Parameters	

SWS Item	**ECUC_CanTSyn_00002 :**		
Name	CanTSynDevErrorDetect		
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		

SWS Item	**ECUC_CanTSyn_00019 :**		
Name	CanTSynMainFunctionPeriod		
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		

SWS Item	**ECUC_CanTSyn_00023 :**		
Name	CanTSynVersionInfoApi		
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

10.2.4 CanTSynGlobalTimeDomain

SWS Item	**ECUC_CanTSyn_00004 :**
Container Name	CanTSynGlobalTimeDomain
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	

SWS Item	**ECUC_CanTSyn_00005 :**	
Name	CanTSynGlobalTimeDomainId	
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		

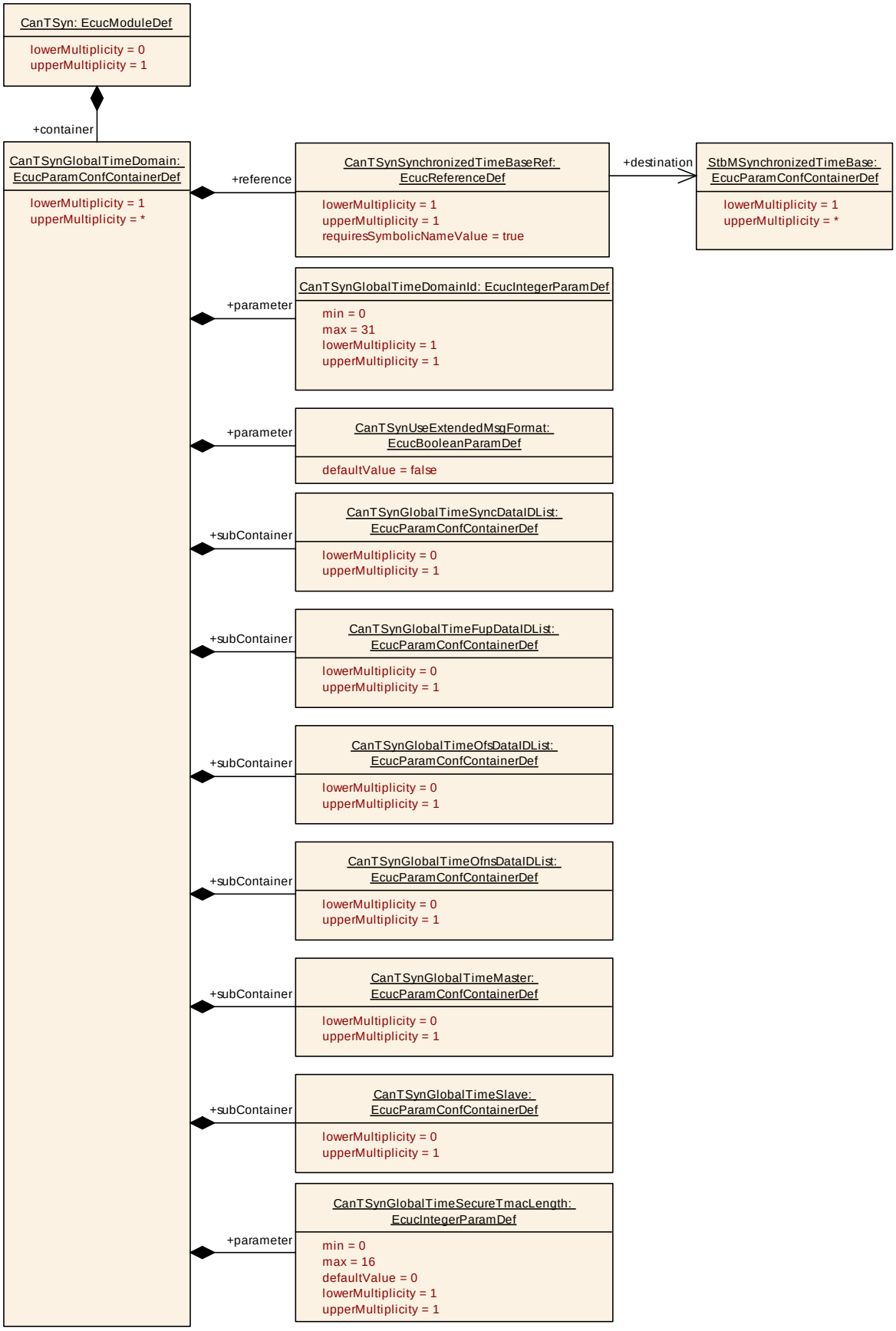
SWS Item	**ECUC_CanTSyn_00046 :**		
Name	CanTSynGlobalTimeSecureTmacLength		
Parent Container	CanTSynGlobalTimeDomain		
Description	Represents the number of bytes for the used Truncated Message Authentication Code (TMAC). If 0, no message authentication will be used. **Tags:** atp.Status=draft		
Multiplicity	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		

SWS Item	**ECUC_CanTSyn_00042 :**		
Name	CanTSynUseExtendedMsgFormat		
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		

SWS Item	**ECUC_CanTSyn_00022 :**		
Name	CanTSynSynchronizedTimeBaseRef		
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

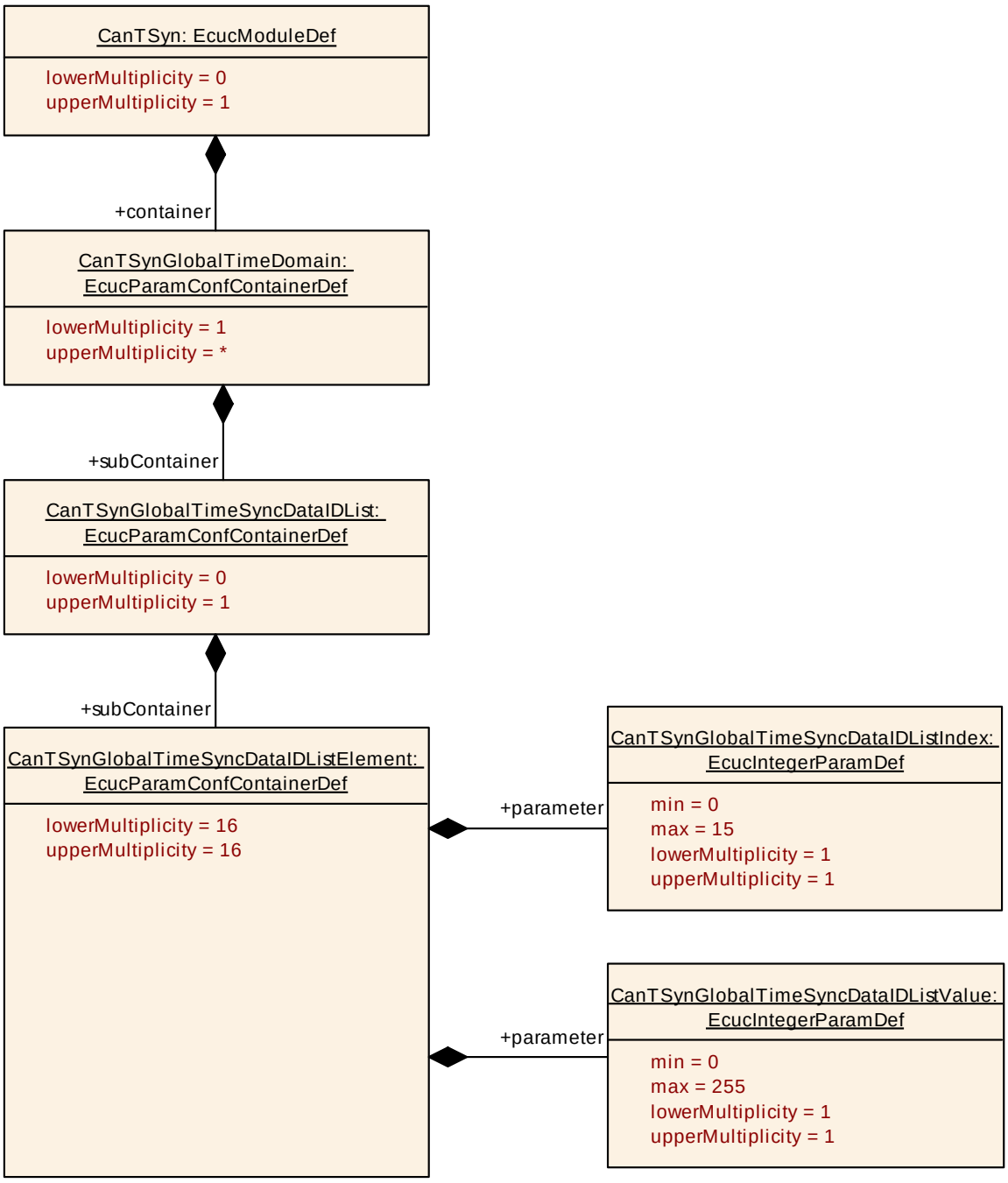
Container Name	*Multiplicity*	*Scope / Dependency*
CanTSynGlobalTimeFupDataIDList	0..1	The DataIDList for FUP messages ensures the identification of data elements due to CRC calculation and message authentication process.
CanTSynGlobalTimeMaster	0..1	Configuration of the global time master. Each global time domain is required to have exactly one global time master. This master may or may not exist on the configured ECU.
CanTSynGlobalTimeOfnsDataIDList	0..1	The DataIDList for OFNS messages ensures the identification of data elements due to CRC calculation and message authentication process.
CanTSynGlobalTimeOfsDataIDList	0..1	The DataIDList for OFS messages ensures the identification of data elements due to CRC calculation and message authentication process.
CanTSynGlobalTimeSlave	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.
CanTSynGlobalTimeSyncDataIDList	0..1	The DataIDList for SYNC messages ensures the identification of data elements due to CRC calculation and message authentication process.


CanTSyn: EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
CanTSynGlobalTimeDomain: EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+reference
CanTSynSynchronizedTimeBaseRef: EcucReferenceDef
lowerMultiplicity = 1
upperMultiplicity = 1
requiresSymbolicNameValue = true
+destination
StbMSynchronizedTimeBase: EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
CanTSynGlobalTimeDomainId: EcucIntegerParamDef
min = 0
max = 31
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
CanTSynUseExtendedMsgFormat: EcucBooleanParamDef
defaultValue = false
+subContainer
CanTSynGlobalTimeSyncDataIDList: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
CanTSynGlobalTimeFupDataIDList: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
CanTSynGlobalTimeOfsDataIDList: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
CanTSynGlobalTimeOfnsDataIDList: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
CanTSynGlobalTimeMaster: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
CanTSynGlobalTimeSlave: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
CanTSynGlobalTimeSecureTmacLength: EcucIntegerParamDef
min = 0
max = 16
defaultValue = 0
lowerMultiplicity = 1
upperMultiplicity = 1

10.2.5 CanTSynGlobalTimeSyncDataIDList

SWS Item	**ECUC_CanTSyn_00024 :**		
Container Name	CanTSynGlobalTimeSyncDataIDList		
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***
CanTSynGlobalTimeSyncDataIDListElement	16	Element of the DataIDList for SYNC messages ensures the identification of data elements due to CRC calculation and message authentication process.



10.2.6 CanTSynGlobalTimeSyncDataIDListElement

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

SWS Item	**ECUC_CanTSyn_00029 :**
Name	CanTSynGlobalTimeSyncDataIDListIndex

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		

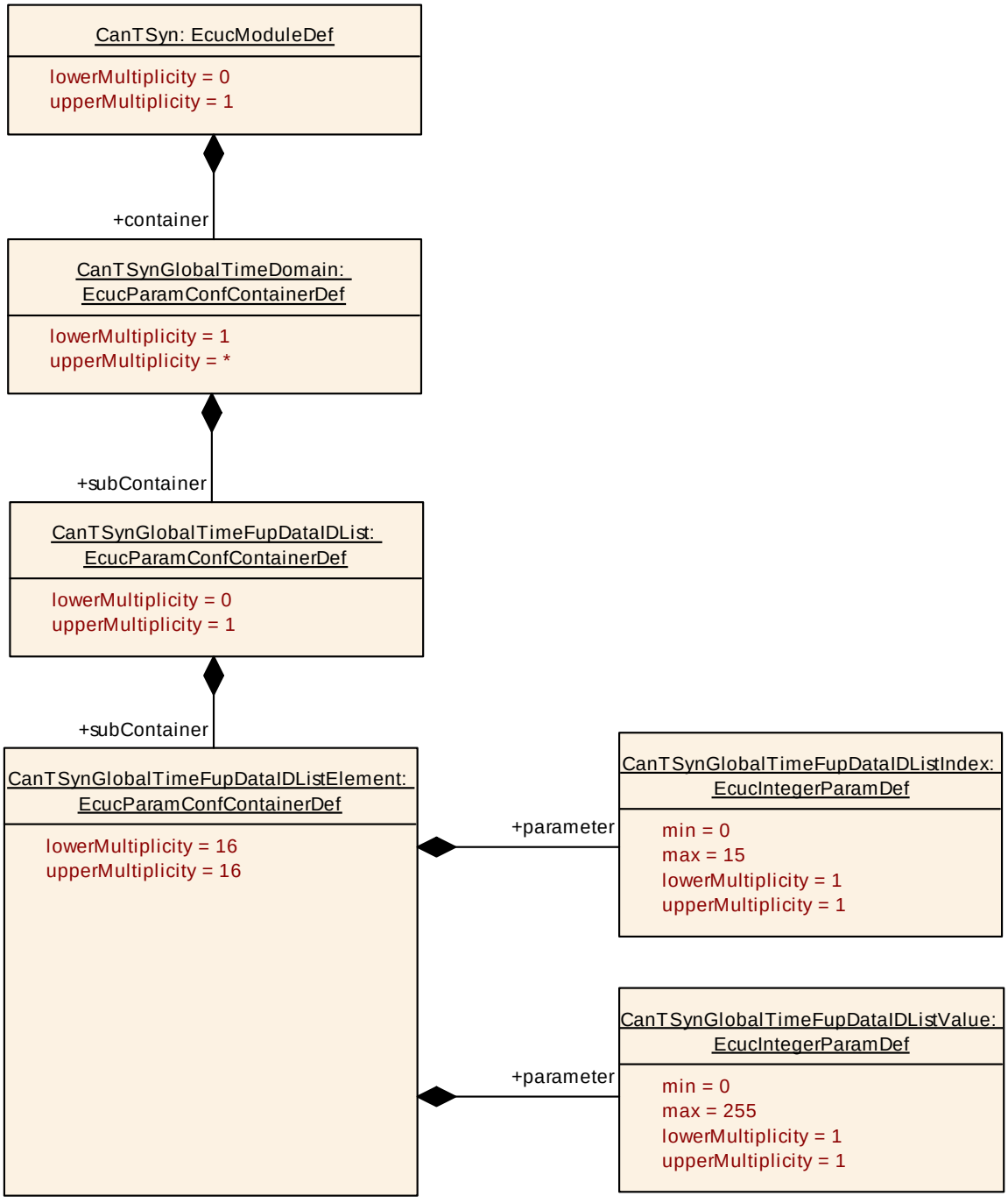
SWS Item	**ECUC_CanTSyn_00030 :**		
Name	CanTSynGlobalTimeSyncDataIDListValue		
Parent Container	CanTSynGlobalTimeSyncDataIDListElement		
Description	Value of the DataIDList for SYNC messages ensures the identification of data elements due to CRC calculation and message authentication process.		
Multiplicity	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		
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***
CanTSynGlobalTimeFupDataIDListElement	16	Element of the DataIDList for FUP messages ensures the identification of data elements due to CRC calculation and message authentication process.



10.2.8 CanTSynGlobalTimeFupDataIDListElement

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

SWS Item	**ECUC_CanTSyn_00032 :**		
Name	CanTSynGlobalTimeFupDataIDListIndex		
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		

SWS Item	**ECUC_CanTSyn_00033 :**		
Name	CanTSynGlobalTimeFupDataIDListValue		
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		

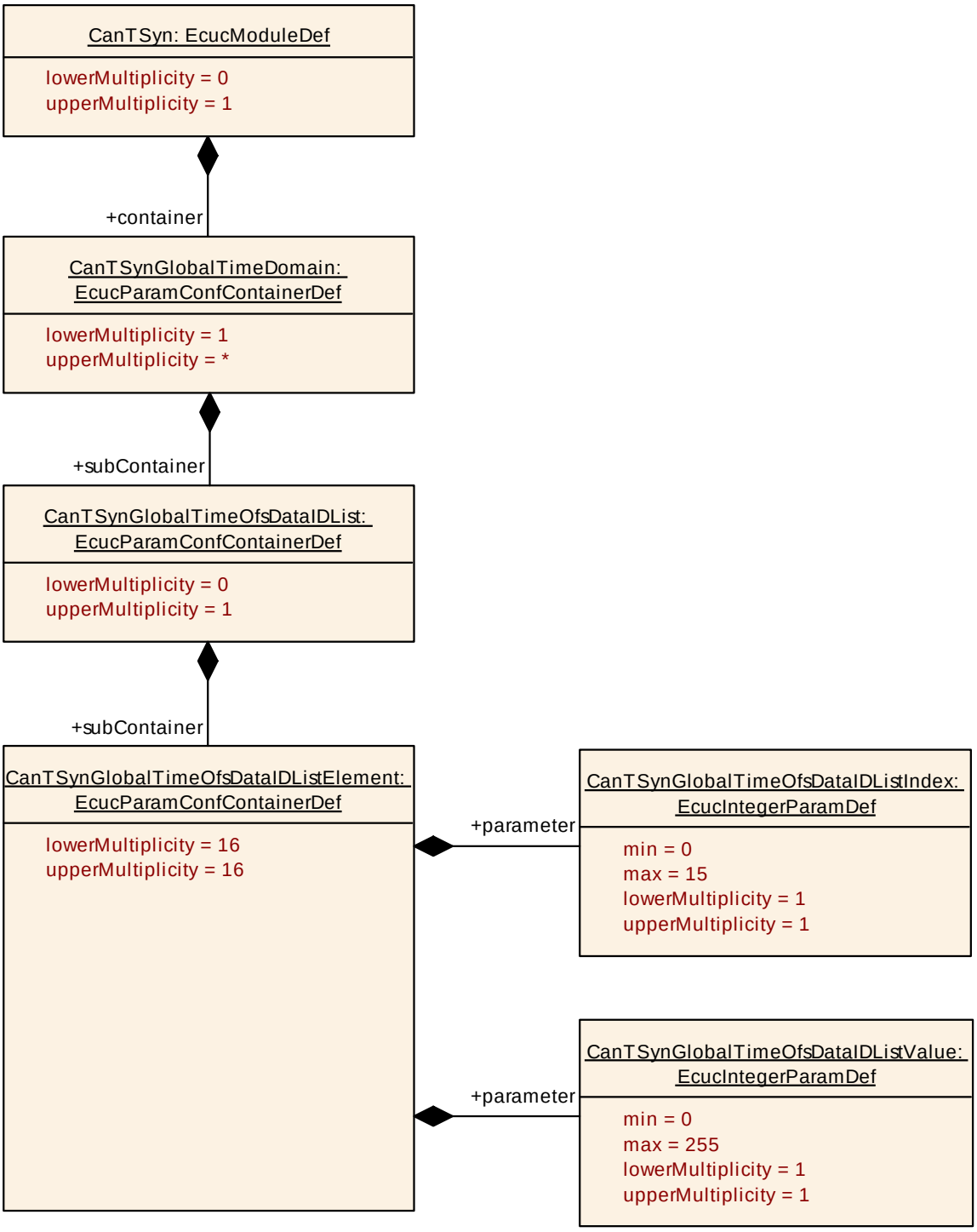
No Included Containers

10.2.9 CanTSynGlobalTimeOfsDataIDList

SWS Item	**ECUC_CanTSyn_00026 :**		
Container Name	CanTSynGlobalTimeOfsDataIDList		
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***
CanTSynGlobalTimeOfsDataIDListElement	16	Element of the DataIDList for OFS messages ensures the identification of data elements due to CRC calculation and message authentication

		process.



10.2.10 CanTSynGlobalTimeOfsDataIDListElement

SWS Item	**ECUC_CanTSyn_00034 :**
Container Name	CanTSynGlobalTimeOfsDataIDListElement

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

SWS Item	**ECUC_CanTSyn_00035 :**		
Name	CanTSynGlobalTimeOfsDataIDListIndex		
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		

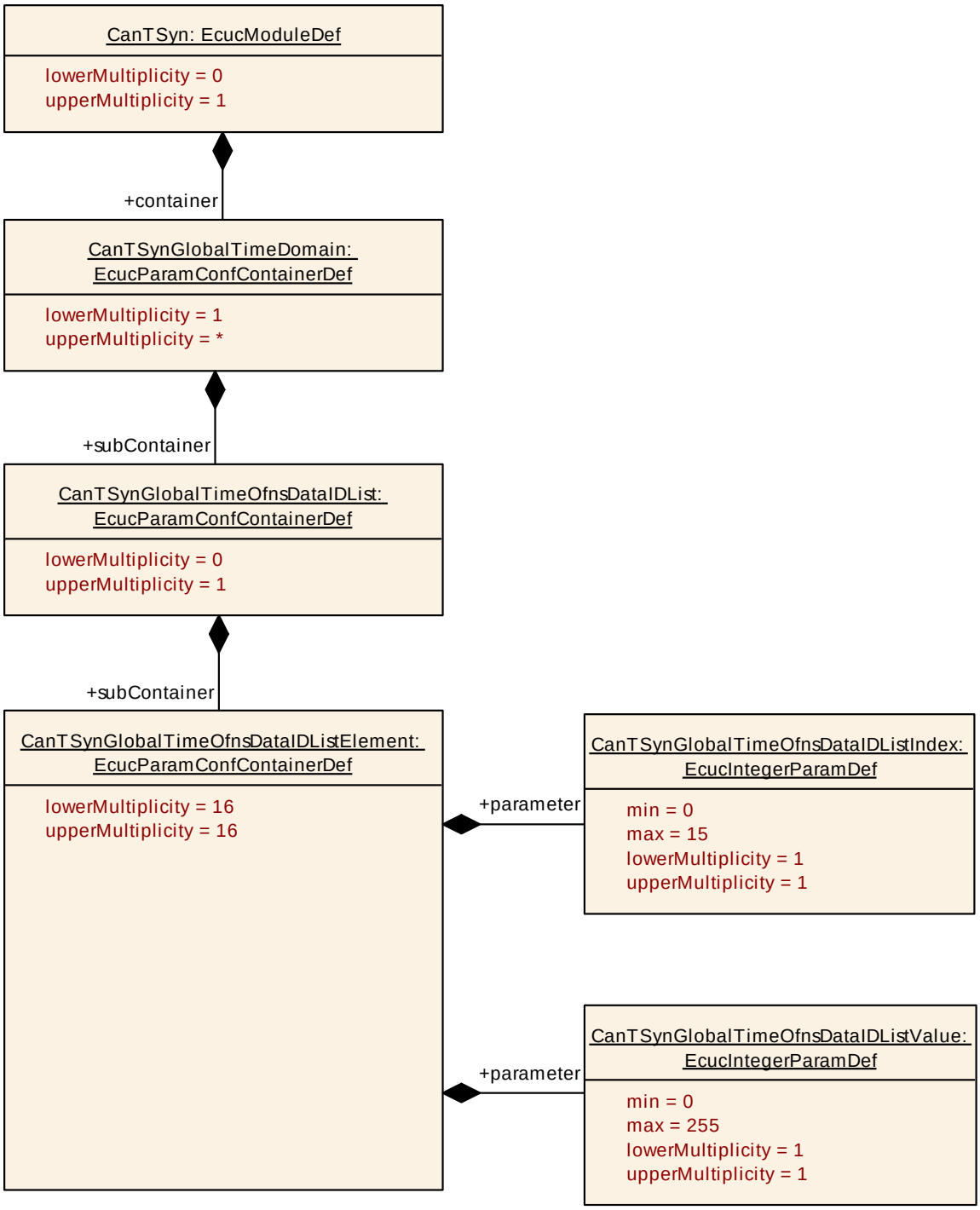
SWS Item	**ECUC_CanTSyn_00036 :**		
Name	CanTSynGlobalTimeOfsDataIDListValue		
Parent Container	CanTSynGlobalTimeOfsDataIDListElement		
Description	Value of the DataIDList for OFS messages ensures the identification of data elements due to CRC calculation and message authentication process.		
Multiplicity	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		
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***
CanTSynGlobalTimeOfnsDataIDListElement	16	Element of the DataIDList for OFNS messages ensures the identification of data elements due to CRC calculation and message authentication process.



10.2.12 CanTSynGlobalTimeOfnsDataIDListElement

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

SWS Item	**ECUC_CanTSyn_00038 :**		
Name	CanTSynGlobalTimeOfnsDataIDListIndex		
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		

SWS Item	**ECUC_CanTSyn_00039 :**		
Name	CanTSynGlobalTimeOfnsDataIDListValue		
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
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			

SWS Item	**ECUC_CanTSyn_00044 :**		
Name	CanTSynCyclicMsgResumeTime		
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		

SWS Item	**ECUC_CanTSyn_00045 :**		
Name	CanTSynGlobalTimeDebounceTime		
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		

SWS Item	**ECUC_CanTSyn_00015 :**		
Name	CanTSynGlobalTimeTxCrcSecured		
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		

SWS Item	ECUC_CanTSyn_00017 :		
Name	CanTSynGlobalTimeTxPeriod		
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		

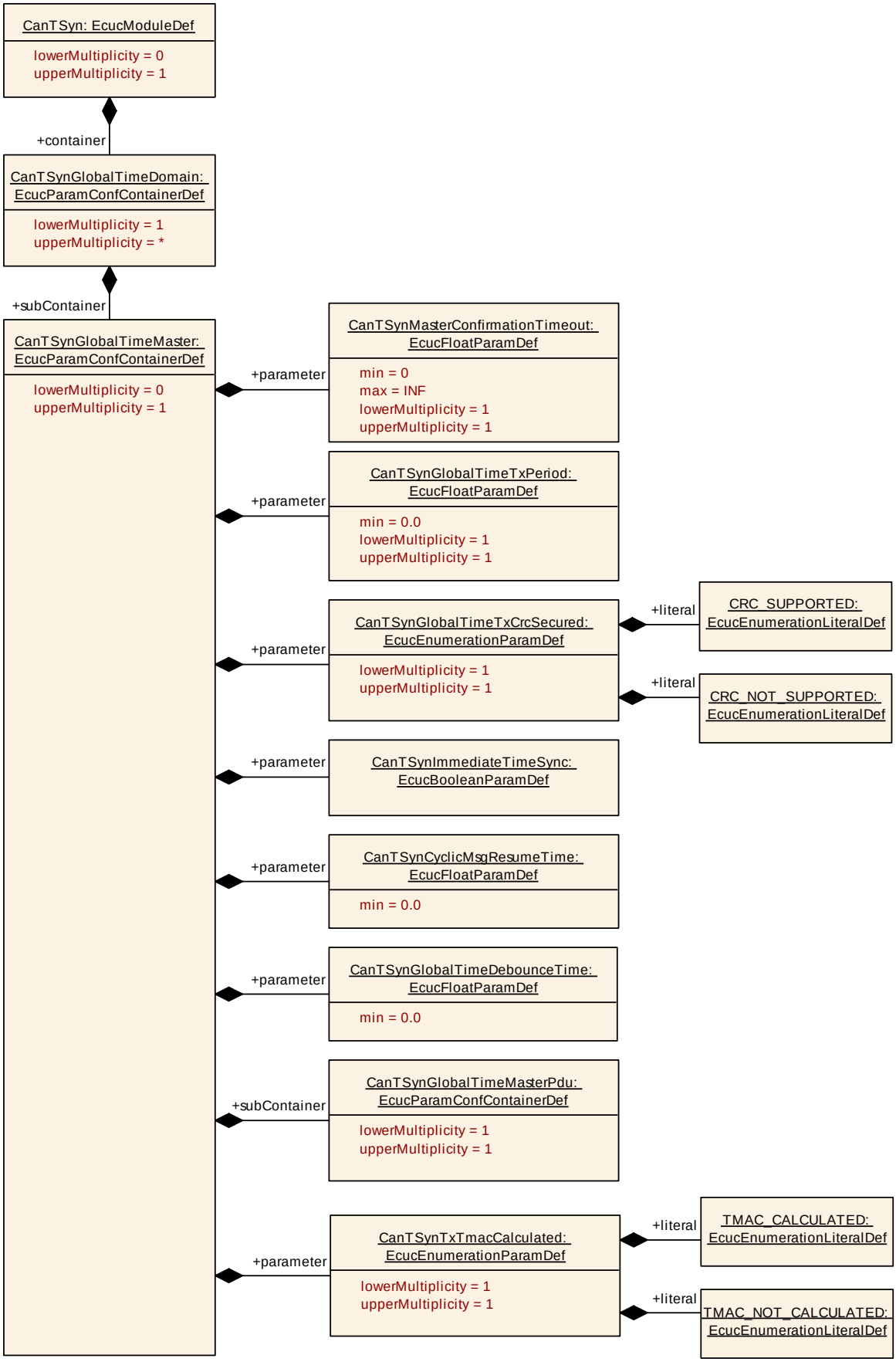
SWS Item	ECUC_CanTSyn_00043 :		
Name	CanTSynImmediateTimeSync		
Parent Container	CanTSynGlobalTimeMaster		
Description	Enables/Disables the cyclic polling of StbM_GetTimeBaseUpdateCounter() within CanTSyn_MainFunction().		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant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

SWS Item	ECUC_CanTSyn_00020 :		
Name	CanTSynMasterConfirmationTimeout		
Parent Container	CanTSynGlobalTimeMaster		
Description	This represents the confirmation timeout after transmission of each Timesync message. 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		

SWS Item	ECUC_CanTSyn_00047 :	
Name	CanTSynTxTmacCalculated	
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.


CanTSyn: EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
CanTSynGlobalTimeDomain: EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+subContainer
CanTSynGlobalTimeMaster: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
CanTSynMasterConfirmationTimeout: EcucFloatParamDef
min = 0
max = INF
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
CanTSynGlobalTimeTxPeriod: EcucFloatParamDef
min = 0.0
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
CanTSynGlobalTimeTxCrcSecured: EcucEnumerationParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
+literal
CRC_SUPPORTED: EcucEnumerationLiteralDef
+literal
CRC_NOT_SUPPORTED: EcucEnumerationLiteralDef
+parameter
CanTSynImmediateTimeSync: EcucBooleanParamDef
+parameter
CanTSynCyclicMsgResumeTime: EcucFloatParamDef
min = 0.0
+parameter
CanTSynGlobalTimeDebounceTime: EcucFloatParamDef
min = 0.0
+subContainer
CanTSynGlobalTimeMasterPdu: EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
CanTSynTxTmacCalculated: EcucEnumerationParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
+literal
TMAC_CALCULATED: EcucEnumerationLiteralDef
+literal
TMAC_NOT_CALCULATED: EcucEnumerationLiteralDef

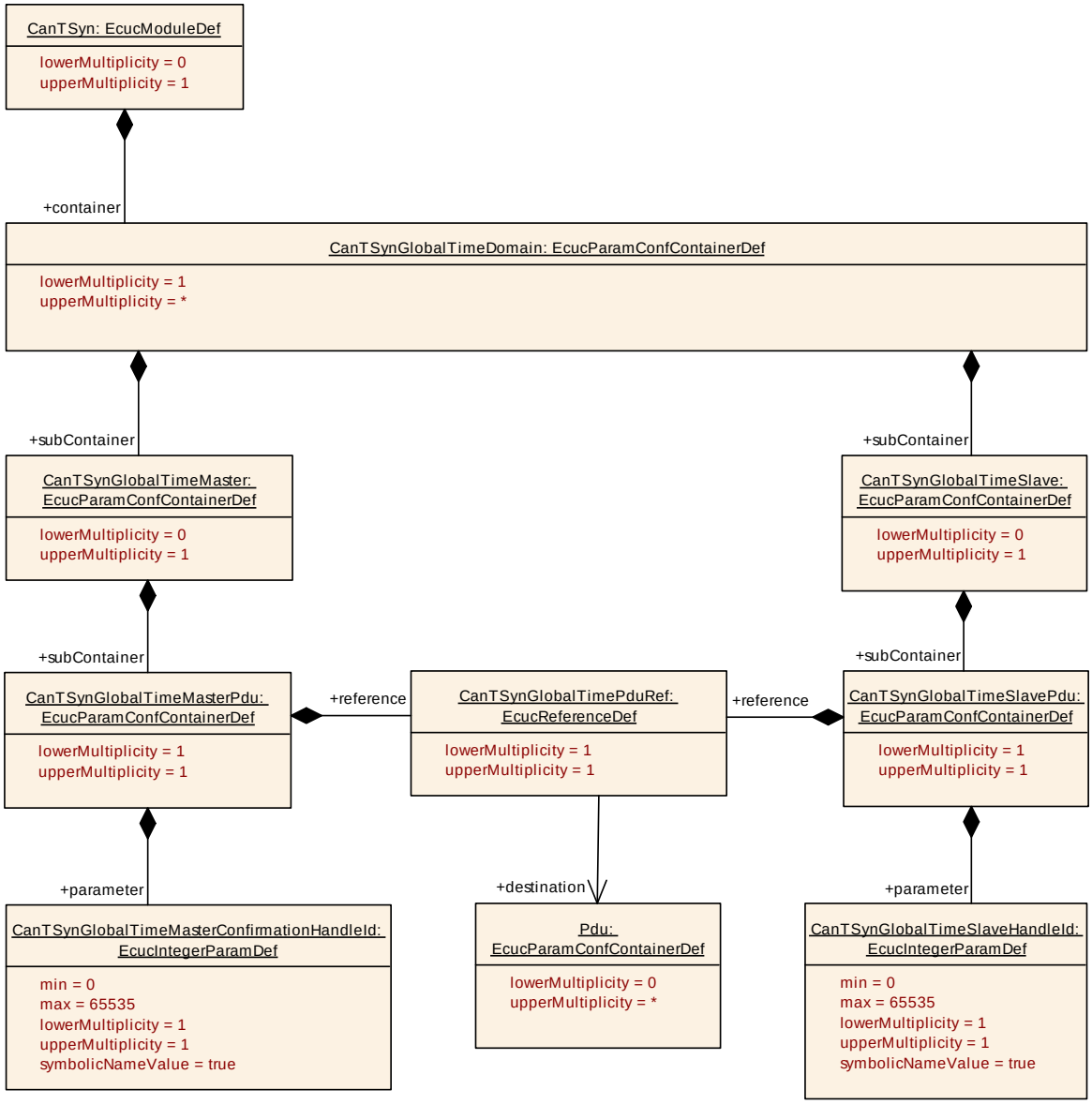
10.2.14 CanTSynGlobalTimeMasterPdu

SWS Item	**ECUC_CanTSyn_00009 :**
Container Name	CanTSynGlobalTimeMasterPdu
Description	This container encloses the configuration of the PDU that is supposed to contain the global time information.
Configuration Parameters	

SWS Item	**ECUC_CanTSyn_00008 :**		
Name	CanTSynGlobalTimeMasterConfirmationHandleId		
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	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_CanTSyn_00027 :**		
Name	CanTSynGlobalTimePduRef		
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



10.2.15 CanTSynGlobalTimeSlave

SWS Item	**ECUC_CanTSyn_00012 :**		
Container Name	CanTSynGlobalTimeSlave		
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			

SWS Item	**ECUC_CanTSyn_00006 :**
Name	CanTSynGlobalTimeFollowUpTimeout

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		

SWS Item	**ECUC_CanTSyn_00049 :**		
Name	CanTSynGlobalTimeMinMsgGap		
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		

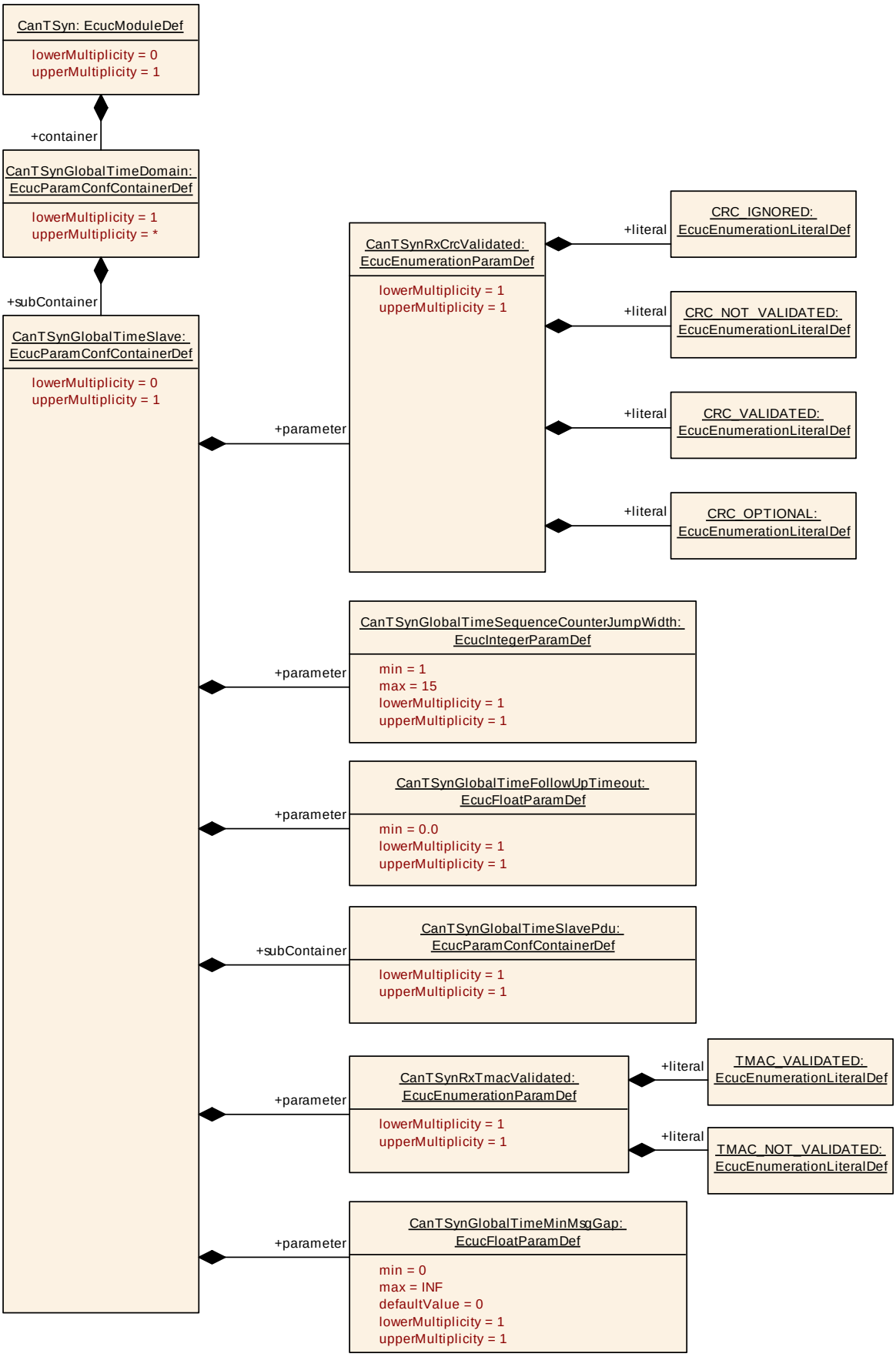
SWS Item	**ECUC_CanTSyn_00011 :**		
Name	CanTSynGlobalTimeSequenceCounterJumpWidth		
Parent Container	CanTSynGlobalTimeSlave		
Description	The SequenceCounterJumpWidth specifies the maximum allowed gap of the Sequence Counter between two SYNC resp. two OFS messages.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 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		

SWS Item	**ECUC_CanTSyn_00021 :**	
Name	CanTSynRxCrcValidated	
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		

SWS Item	**ECUC_CanTSyn_00048 :**		
Name	CanTSynRxTmacValidated		
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.


CanTSyn: EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
CanTSynGlobalTimeDomain: EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+subContainer
CanTSynGlobalTimeSlave: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
CanTSynRxCrcValidated: EcucEnumerationParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
+literal
CRC_IGNORED: EcucEnumerationLiteralDef
+literal
CRC_NOT_VALIDATED: EcucEnumerationLiteralDef
+literal
CRC_VALIDATED: EcucEnumerationLiteralDef
+literal
CRC_OPTIONAL: EcucEnumerationLiteralDef
+parameter
CanTSynGlobalTimeSequenceCounterJumpWidth: EcucIntegerParamDef
min = 1
max = 15
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
CanTSynGlobalTimeFollowUpTimeout: EcucFloatParamDef
min = 0.0
lowerMultiplicity = 1
upperMultiplicity = 1
+subContainer
CanTSynGlobalTimeSlavePdu: EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
CanTSynRxTmacValidated: EcucEnumerationParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
+literal
TMAC_VALIDATED: EcucEnumerationLiteralDef
+literal
TMAC_NOT_VALIDATED: EcucEnumerationLiteralDef
+parameter
CanTSynGlobalTimeMinMsgGap: EcucFloatParamDef
min = 0
max = INF
defaultValue = 0
lowerMultiplicity = 1
upperMultiplicity = 1

10.2.16 CanTSynGlobalTimeSlavePdu

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

SWS Item	**ECUC_CanTSyn_00013 :**		
Name	CanTSynGlobalTimeSlaveHandleId		
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	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_CanTSyn_00040 :**		
Name	CanTSynGlobalTimePduRef		
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 *SWS_BSWGeneral*.