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

Document Status	Final
Part of AUTOSAR Release	4.2.2

Document Change History		
Release	**Changed by**	**Change Description**
4.2.2	AUTOSAR Release Management	• CanTSyn_SetTransmissionMode changed to return "void" • minor corrections / clarifications / editorial changes
4.2.1	AUTOSAR Release Management	• Initial Release

Disclaimer

Advice for users

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

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

Table of Contents

1 Introduction and functional overview 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 12

7 Functional specification 15

7.1 Overview 15
7.2 Module Handling 15
7.2.1 Initialization 15
7.3 Message Format 16
7.3.1 SYNC and FUP Message 17
7.3.2 Offset Message 18
7.4 Acting as Time Master 20
7.4.1 SYNC and FUP message processing 21
7.4.2 OFS message processing 22
7.4.3 Transmission mode 24
7.4.4 Calculation and Assembling of Time Synchronization Messages on CAN 25
7.5 Acting as Time Slave 28
7.5.1 SYNC and FUP message processing 28
7.5.2 OFS message processing 29
7.5.3 Validation and Disassembling of Time Synchronization Messages on CAN 31
7.6 Error Classification 34
7.7 Error Detection 34
7.8 Error Notification 34

8 API specification 35

8.1 API 35
8.1.1 Imported types 35
8.1.2 Type definitions 35
8.1.3 Function definitions 36
8.1.4 Call-back notifications 37

8.1.5 Scheduled functions 39
8.1.6 Expected Interfaces 41
9 Sequence diagrams 42
9.1 StbM_GetCurrentTime <Master CAN SYNC/FUP> 42
9.2 StbM_BusSetGlobalTime <Slave CAN SYNC/FUP> 43
10 Configuration specification 44
10.1 How to read this chapter 44
10.2 Containers and configuration parameters 45
10.2.1 Variants 45
10.2.2 CanTSyn 45
10.2.3 CanTSynGeneral 47
10.2.4 CanTSynGlobalTimeSyncDataIDList 48
10.2.5 CanTSynGlobalTimeFupDataIDList 49
10.2.6 CanTSynGlobalTimeFupDataIDListElement 51
10.2.7 CanTSynGlobalTimeOfsDataIDList 51
10.2.8 CanTSynGlobalTimeOfsDataIDListElement 52
10.2.9 CanTSynGlobalTimeOfnsDataIDList 53
10.2.10 CanTSynGlobalTimeOfnsDataIDListElement 54
10.2.11 CanTSynGlobalTimeSlavePdu 55
10.2.12 CanTSynGlobalTimeDomain 56
10.2.13 CanTSynGlobalTimeMaster 58
10.2.14 CanTSynGlobalTimeMasterPdu 60
10.2.15 CanTSynGlobalTimeSlave 62
10.3 Published Information 64

1 Introduction and functional overview

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

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 to 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.

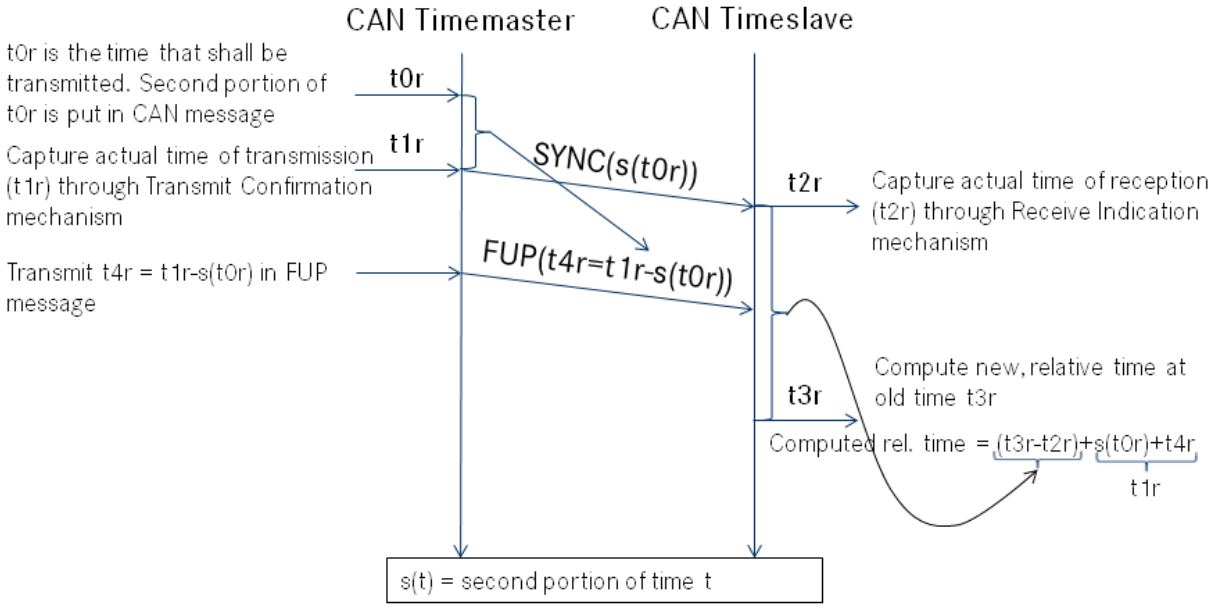


Figure 1: CAN time synchronization mechanism

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 [3].

Abbreviation / Acronym:	Description
(G)TD	(Global) Time Domain
(G)TM	(Global)Time Master
<Bus>TSyn	A bus specific Time Synchronization Provider module
CAN	Controller Area Network
CanTSyn	Time Synchronization Provider module for CAN
CRC	Cyclic Redundancy Checksum
DEM	Diagnostic Event Manager
DET	Development Error Tracer
FUP message	Time adjustment message (Follow-Up)
OFNS message	Offset adjustment message
OFS message	Offset synchronization message
StbM	Synchronized Time-Base Manager
SYNC message	Time synchronization message
TG	Time Gateway
TS	Time Slave
TSD	Time Sub-domain

Definitions	Description
Debounce Time	Minimum gap between two Tx messages with the same PCI.

3 Related documentation

3.1 Input documents

[1] AUTOSAR Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[2] AUTOSAR General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

[3] AUTOSAR Synchronized Time Base Manager
AUTOSAR_SWS_SynchronizedTimeBaseManager.pdf

[4] AUTOSAR Specification of CRC Routines
AUTOSAR_SWS_CRCLibrary.pdf

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software (SWS BSW General [2]) 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 timestamp capabilities. The first consequence is that the time synchronization is less accurate due to Rx-/Tx-ISR latencies and execution time until the current time is retrieved.
The second consequence is the need of interrupts in the CAN driver for the global time PDUs.

The time base in the SYNC message is limited to 32 bit, wherefore the maximum supported time value is 4294967295 seconds (2^{32}-1).

Time Master and Time Slaves shall work with a Time Base reference clock with a worst-case accuracy of 10µs.

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 Development Error Tracer (DET).

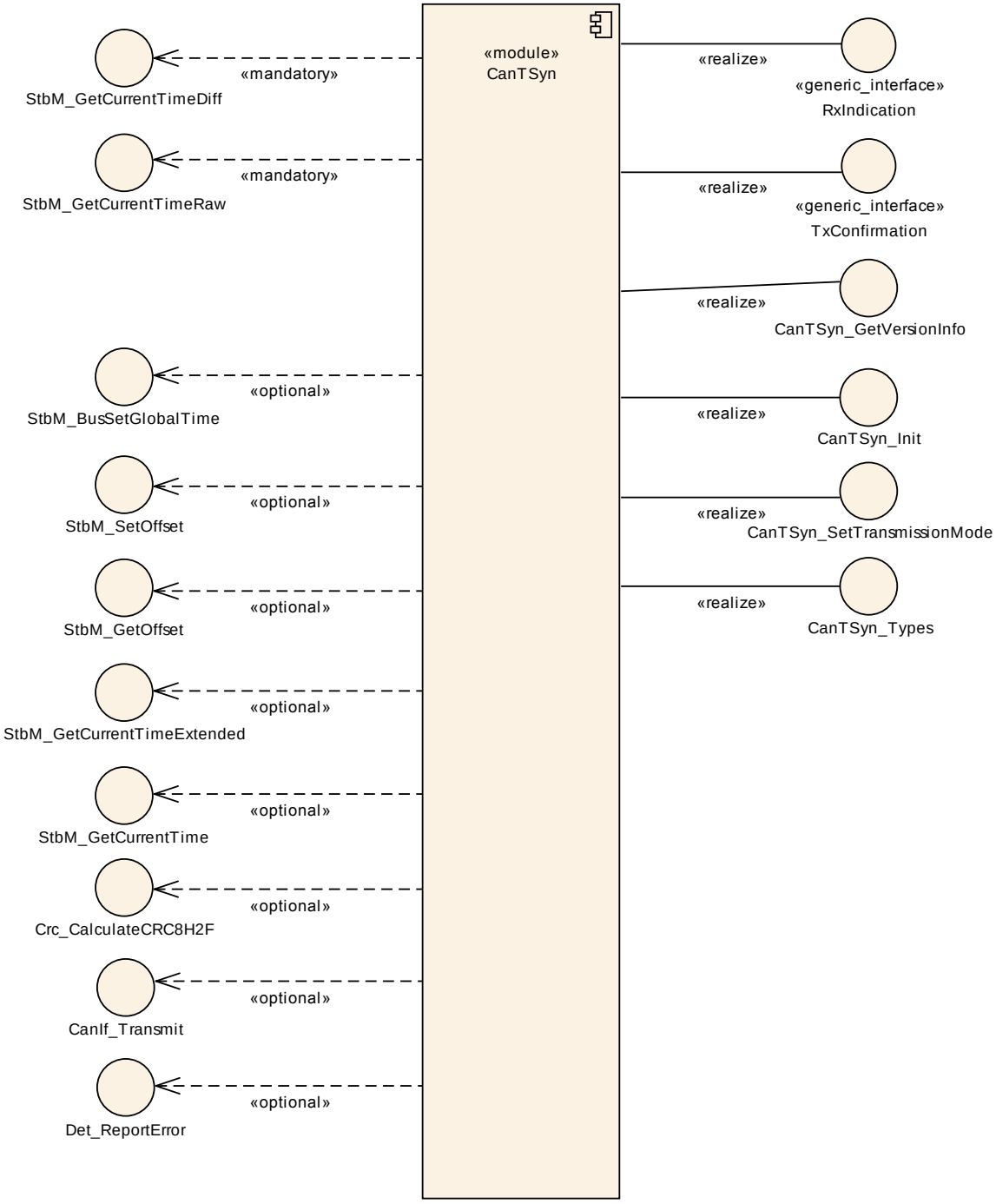


Figure 2: Module dependencies of the CanTSyn module

- 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 [2].

5.1.2 Header file structure

Besides the files defined in section 5.1.7 “Header file structure” of the SWS BSW General [2], the Time Synchronization over CAN needs to include the files defined below.

[SWS_CanTSyn_00002]
[The implementation header files shall include *`ComStack_Types.h`*.
](SRS_BSW_00301, SRS_BSW_00456)

The following picture shows the include hierarchy of the Time Synchronization over CAN.

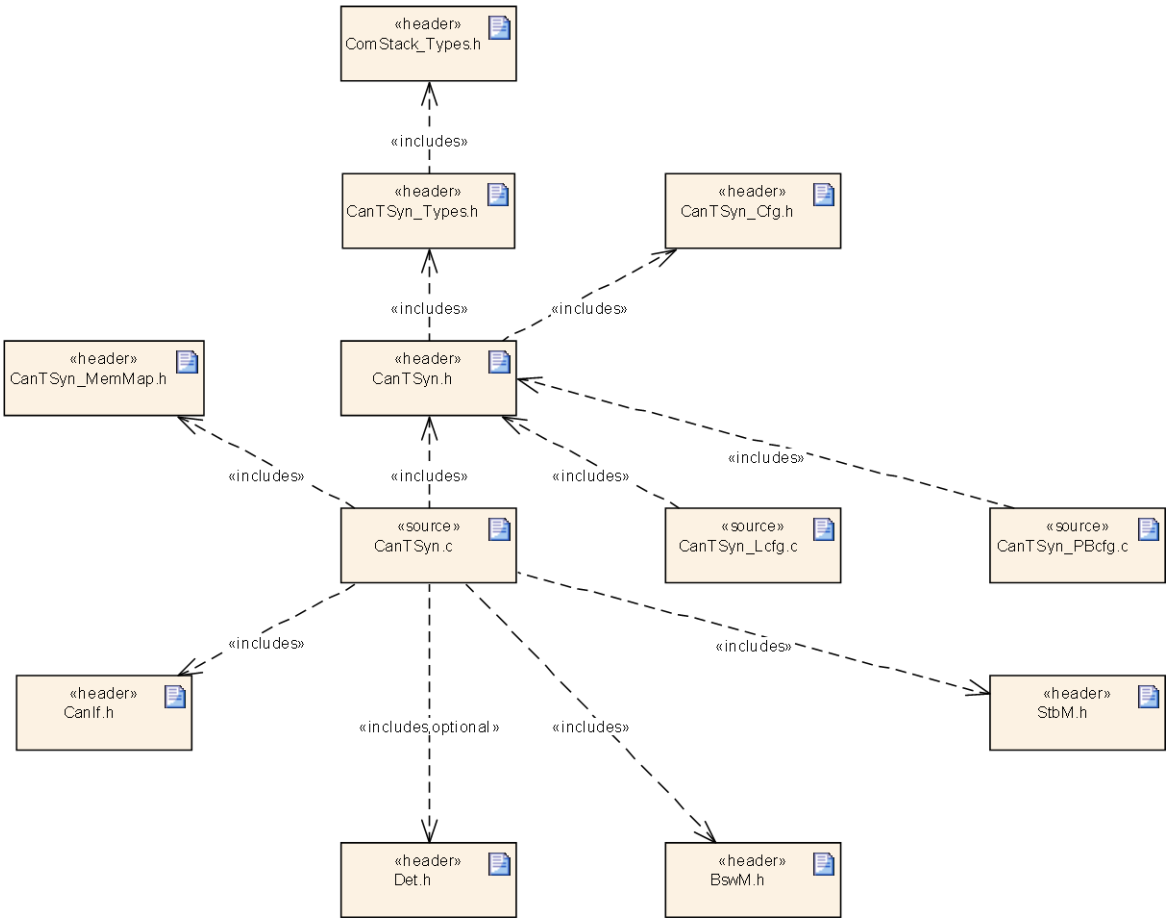


Figure 3: File structure of CanTSyn

6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00301	All AUTOSAR Basic Software Modules shall only import the necessary information	SWS_CanTSyn_00002
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_CanTSyn_00088, SWS_CanTSyn_00097, SWS_CanTSyn_00100
SRS_BSW_00337	Classification of development errors	SWS_CanTSyn_00005, SWS_CanTSyn_00097, SWS_CanTSyn_00100, SWS_CanTSyn_00104
SRS_BSW_00385	List possible error notifications	SWS_CanTSyn_00089
SRS_BSW_00396	The Basic Software Module specifications shall specify the supported configuration classes for changing values and multiplicities for each parameter/container	SWS_CanTSyn_00107
SRS_BSW_00456	- A Header file shall be defined in order to harmonize BSW Modules	SWS_CanTSyn_00002
SRS_StbM_20017	The StbM shall initialize the local time base if configured as Time Slave	SWS_CanTSyn_00003, SWS_CanTSyn_00006
SRS_StbM_20018	The StbM shall initialize the local time base with 0 at startup if configured as Time Slave	SWS_CanTSyn_00003, SWS_CanTSyn_00006
SRS_StbM_20019	The StbM shall initialize the global time base with a configurable startup value if configured as Time Master	SWS_CanTSyn_00003, SWS_CanTSyn_00006
SRS_StbM_20031	The CAN TimeSync module shall trigger time base synchronization transmission	SWS_CanTSyn_00024, 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
SRS_StbM_20032	The CAN TimeSync Module shall provide a time base after reception of a valid protocol information	SWS_CanTSyn_00064, SWS_CanTSyn_00072
SRS_StbM_20033	The CAN TimeSync 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_00019, SWS_CanTSyn_00020, SWS_CanTSyn_00021, SWS_CanTSyn_00022, SWS_CanTSyn_00031, SWS_CanTSyn_00041, SWS_CanTSyn_00048, SWS_CanTSyn_00049, SWS_CanTSyn_00050, SWS_CanTSyn_00051, SWS_CanTSyn_00052, SWS_CanTSyn_00053, SWS_CanTSyn_00054, SWS_CanTSyn_00055

SRS_StbM_20034	The CAN TimeSync Module shall detect and handle timeout and integrity errors in the time synchronization protocol	SWS_CanTSyn_00017, SWS_CanTSyn_00018, SWS_CanTSyn_00021, SWS_CanTSyn_00022, 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_00081, SWS_CanTSyn_00082, SWS_CanTSyn_00083, SWS_CanTSyn_00084, SWS_CanTSyn_00085, SWS_CanTSyn_00087, SWS_CanTSyn_00088
SRS_StbM_20035	The CAN TimeSync module shall support a protocol for precise time measurement and synchronization over CAN	SWS_CanTSyn_00008, SWS_CanTSyn_00009, SWS_CanTSyn_00010, SWS_CanTSyn_00011, SWS_CanTSyn_00015, SWS_CanTSyn_00016, SWS_CanTSyn_00017, SWS_CanTSyn_00018, SWS_CanTSyn_00019, SWS_CanTSyn_00020, SWS_CanTSyn_00021, SWS_CanTSyn_00022, SWS_CanTSyn_00024, 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_00034, 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_00045, SWS_CanTSyn_00046, SWS_CanTSyn_00047, SWS_CanTSyn_00048, SWS_CanTSyn_00049, SWS_CanTSyn_00050, SWS_CanTSyn_00051, SWS_CanTSyn_00052, SWS_CanTSyn_00053, 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_00064, SWS_CanTSyn_00065, SWS_CanTSyn_00066, SWS_CanTSyn_00067, SWS_CanTSyn_00068, SWS_CanTSyn_00069, SWS_CanTSyn_00070, SWS_CanTSyn_00071, SWS_CanTSyn_00072, SWS_CanTSyn_00073, SWS_CanTSyn_00074, SWS_CanTSyn_00075, SWS_CanTSyn_00076, SWS_CanTSyn_00077, SWS_CanTSyn_00078, SWS_CanTSyn_00079, SWS_CanTSyn_00080, SWS_CanTSyn_00081, SWS_CanTSyn_00082, SWS_CanTSyn_00083, 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_00098, SWS_CanTSyn_00099, SWS_CanTSyn_00101, SWS_CanTSyn_00102, SWS_CanTSyn_00103,

		SWS_CanTSyn_00105, SWS_CanTSyn_00106
SRS_StbM_20036	The CAN TimeSync module shall use the time measurement and synchronization protocol to transmit and receive an offset value	SWS_CanTSyn_00019, SWS_CanTSyn_00020, SWS_CanTSyn_00021, SWS_CanTSyn_00022, SWS_CanTSyn_00034, 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_00046, SWS_CanTSyn_00065, SWS_CanTSyn_00066, SWS_CanTSyn_00067, SWS_CanTSyn_00068, SWS_CanTSyn_00069, SWS_CanTSyn_00070, SWS_CanTSyn_00071, SWS_CanTSyn_00072, SWS_CanTSyn_00074, SWS_CanTSyn_00077, SWS_CanTSyn_00079
SRS_StbM_20037	The CAN TimeSync module shall support user specific data within the time measurement and synchronization protocol	SWS_CanTSyn_00009, SWS_CanTSyn_00011, SWS_CanTSyn_00012, SWS_CanTSyn_00013, SWS_CanTSyn_00014
SRS_StbM_20038	The CAN TimeSync configuration shall allow the CanTSyn to support different roles for a time base	SWS_CanTSyn_00108

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 [3].

7.2 Module Handling

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

7.2.1 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.
⌋(SRS_StbM_20017, SRS_StbM_20018, SRS_StbM_20019)

[SWS_CanTSyn_00005]
⌈When DET reporting is enabled (see `CanTSynDevErrorDetect`), the Time Synchronization over CAN shall call `Det_ReportError()` with the error code `CANTSYN_E_NOT_INITIALIZED` when any API other than `CanTSyn_GetVersionInfo()` or `CanTSyn_Init()` is called in uninitialized state.
⌋(SRS_BSW_00337)

[SWS_CanTSyn_00006]
⌈When `CanTSyn_Init()` is called in initialized state, the Time Synchronization over CAN shall re-initialize its internal variables.
⌋(SRS_StbM_20017, SRS_StbM_20018, SRS_StbM_20019)

[SWS_CanTSyn_00007]
⌈Sequence Counter (*SC*) shall be initialized with 0.
⌋(SRS_StbM_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 share the same CAN ID by using a multiplexed signal group. The Multiplexer is located at Byte 0, named as “*Type*”.

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

In case of CAN FD an adaptation of the message format to harmonize the Time Base resolution between CAN / FlexRay and Ethernet might be possible.

[SWS_CanTSyn_00008]
⌈The byte order for time synchronization message time stamp signals is “Big Endian”.
⌋(SRS_StbM_20035)

[SWS_CanTSyn_00009]
⌈The byte order for time synchronization message User Data bytes is “Opaque”.
⌋(SRS_StbM_20035, SRS_StbM_20037)

[SWS_CanTSyn_00010]
⌈The DLC of SYNC, FUP, OFS and OFNS messages is 8.
⌋(SRS_StbM_20035)

[SWS_CanTSyn_00011]
⌈Each time synchronization message contains User Data according to the given message format.⌋(SRS_StbM_20035, SRS_StbM_20037)

[SWS_CanTSyn_00012]
⌈User Data shall be read from the current incoming message consistently.
⌋(SRS_StbM_20037)

[SWS_CanTSyn_00013]
⌈User Data shall be written to the next outgoing message consistently.
⌋(SRS_StbM_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.
⌋(SRS_StbM_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
⌋(SRS_StbM_20033, SRS_StbM_20035)

[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 stamp in nanoseconds
⌋(SRS_StbM_20033, SRS_StbM_20035)

[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
⌋(SRS_StbM_20033, SRS_StbM_20034, SRS_StbM_20035)

[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 stamp in nanoseconds
⌋(SRS_StbM_20033, SRS_StbM_20034, SRS_StbM_20035)

7.3.2 Offset Message

Offset messages are multiplexed with the SYNC message (using the same PDU, etc.).

[SWS_CanTSyn_00019]
⌈OFS not CRC secured message format:
Byte 0: *Type* = 0x30
Byte 1: reserved, 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: *OfsTimeSecLsbHi* = 8 Bit offset time stamp (LSB) from secondsHi
Byte 4-7: *OfsTimeSecLsbLo* = 32 Bit offset time stamp (LSB) from secondsLo
⌋(SRS_StbM_20033, SRS_StbM_20035, SRS_StbM_20036)

[SWS_CanTSyn_00020]
⌈OFNS not CRC secured message format:
Byte 0: *Type* = 0x38
Byte 1: reserved, 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: *OfsTimeSecMsbHi* = 8 Bit offset time stamp (MSB) from secondsHi
Byte 4-7: *OfsTimeNSec* = 32 Bit offset time stamp in nanoseconds
⌋(SRS_StbM_20033, SRS_StbM_20035, SRS_StbM_20036)

[SWS_CanTSyn_00021]
⌈OFS CRC secured message format:
Byte 0: *Type* = 0x40
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: *OfsTimeSecLsbHi* = 8 Bit offset time stamp (LSB) from secondsHi
Byte 4-7: *OfsTimeSecLsbLo* = 32 Bit offset time stamp (LSB) from secondsLo
⌋(SRS_StbM_20033,SRS_StbM_20034,SRS_StbM_20035, SRS_StbM_20036)

[SWS_CanTSyn_00022]
⌈OFNS CRC secured message format:
Byte 0: *Type* = 0x48
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: *OfsTimeSecMsb* = 8 Bit offset time stamp (MSB) from secondsHi
Byte 4-7: *OfsTimeNSec* = 32 Bit offset time stamp in nanoseconds
⌋(SRS_StbM_20033, SRS_StbM_20034, SRS_StbM_20035, SRS_StbM_20036)

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 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.

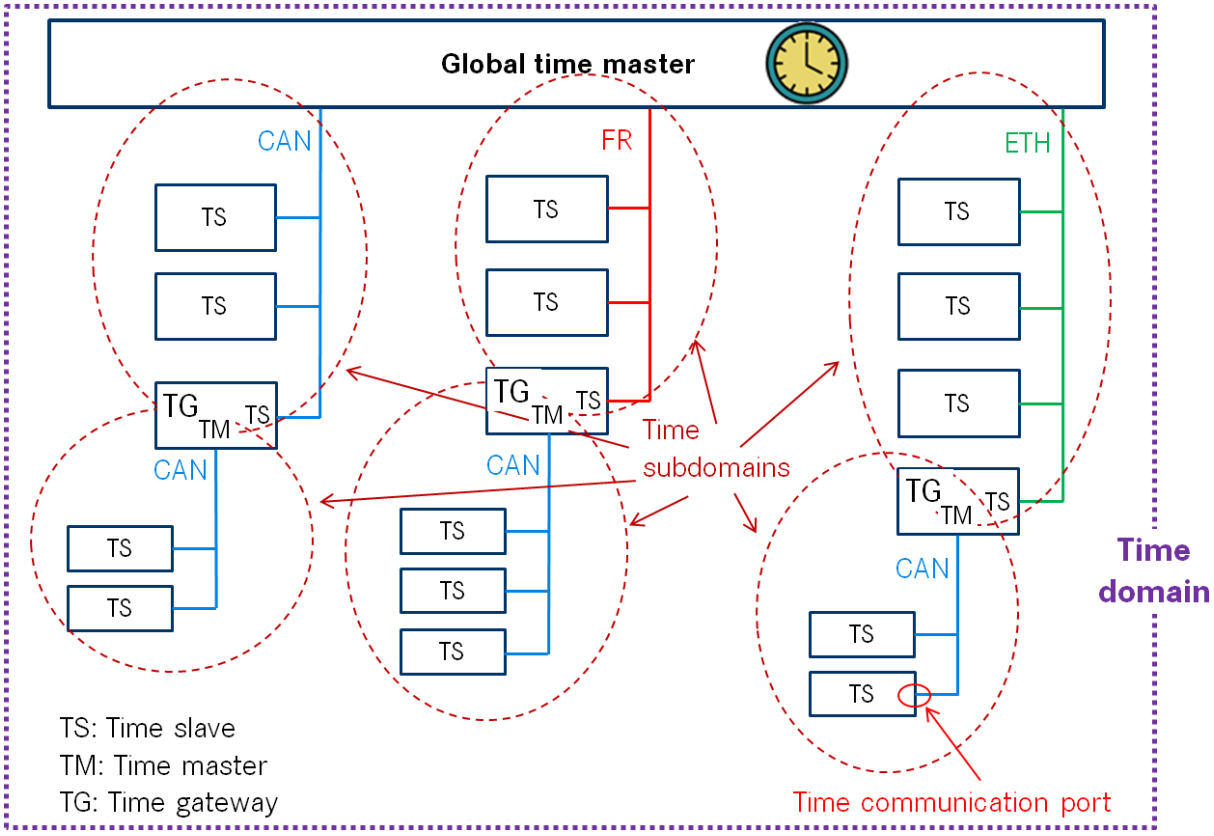


Figure 4: Terminology Example

7.4.1 SYNC and FUP message processing

[SWS_CanTSyn_00024]
⌈A synchronization message sequence consists of a SYNC message followed by a FUP message per Time Domain.
⌋(SRS_StbM_20031, SRS_StbM_20035)

[SWS_CanTSyn_00025]
⌈A transmitter of SYNC messages (Time Master) shall start each time synchronization sequence with a SYNC message.
⌋(SRS_StbM_20031, SRS_StbM_20035)

[SWS_CanTSyn_00026]
⌈A transmitter of FUP messages (Time Master) shall finish each time synchronization sequence with a FUP message.
⌋(SRS_StbM_20031, SRS_StbM_20035)

[SWS_CanTSyn_00027]
⌈Any Timeout while waiting for `CanTSyn_TxConfirmation()` function resets the state machine to start with a new SYNC transmission again.
⌋(SRS_StbM_20034, SRS_StbM_20035)

Note: The timeout value shall be always shorter than the time between SYNC to FUP **and** FUP to SYNC (smallest value compared to both).

[SWS_CanTSyn_00028]
⌈A transmitter of SYNC messages (Time Master with Time Base Id 0..15) is using a cyclic transmission of SYNC messages (according Figure 5: Master CAN SYNC/FUP) with `CanTSynGlobalTimeTxPeriod` (**ECUC_CanTSyn_00017 :**) if the `GLOBAL_TIME_BASE` bit within the `timeBaseStatus` is set.
⌋(SRS_StbM_20031, SRS_StbM_20035)

Note: The SYNC cycle time is always bigger or equal than the system debounce time of this PCI.

[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. Further a minimum time delay of the regular FUP shall be waited before starting the SYNC / FUP sequence of the same Time Domain.
⌋(SRS_StbM_20035)

[SWS_CanTSyn_00031]
⌈Depending on `CanTSynGlobalTimeTxCrcSecured` 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 secured message	`0x28` FUP CRC secured message

⌋(SRS_StbM_20033, SRS_StbM_20035)

[SWS_CanTSyn_00032]
⌈A transmitter of FUP messages (Time Master) is using as trigger condition for SYNC to FUP a trigger offset configuration value of `CanTSynGlobalTimeTxFollowUpOffset` (**ECUC_CanTSyn_00016 :**).
⌋(SRS_StbM_20031, SRS_StbM_20035)

Note: The FUP trigger offset value is always bigger or equal than the system debounce time of this PCI.

[SWS_CanTSyn_00033]
⌈Each transmission request of SYNC messages shall be observed with the transmit confirmation timeout `CanTSynMasterConfirmationTimeout` (**ECUC_CanTSyn_00020 :**). If the timeout occurs, the transmission request shall be revoked and no FUP message shall be sent.
⌋(SRS_StbM_20034, SRS_StbM_20035)

7.4.2 OFS message processing

[SWS_CanTSyn_00034]
⌈An offset message sequence consists of an OFS message followed by an OFNS message per Time Domain.
⌋(SRS_StbM_20035, SRS_StbM_20036)

[SWS_CanTSyn_00035]
⌈A transmitter of OFS messages (Time Master) shall start each time offset synchronization sequence with an OFS message.
⌋(SRS_StbM_20031, SRS_StbM_20035, SRS_StbM_20036)

[SWS_CanTSyn_00036]
⌈A transmitter of OFNS messages (Time Master) shall finish each time offset synchronization sequence with an OFNS message.
⌋(SRS_StbM_20031, SRS_StbM_20035, SRS_StbM_20036)

[SWS_CanTSyn_00037]
⌈Any Timeout while waiting for `CanTSyn_TxConfirmation()` function resets the state machine to start with a new OFS transmission again.
⌋(SRS_StbM_20034, SRS_StbM_20035, SRS_StbM_20036)

Note: The timeout value shall be always shorter than the time between OFS to OFNS **and** OFNS to OFS (smallest value compared to both).

[SWS_CanTSyn_00038]
⌈A transmitter of OFS messages (Time Master with Time Domain Id 16..31) is using a cyclic transmission of OFS messages (`CanTSynGlobalTimeTxPeriod` (refer **ECUC_CanTSyn_00017 :**) if the `GLOBAL_TIME_BASE` bit within the `timeBaseStatus` is set.
⌋(SRS_StbM_20031, SRS_StbM_20035, SRS_StbM_20036)

Note: The OFS cycle time is always bigger or equal than the system debounce time of this PCI.

[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. Further a minimum time delay of the regular OFNS shall be waited before starting the OFS / OFNS sequence of the same Time Domain.
⌋(SRS_StbM_20035, SRS_StbM_20036)

[SWS_CanTSyn_00040]
⌈A transmitter of OFNS messages (Time Master) is using as trigger condition for OFS to OFNS a trigger offset configuration value of `CanTSynGlobalTimeTxFollowUpOffset` (**ECUC_CanTSyn_00016 :**).
⌋(SRS_StbM_20035, SRS_StbM_20036)

Note: The OFNS trigger offset value is always bigger or equal than the system debounce time of this PCI.

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

`CanTSynGlobalTimeTxCrcSecured`	OFS	OFNS
`CRC_NOT_SUPPORTED`	`0x30` OFS not CRC secured message format	`0x38` OFNS not CRC secured message
`CRC_SUPPORTED`	`0x40` OFS CRC secured message	`0x48` OFNS CRC secured message

⌋(SRS_StbM_20033, SRS_StbM_20035, SRS_StbM_20036)

[SWS_CanTSyn_00042]
⌈Each OFS transmission request shall be observed with the transmit confirmation timeout `CanTSynMasterConfirmationTimeout` (**ECUC_CanTSyn_00020 :**). If the timeout occurs, the transmission request shall be revoked and no OFNS message shall be sent.
⌋(SRS_StbM_20034, SRS_StbM_20035, SRS_StbM_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.
⌋(SRS_StbM_20031, SRS_StbM_20035)

[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.
⌋(SRS_StbM_20031, SRS_StbM_20035)

7.4.4 Calculation and Assembling of Time Synchronization Messages on CAN

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).

7.4.4.1 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:

1. On transmission of SYNC message
   a. Get Synchronized Time Base T0 via `StbM_GetCurrentTime()` and write second portion of T0 to *SyncTimeSec*
   b. Get raw time $T0_{raw}$ for time measurement of transmission delay via `StbM_GetCurrentTimeRaw()`

2. On SYNC message TX confirmation
   a. Retrieve time difference $T0_{diff}$ (calculated with $T0_{raw}$) of the transmission delay via `StbM_GetCurrentTimeDiff()`
   b. Calculate T4 for FUP message as T4 = ($T0_{ns}$ + $T0_{diff}$) with $T0_{ns}$ as nanosecond portion of T0

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

](SRS_StbM_20035)

With these steps, the Synchronized Time Base at the transmitter side has been calculated (T0 + T4).

[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. Get second portion of the Offset Time Base via `StbM_GetOffset()` and write to *OfsTimeSecLsbLo, OfsTimeSecLsbHi, OfsTimeSecMsb*
2. Use nanosecond portion of the Offset Time and write to *OfsTimeNSec*

](SRS_StbM_20035, SRS_StbM_20036)

Note: The Offset Time Base has not been time stamped.

7.4.4.2 OVS Calculation

[SWS_CanTSyn_00047]
⌈*OVS* has to 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` will be written to *OVS*.
⌋(SRS_StbM_20035)

7.4.4.3 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*.
⌋(SRS_StbM_20035)

7.4.4.4 Sequence Counter Calculation

[SWS_CanTSyn_00048]
⌈A Sequence Counter (*SC*) of 4bit is representing numbers from 0 to 15 per Time Domain. The Sequence Counter is independent between SYNC and OFS messages and shall be incremented by 1 continuously on every transmission request of a SYNC or OFS message. It wraps around at 15 to 0 again.
⌋(SRS_StbM_20033, SRS_StbM_20035)

[SWS_CanTSyn_00049]
⌈Sequence Counter (*SC*) remains unchanged on every transmission request of a FUP or OFNS message.
⌋(SRS_StbM_20033, SRS_StbM_20035)

7.4.4.5 CRC Calculation

[SWS_CanTSyn_00050]
⌈The function `Crc_CalculateCRC8H2F()` as defined in [4] shall be used to calculate the *CRC* if configured.
⌋(SRS_StbM_20033, SRS_StbM_20035)

[SWS_CanTSyn_00051]
⌈The *CRC* start value shall be 0xFF.
⌋(SRS_StbM_20033, SRS_StbM_20035)

[SWS_CanTSyn_00052]
⌈The *CRC* final XOR-value shall be 0xFF.
⌋(SRS_StbM_20033, SRS_StbM_20035)

[SWS_CanTSyn_00053]
⌈The *CRC* polynomial shall be 0x2F.
⌋(SRS_StbM_20033, SRS_StbM_20035)

[SWS_CanTSyn_00054]
⌈The `DataID` will be calculated as `DataID = DataIDList[`*SC*`]`, whereat `DataIDList` is given by configuration for each message *Type*.
⌋(SRS_StbM_20033, SRS_StbM_20035)

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

[SWS_CanTSyn_00055]
⌈The *CRC* will be calculated over time synchronization message *Byte 2* to *Byte 7* and `DataID`.
⌋(SRS_StbM_20033, SRS_StbM_20035)

7.4.4.6 Message Assembling

[SWS_CanTSyn_00056]
⌈For each transmission of a time synchronization message the CanTSyn module shall assemble the message like follows:

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

⌋(SRS_StbM_20035)

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 SYNC message with *Type* equal to `0x20` and a correct CRC value if `CanTSynRxCrcValidated` is configured to `CRC_VALIDATED`. All other messages shall be ignored.
⌋(SRS_StbM_20034, SRS_StbM_20035)

[SWS_CanTSyn_00058]
⌈The CanTSyn shall only accept SYNC message with *Type* equal to `0x10` if `CanTSynRxCrcValidated` is configured to `CRC_NOT_VALIDATED`. All other messages shall be ignored.
⌋(SRS_StbM_20035)

[SWS_CanTSyn_00059]
⌈The CanTSyn shall only accept SYNC message with *Type* equal to `0x10` or `0x20` if `CanTSynRxCrcValidated` is configured to `CRC_IGNORED`. All other messages shall be ignored.
⌋(SRS_StbM_20035)

[SWS_CanTSyn_00060]
⌈The CanTSyn shall only accept 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`. All other messages shall be ignored.
⌋(SRS_StbM_20034, SRS_StbM_20035)

[SWS_CanTSyn_00061]
⌈The CanTSyn shall only accept 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`. All other messages shall be ignored.
⌋(SRS_StbM_20034, SRS_StbM_20035)

[SWS_CanTSyn_00062]
⌈The CanTSyn shall only accept 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`. All other messages shall be ignored.
⌋(SRS_StbM_20034, SRS_StbM_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).
](SRS_StbM_20034, SRS_StbM_20035)

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

[SWS_CanTSyn_00064]
[For valid FUP messages a new Global Time value shall be calculated and forwarded to the StbM module via `StbM_BusSetGlobalTime()` (according Figure 6: Slave CAN SYNC/FUP).
](SRS_StbM_20032, SRS_StbM_20034, SRS_StbM_20035)

7.5.2 OFS message processing

[SWS_CanTSyn_00065]
[The CanTSyn shall only accept OFS message with *Type* equal to `0x40` and a correct *CRC* value if `CanTSynRxCrcValidated` is configured to `CRC_VALIDATED`. All other messages shall be ignored.
](SRS_StbM_20034, SRS_StbM_20035, SRS_StbM_20036)

[SWS_CanTSyn_00066]
[The CanTSyn shall only accept OFS message with *Type* equal to `0x30` if `CanTSynRxCrcValidated` is configured to `CRC_NOT_VALIDATED`. All other messages shall be ignored.
](SRS_StbM_20035, SRS_StbM_20036)

[SWS_CanTSyn_00067]
[The CanTSyn shall only accept OFS message with *Type* equal to `0x30` or `0x40` if `CanTSynRxCrcValidated` is configured to `CRC_IGNORED`. All other messages shall be ignored.
](SRS_StbM_20035, SRS_StbM_20036)

[SWS_CanTSyn_00068]
[The CanTSyn shall only accept OFNS message with an identical Sequence Counter to the value of the corresponding OFS message and *Type* equal to `0x48` and a correct *CRC* value if `CanTSynRxCrcValidated` is configured to `CRC_VALIDATED`. All other messages shall be ignored.
](SRS_StbM_20034, SRS_StbM_20035, SRS_StbM_20036)

[SWS_CanTSyn_00069]
⌈The CanTSyn shall only accept OFNS message with an identical Sequence Counter to the value of the corresponding OFS message and *Type* equal to `0x38` if `CanTSynRxCrcValidated` is configured to `CRC_NOT_VALIDATED`. All other messages shall be ignored.
⌋(SRS_StbM_20035, SRS_StbM_20036)

[SWS_CanTSyn_00070]
⌈The CanTSyn shall only accept OFNS message with an identical Sequence Counter to the value of the corresponding OFS message and *Type* equal to `0x38` or `0x48` if `CanTSynRxCrcValidated` is configured to `CRC_IGNORED`. All other messages shall be ignored.
⌋(SRS_StbM_20035, SRS_StbM_20036)

[SWS_CanTSyn_00071]
⌈For each configured Time Slave (`CanTSynGlobalTimeSlave`) the CanTSyn module shall observe 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).
⌋(SRS_StbM_20034, SRS_StbM_20035, SRS_StbM_20036)

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

[SWS_CanTSyn_00072]
⌈For valid OFNS messages a new offset time value shall be calculated (according **[SWS_CanTSyn_00074]**) and forwarded to the StbM module via `StbM_SetOffset()`.
⌋(SRS_StbM_20032, SRS_StbM_20034, SRS_StbM_20035, SRS_StbM_20036)

7.5.3 Validation and Disassembling of Time Synchronization Messages on CAN

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 (Time Slave) shall perform the following steps to retrieve the Synchronized Time Base exactly:

1. On SYNC message RX indication, which delivers Synchronized Time Base part T0, retrieve of Local Time Stamp via `StbM_GetCurrentTimeRaw()` as $T2_{raw}$
2. On FUP message RX indication, which delivers Synchronized Time Base part T4 = (*OVS* + *SyncTimeNSec*), retrieve the time difference between internal during function call time stamped $T3_{raw}$ and time stamp of the previously received Synchronized Time Base $T2_{raw}$ via `StbM_GetCurrentTimeDiff()`, which delivers $T3_{diff} = (T3_{raw} - T2_{raw})$
3. Calculate Global Time Base to update the Time Slaves Local Time Base as: GlobalTimeBase = $(T3_{raw} - T2_{raw}) + (T0 + T4)$

⌋(SRS_StbM_20035)

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

1. Assemble second portion of the Offset Time Base out of *OfsTimeSecLsbLo, OfsTimeSecLsbHi, OfsTimeSecMsb*
2. Get nanosecond portion of the Offset Time out of *OfsTimeNSec*

⌋(SRS_StbM_20035, SRS_StbM_20036)

Note: The Offset Time Base has not been time stamped.

7.5.3.2 OVS Consideration

[SWS_CanTSyn_00075]
⌈*OVS* (FUP only) has to be considered on the receiver side to retrieve the second portion of the received Synchronized Time Base.
⌋(SRS_StbM_20035)

7.5.3.3 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.
⌋(SRS_StbM_20034, SRS_StbM_20035)

[SWS_CanTSyn_00077]
⌈The Sequence Counter of each OFS message must match to the Sequence Counter of the next incoming OFNS message of the same Time Domain. Otherwise, the contents of the already received OFS message shall be discarded and the received OFNS message shall be ignored.
⌋(SRS_StbM_20034, SRS_StbM_20035, SRS_StbM_20036)

[SWS_CanTSyn_00078]
⌈The Sequence Counter Jump Width between two SYNC resp. two OFS messages must be always smaller or equal to `CanTSynGlobalTimeSequenceCounterJumpWidth`. The value 0 is not allowed.
⌋(SRS_StbM_20034, SRS_StbM_20035)

[SWS_CanTSyn_00079]
⌈Due to an asynchronous startup of Time Slaves compared to the Time Master the 1st Sequence Counter of the 1st received SYNC and OFS message per Time Domain shall not be checked against the defined Sequence Counter Jump Width.
⌋(SRS_StbM_20034, SRS_StbM_20035, SRS_StbM_20036)

7.5.3.4 CRC Validation

[SWS_CanTSyn_00080]
⌈The function `Crc_CalculateCRC8H2F()` as defined in [4] shall be used to validate the *CRC* if configured.
⌋(SRS_StbM_20034, SRS_StbM_20035)

[SWS_CanTSyn_00081]
⌈The *CRC* start value shall be 0xFF.
⌋(SRS_StbM_20034, SRS_StbM_20035)

[SWS_CanTSyn_00082]
⌈The *CRC* final XOR-value shall be 0xFF.
⌋(SRS_StbM_20034, SRS_StbM_20035)

[SWS_CanTSyn_00083]
⌈The *CRC* polynomial shall be 0x2F.
⌋(SRS_StbM_20034, SRS_StbM_20035)

[SWS_CanTSyn_00084]
⌈The DataID shall be calculated as `DataID = DataIDList[`*SC*`]`, whereat `DataIDList` is given by configuration for each message *Type*.
⌋(SRS_StbM_20034, SRS_StbM_20035)

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

[SWS_CanTSyn_00085]
⌈The *CRC* shall be calculated over time synchronization message *Byte 2* to *Byte 7* and `DataID`.
⌋(SRS_StbM_20034, SRS_StbM_20035)

7.5.3.5 Message Disassembling

[SWS_CanTSyn_00086]
⌈For each received time synchronization message the CanTSyn module shall validate the message like follows (all conditions must match):

1. *Type* matches depending on the `CanTSynRxCrcValidated` parameter
2. *SC* matches to the expected value
3. *D* matches to the defined Time Domain range for each *Type*
4. *D* matches to one of the configured Time Domains
5. *SyncTimeNSec* (FUP / OFNS only) matches the defined range of `StbM_TimeStampType.nanoseconds`.
6. *CRC* (including `DataID`) matches depending on the `CanTSynRxCrcValidated` parameter

⌋(SRS_StbM_20035)

[SWS_CanTSyn_00087]
⌈For each received time synchronization message the CanTSyn module shall disassemble the message after successful validation **[SWS_CanTSyn_00086]**.
⌋(SRS_StbM_20034, SRS_StbM_20035)

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.
⌋(SRS_StbM_20034, SRS_BSW_00323)

[SWS_CanTSyn_00089]
⌈CanTSyn shall use following errors:

Type or error	*Relevance*	*Related error code*	*Value [hex]*
API service called with wrong PDU or SDU.	Development	CANTSYN_E_INVALID_PDUID	0x01
API service used in un-initialized state	Development	CANTSYN_E_NOT_INITIALIZED	0x02
A pointer is NULL	Development	CANTSYN_E_NULL_POINTER	0x03

⌋(SRS_BSW_00385)

7.7 Error Detection

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

7.8 Error Notification

The module ID 161 of CanTSyn, which is used as a parameter in the Det_ReportError() call, is exported via the macro definition CANTSYN_MODULE_ID in *CanTSyn.h*.

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	*Imported Type*
ComStack_Types	PduIdType
	PduInfoType
StbM	StbM_SynchronizedTimeBaseType
	StbM_TimeStampExtendedType
	StbM_TimeStampRawType
	StbM_TimeStampType
	StbM_UserDataType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋(SRS_StbM_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.		

⌋(SRS_StbM_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	

](SRS_StbM_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.	

](SRS_StbM_20035)

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.	

⌋(SRS_StbM_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.	

⌋(SRS_StbM_20035)

8.1.4 Call-back notifications

This is a list of functions provided for other modules. The function prototypes of the callback functions shall be provided in the file *CanTSyn_Cbk.h*.

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 I-PDU.
	PduInfoPtr	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Indication of a received I-PDU from a lower layer communication interface module.	

⌋(SRS_StbM_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 (`CANTSYN_DEV_ERROR_DETECT` is set to TRUE) and if function call has failed because of the following reasons:

- Invalid PDU ID (`CANTSYN_E_INVALID_PDUID`)
- CanTSyn was not initialized (`CANTSYN_E_NOT_INITIALIZED`)
- `PduInfoPtr` or `SduDataPtr` equals `NULL_PTR` (`CANTSYN_E_NULL_POINTER`)

⌋(SRS_BSW_00323, SRS_BSW_00337)

[SWS_CanTSyn_00098]
⌈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.
- The CanTSyn module is initialized correctly.

⌋(SRS_StbM_20035)

8.1.4.2 CanTSyn_TxConfirmation

[SWS_CanTSyn_00099]
⌈

Service name:	CanTSyn_TxConfirmation	
Syntax:	`void CanTSyn_TxConfirmation( PduIdType TxPduId )`	
Service ID[hex]:	0x40	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the I-PDU that has been transmitted.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	The lower layer communication interface module confirms the transmission of an I-PDU.	

⌋(SRS_StbM_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 (`CANTSYN_DEV_ERROR_DETECT` is set to `TRUE`) and if function call has failed because of the following reasons:

- Invalid PDU ID (`CANTSYN_E_INVALID_PDUID`)
- CanTSyn was not initialized (`CANTSYN_E_NOT_INITIALIZED`)

⌋(SRS_BSW_00323, SRS_BSW_00337)

[SWS_CanTSyn_00101]
⌈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.
- The CanTSyn module is initialized correctly.

⌋(SRS_StbM_20035)

8.1.5 Scheduled functions

These functions are directly called by 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. SYNC and FUP transmission

⌋(SRS_StbM_20035)

[SWS_CanTSyn_00103]
⌈The frequency of invocations of `CanTSyn_MainFunction()` is determined by the configuration parameter `CanTSynMainFunctionPeriod`.
⌋(SRS_StbM_20035)

[SWS_CanTSyn_00104]
⌈The scheduled function `CanTSyn_MainFunction()` shall inform the DET, if development error detection is enabled (`CANTSYN_DEV_ERROR_DETECT` is set to TRUE) and if function call has failed because of the following reasons:

- The CanTSyn module was not initialized (`CANTSYN_E_NOT_INITIALIZED`).

⌋(SRS_BSW_00337)

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	*Description*
StbM_GetCurrentTimeDiff	Returns the time difference of current time raw that is valid at this time minus given time raw by using a most accurate time source.
StbM_GetCurrentTimeRaw	Returns a time value in raw format from the most accurate time source.

](SRS_StbM_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	*Description*
CanIf_Transmit	This service initiates a request for transmission of the CAN L-PDU specified by the CanTxSduId and CAN related data in the L-SDU structure.
Crc_CalculateCRC8H2F	This service makes a CRC8 calculation with the Polynomial 0x2F on Crc_Length
Det_ReportError	Service to report development errors.
StbM_BusSetGlobalTime	Allows the Timebase Provider Modules to forward a new Global Time to the StbM, which has been received from different busses.
StbM_GetCurrentTime	Returns a time value (Local Time Base derived from Global Time Base) in standard format.
StbM_GetCurrentTimeExtended	Returns a time value (Local Time Base derived from Global Time Base) in extended format.
StbM_GetOffset	Allows the Timebase Provider Modules to get the currentoffset time.
StbM_SetOffset	Allows the Customers and the Timebase Provider Modules to set the offset time that has to be valid for the system.

](SRS_StbM_20035)

9 Sequence diagrams

9.1 StbM_GetCurrentTime <Master CAN SYNC/FUP>

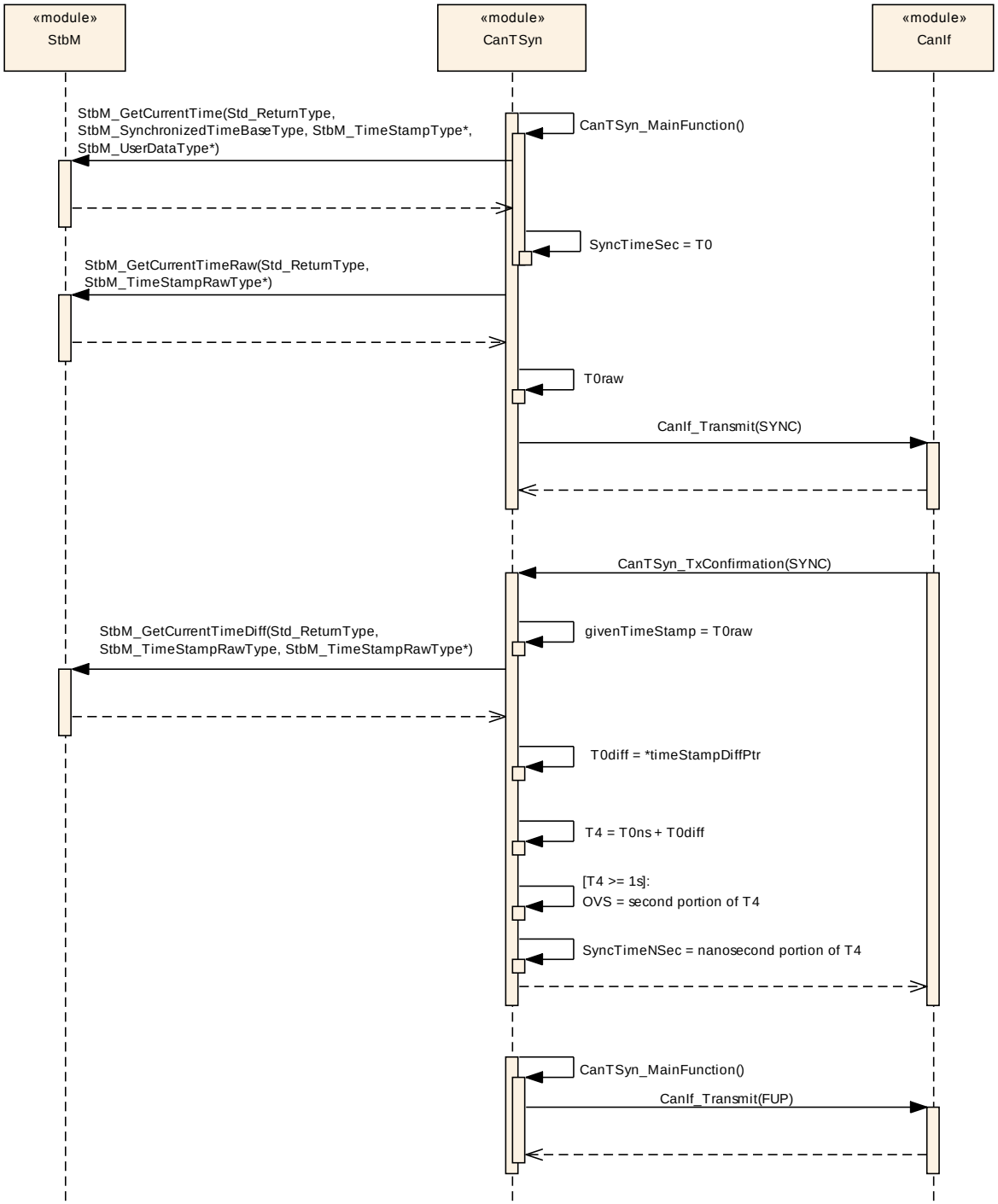


Figure 5: Master CAN SYNC/FUP

9.2 StbM_BusSetGlobalTime <Slave CAN SYNC/FUP>

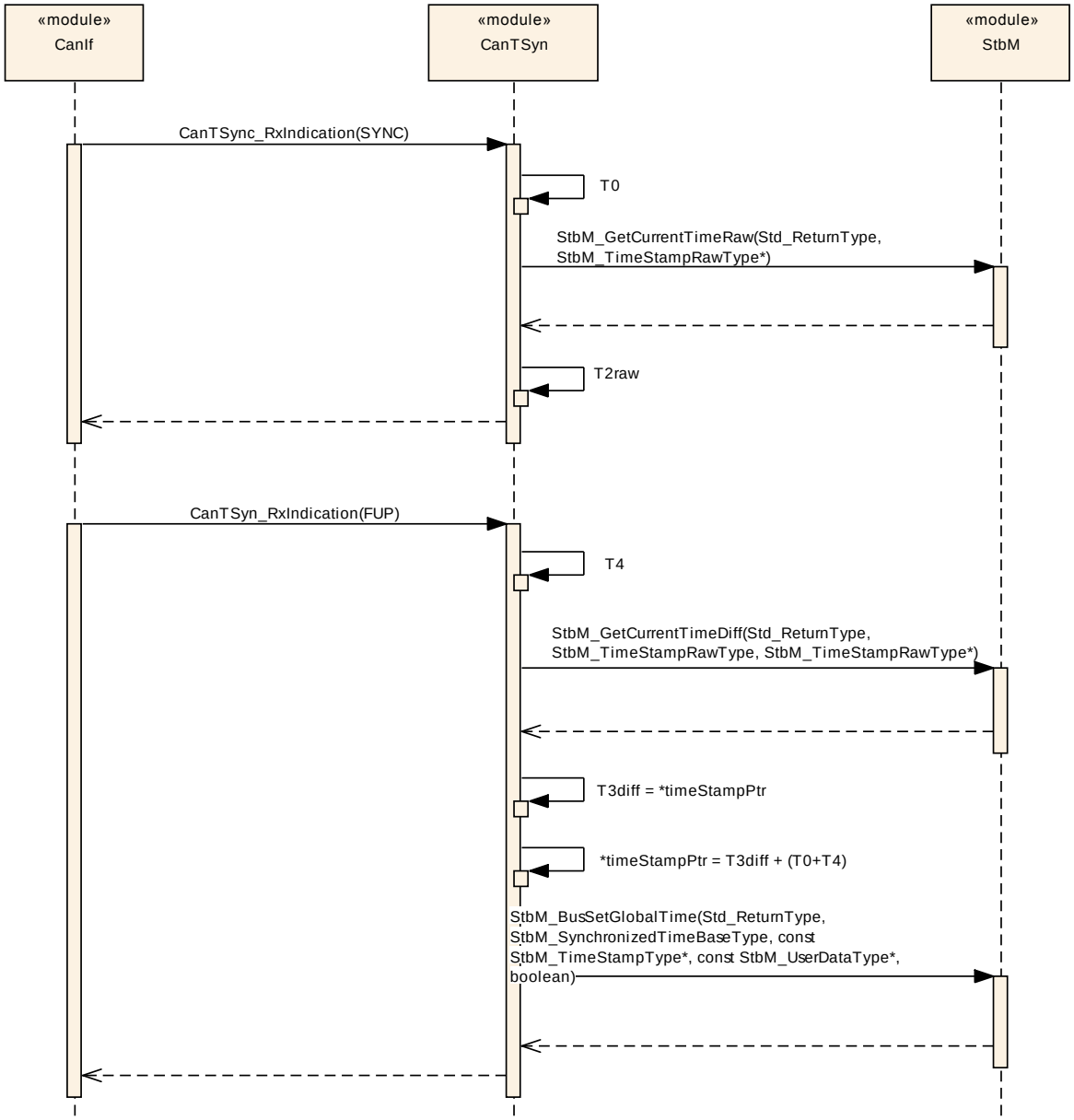


Figure 6: Slave CAN SYNC/FUP

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 10.3 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

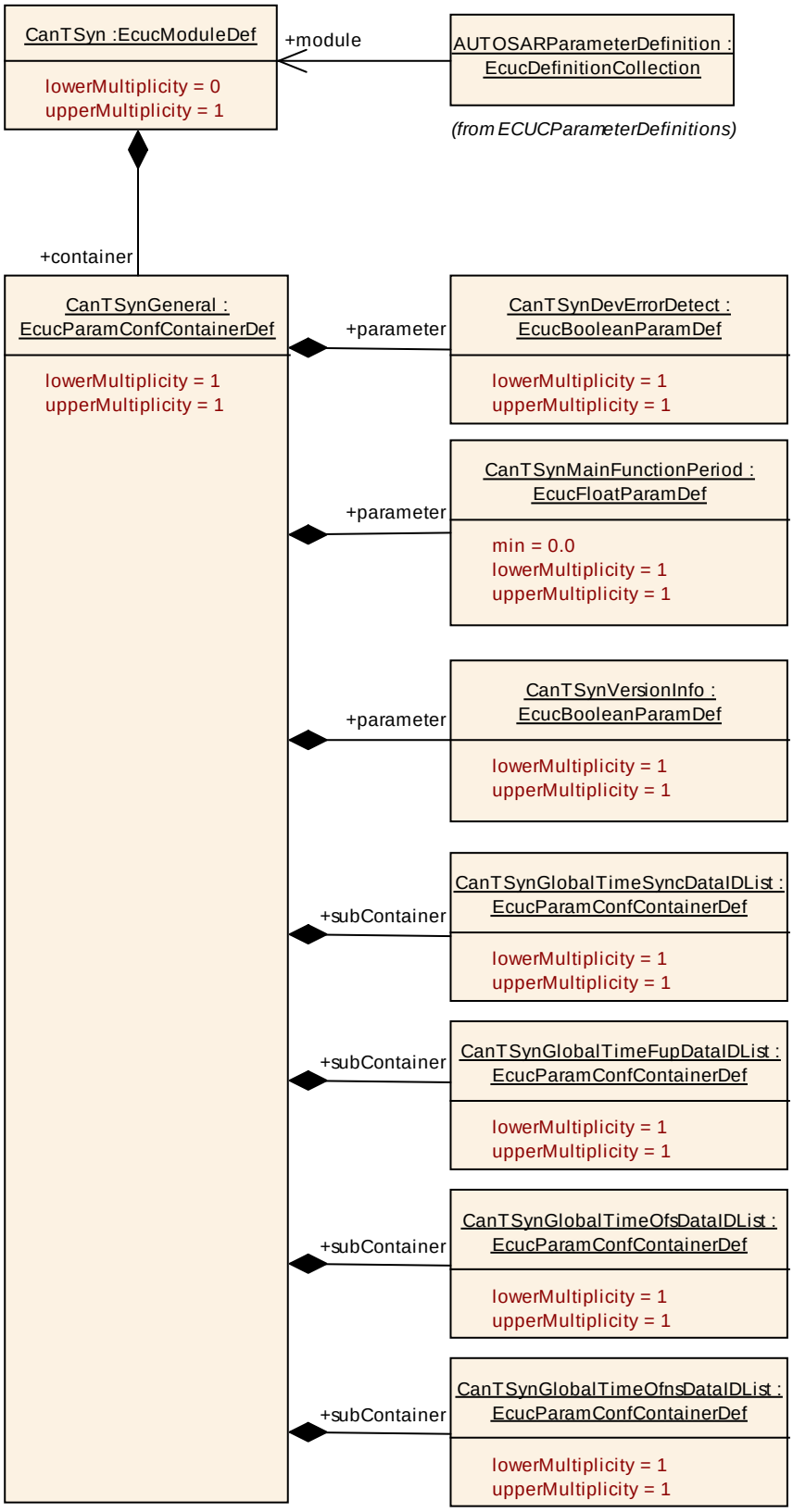
[SWS_CanTSyn_00107]
⌈The Time Synchronization over CAN shall support the configuration variants VARIANT-PRE-COMPILE, VARIANT-LINK-TIME, and VARIANT-POST-BUILD.
⌋(SRS_BSW_00396)

[SWS_CanTSyn_00108]
⌈The Time Synchronization over CAN shall support the configuration for Time Master, Time Slave and Time Gateway.
⌋(SRS_StbM_20038)

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

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.


CanTSyn :EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+module
AUTOSARParameterDefinition : EcucDefinitionCollection
(from ECUCParameterDefinitions)
+container
CanTSynGeneral : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
CanTSynDevErrorDetect : EcucBooleanParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
CanTSynMainFunctionPeriod : EcucFloatParamDef
min = 0.0
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
CanTSynVersionInfo : EcucBooleanParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
+subContainer
CanTSynGlobalTimeSyncDataIDList : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = 1
+subContainer
CanTSynGlobalTimeFupDataIDList : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = 1
+subContainer
CanTSynGlobalTimeOfsDataIDList : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = 1
+subContainer
CanTSynGlobalTimeOfnsDataIDList : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = 1

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		
Description	Switches the Default Error Tracer (Det) detection and notification ON or OFF. • true: enabled (ON). • false: disabled (OFF).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_CanTSyn_00019 :**		
Name	CanTSynMainFunctionPeriod		
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	CanTSynVersionInfo		
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	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

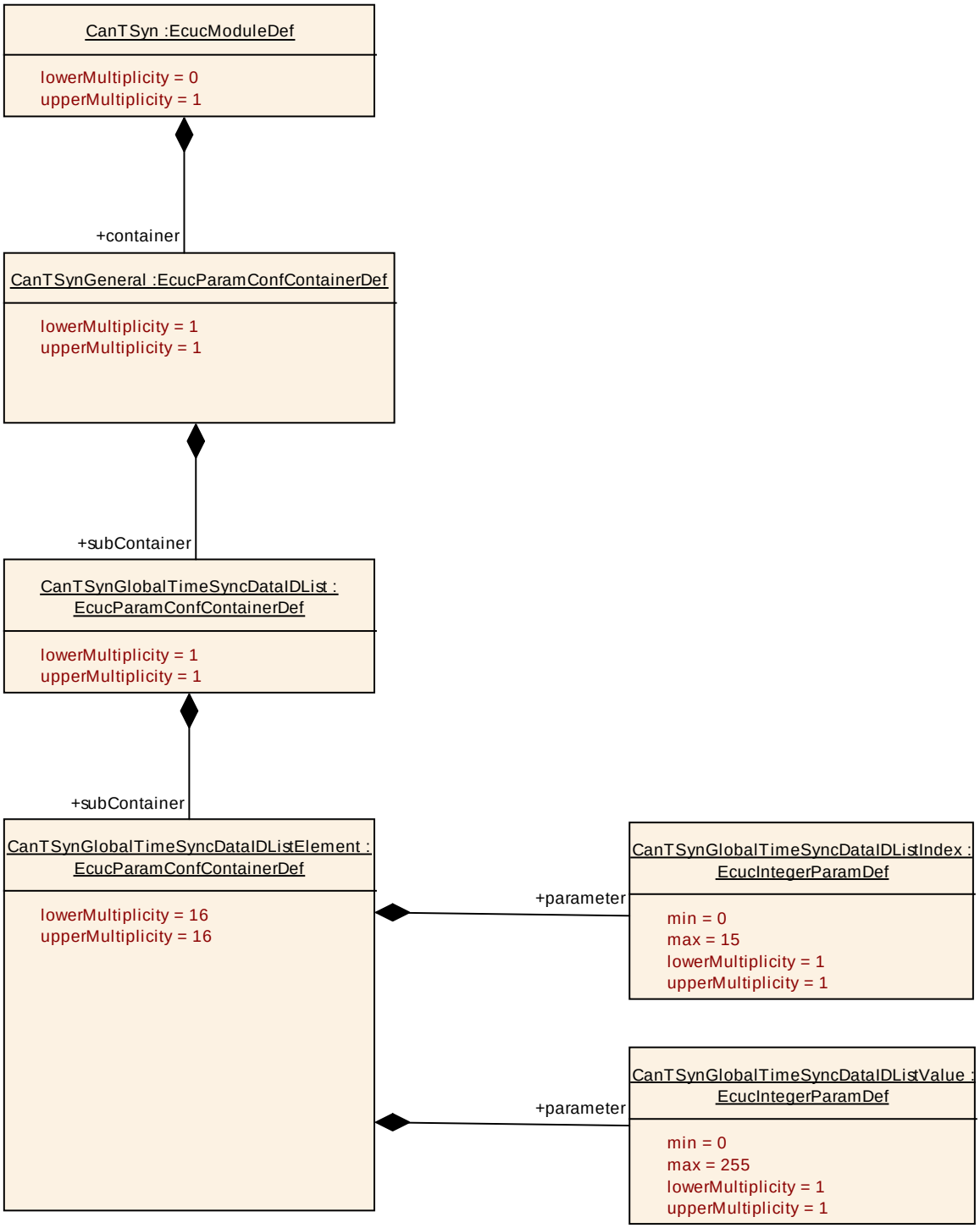
Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CanTSynGlobalTimeFupDataIDList	1	The DataIDList for FUP messages ensures the identification of data elements due to CRC calculation

		process.
CanTSynGlobalTimeOfnsDataIDList	1	The DataIDList for OFNS messages ensures the identification of data elements due to CRC calculation process.
CanTSynGlobalTimeOfsDataIDList	1	The DataIDList for OFS messages ensures the identification of data elements due to CRC calculation process.
CanTSynGlobalTimeSyncDataIDList	1	The DataIDList for SYNC messages ensures the identification of data elements due to CRC calculation process.

10.2.4 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 process.
Configuration Parameters	

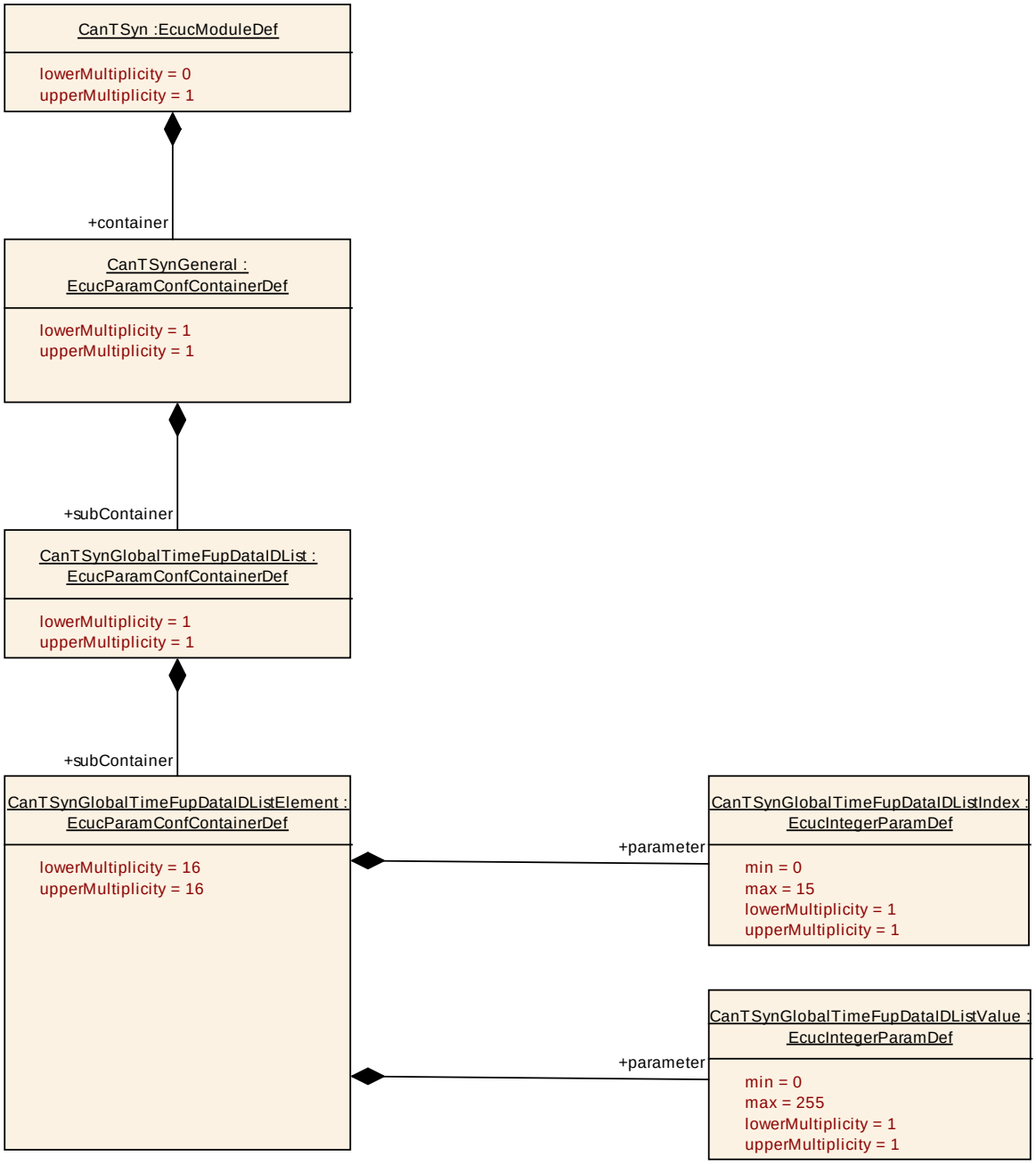
Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CanTSynGlobalTimeSyncDataIDListElement	16	Element of the DataIDList for SYNC messages ensures the identification of data elements due to CRC calculation process.



10.2.5 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 process.
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 process.



10.2.6 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 process.		
Configuration Parameters			

SWS Item	**ECUC_CanTSyn_00032 :**		
Name	CanTSynGlobalTimeFupDataIDListIndex		
Description	Index of the DataIDList for FUP messages ensures the identification of data elements due to CRC calculation process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 15		
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		

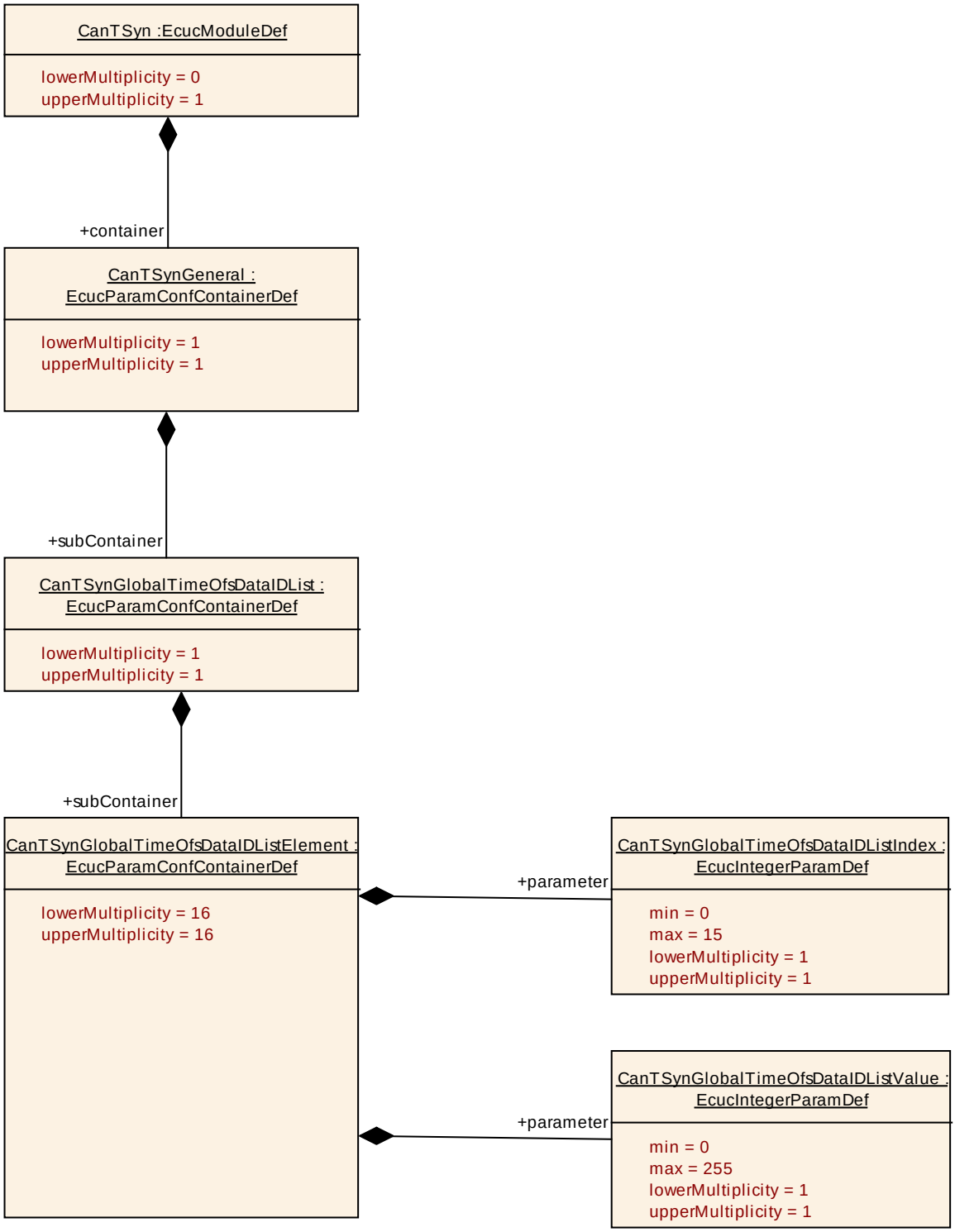
SWS Item	**ECUC_CanTSyn_00033 :**		
Name	CanTSynGlobalTimeFupDataIDListValue		
Description	Value of the DataIDList for FUP messages ensures the identification of data elements due to CRC calculation process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.7 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 process.
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 process.



10.2.8 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 process.
Configuration Parameters	

SWS Item	**ECUC_CanTSyn_00035 :**		
Name	CanTSynGlobalTimeOfsDataIDListIndex		
Description	Index of the DataIDList for OFS messages ensures the identification of data elements due to CRC calculation process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 15		
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		

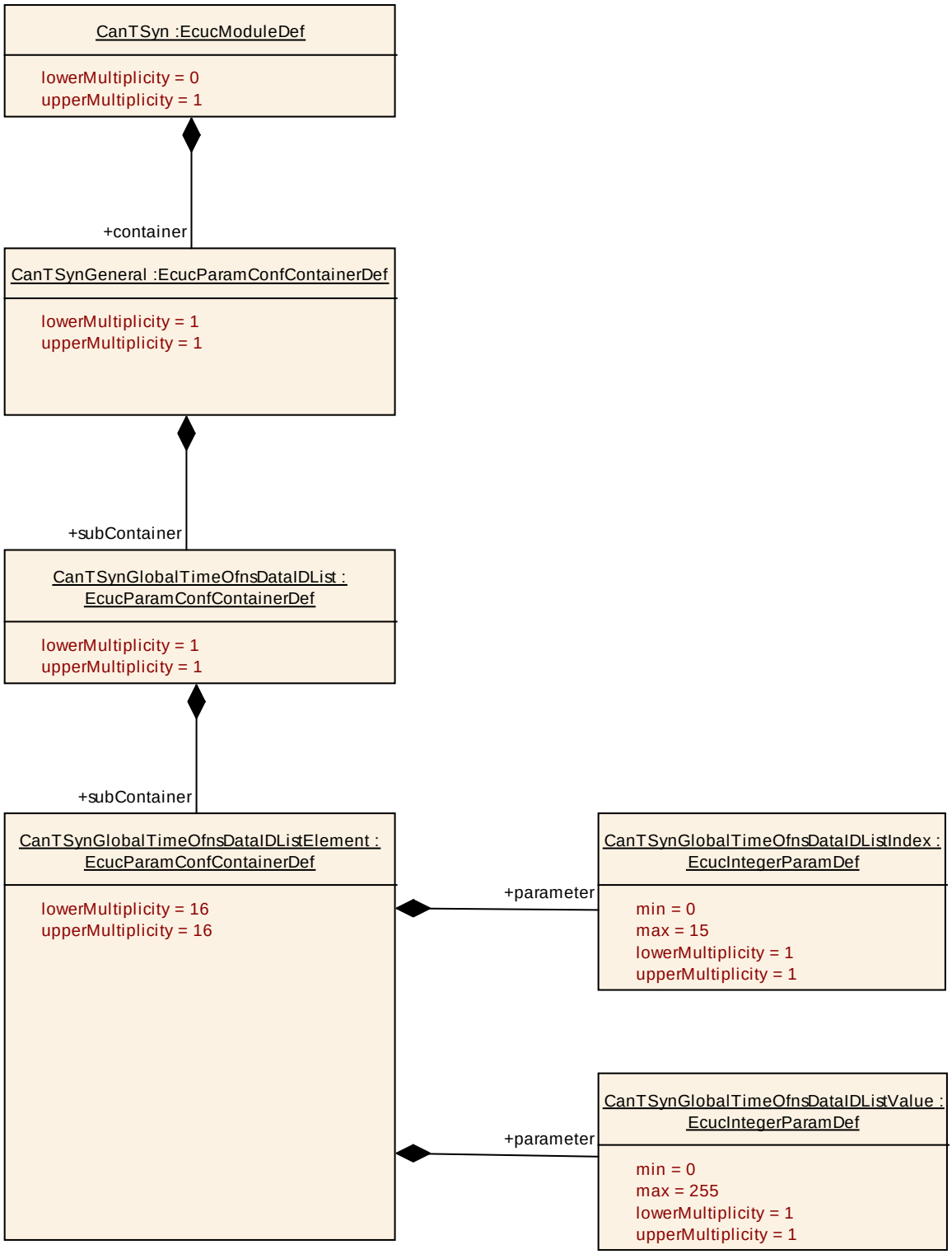
SWS Item	**ECUC_CanTSyn_00036 :**		
Name	CanTSynGlobalTimeOfsDataIDListValue		
Description	Value of the DataIDList for OFS messages ensures the identification of data elements due to CRC calculation process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.9 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 process.
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 process.



10.2.10 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 process.

Configuration Parameters

SWS Item	**ECUC_CanTSyn_00038 :**		
Name	CanTSynGlobalTimeOfnsDataIDListIndex		
Description	Index of the DataIDList for OFNS messages ensures the identification of data elements due to CRC calculation process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 15		
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_00039 :**		
Name	CanTSynGlobalTimeOfnsDataIDListValue		
Description	Value of the DataIDList for OFNS messages ensures the identification of data elements due to CRC calculation process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
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		

No Included Containers

10.2.11 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		
Description	This represents the handle ID of the PDU that contains the global time information.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

SWS Item	**ECUC_CanTSyn_00040 :**		
Name	CanTSynGlobalTimePduRef		
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	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2.12 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		
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		

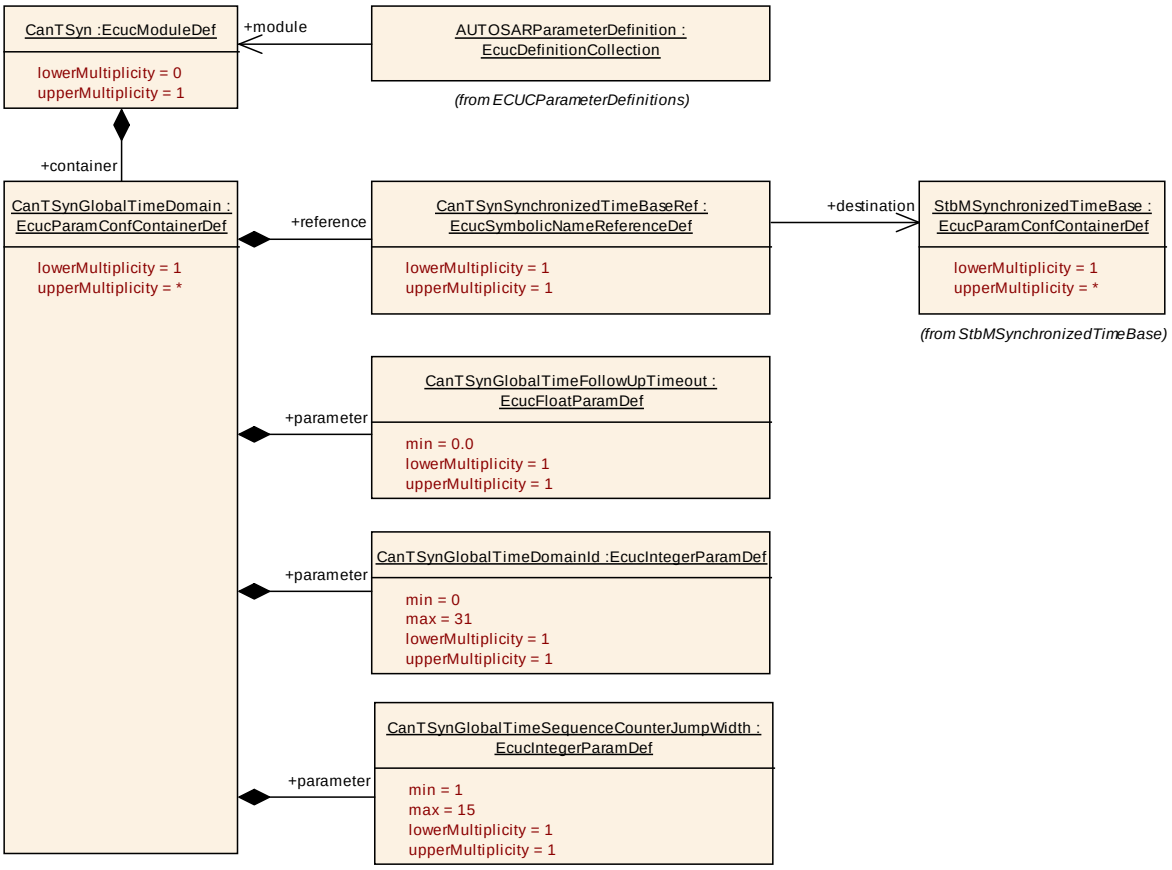
SWS Item	**ECUC_CanTSyn_00006 :**		
Name	CanTSynGlobalTimeFollowUpTimeout		
Description	Rx timeout for the follow-up message. This is only relevant for selected bus systems Unit:seconds		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants

	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_CanTSyn_00011 :**	
Name	CanTSynGlobalTimeSequenceCounterJumpWidth	
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	false	
Scope / Dependency	scope: local	

SWS Item	**ECUC_CanTSyn_00022 :**		
Name	CanTSynSynchronizedTimeBaseRef		
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***
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.
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.



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.
Configuration Parameters	

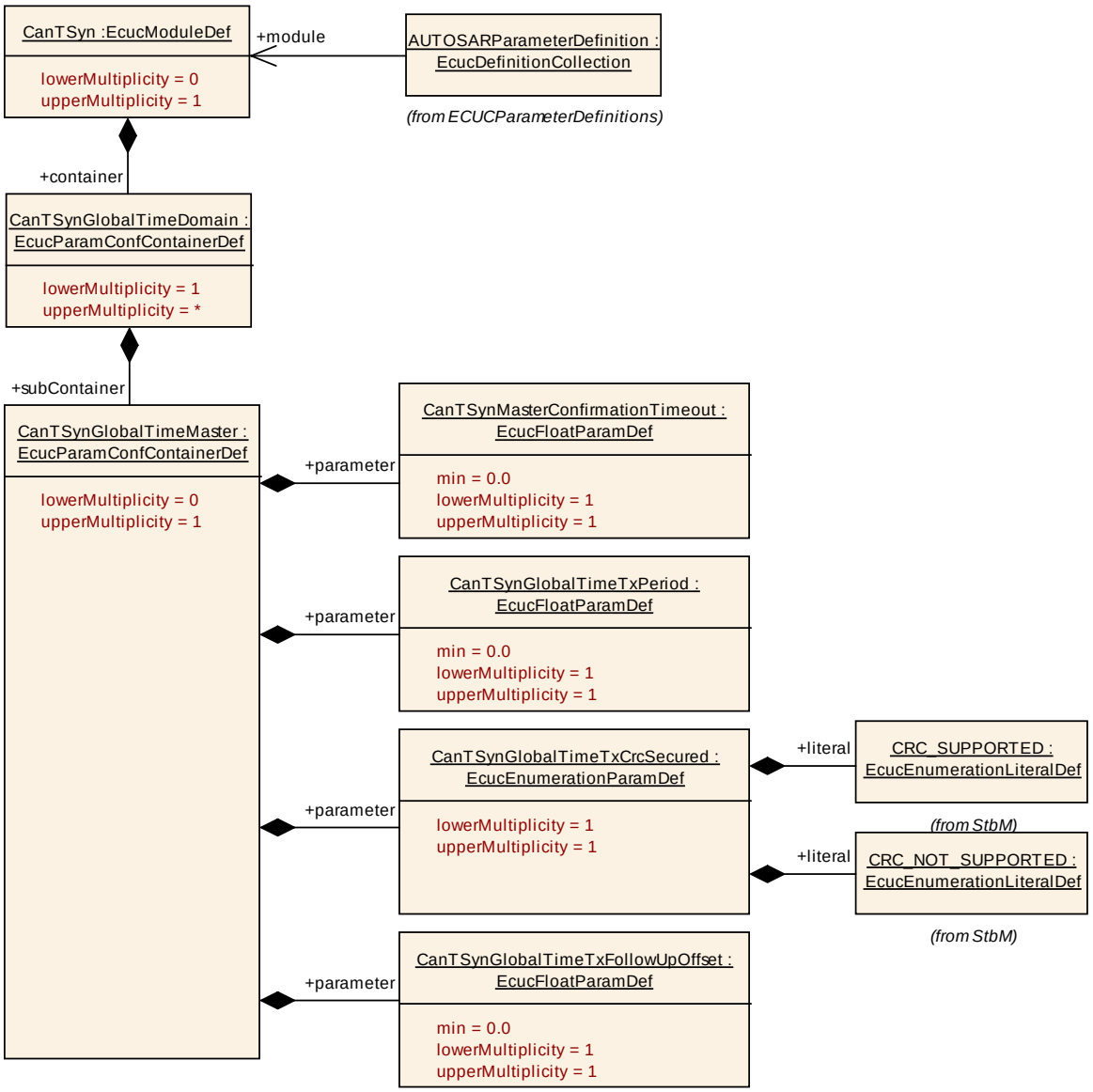
SWS Item	**ECUC_CanTSyn_00015 :**		
Name	CanTSynGlobalTimeTxCrcSecured		
Description	This represents the configuration of whether or not CRC is supported.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRC_NOT_SUPPORTED	This represents a configuration where CRC is not supported.	
	CRC_SUPPORTED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

SWS Item	**ECUC_CanTSyn_00016 :**		
Name	CanTSynGlobalTimeTxFollowUpOffset		
Description	This represents the transmission time offset between a SYNC message and the related FUP message resp. OFS message and related OFNS message. Unit: seconds.		
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			

SWS Item	**ECUC_CanTSyn_00017 :**		
Name	CanTSynGlobalTimeTxPeriod		
Description	This represents configuration of the TX period. Unit: seconds		
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_00020 :**		
Name	CanTSynMasterConfirmationTimeout		
Description	This represents the confirmation timeout after transmission of a SYNC message resp. OFS message. Unit: seconds.		
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		

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.



10.2.14 CanTSynGlobalTimeMasterPdu

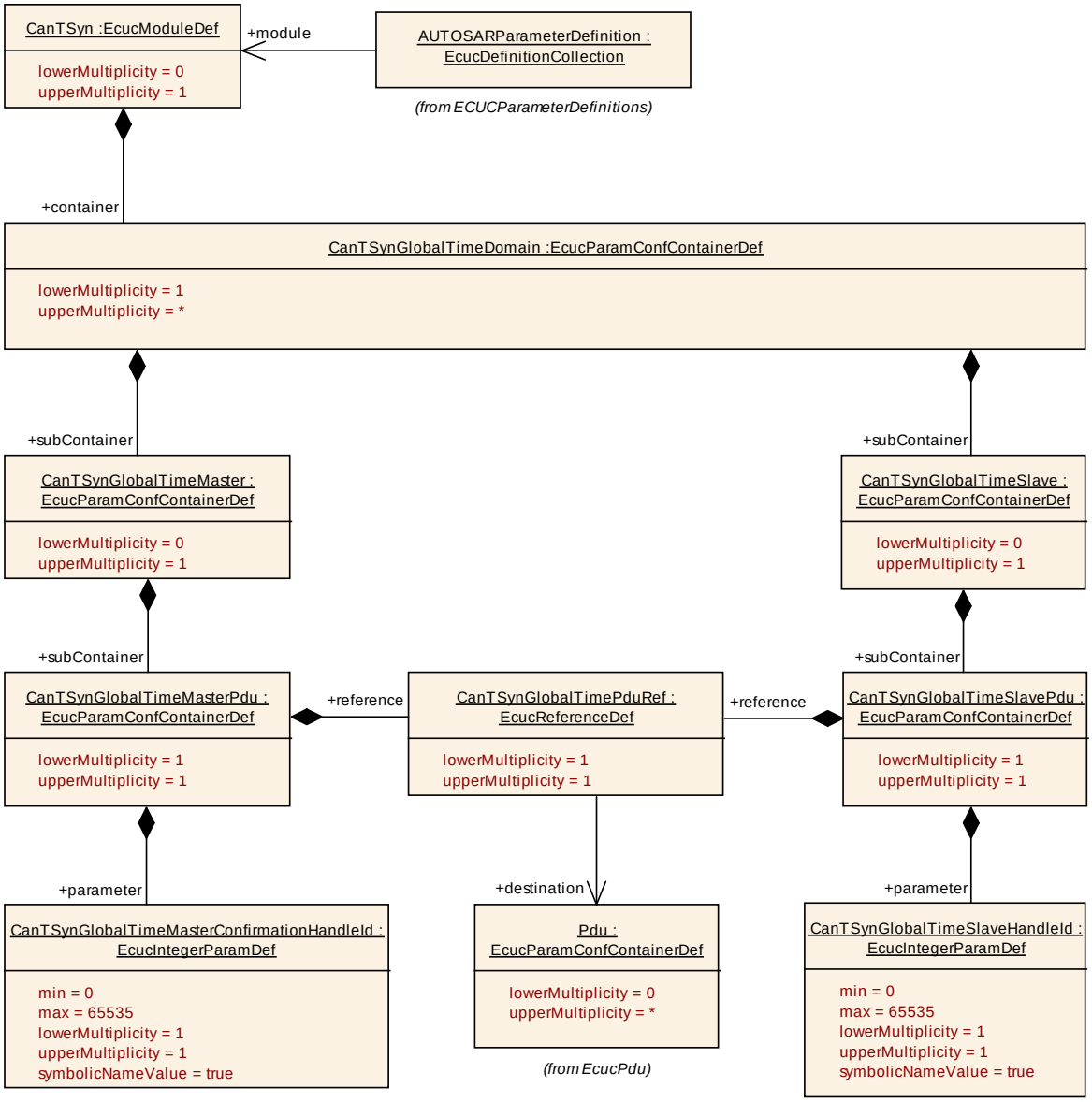
SWS Item	**ECUC_CanTSyn_00009 :**
Container Name	CanTSynGlobalTimeMasterPdu
Description	This container encloses the configuration of the PDU that is supposed to contain the global time information.
Configuration Parameters	

SWS Item	**ECUC_CanTSyn_00008 :**
Name	CanTSynGlobalTimeMasterConfirmationHandleId
Description	This represents the handle ID of the PDU that contains the global time information.
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 65535
Default value	--

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

SWS Item	**ECUC_CanTSyn_00027 :**		
Name	CanTSynGlobalTimePduRef		
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	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers



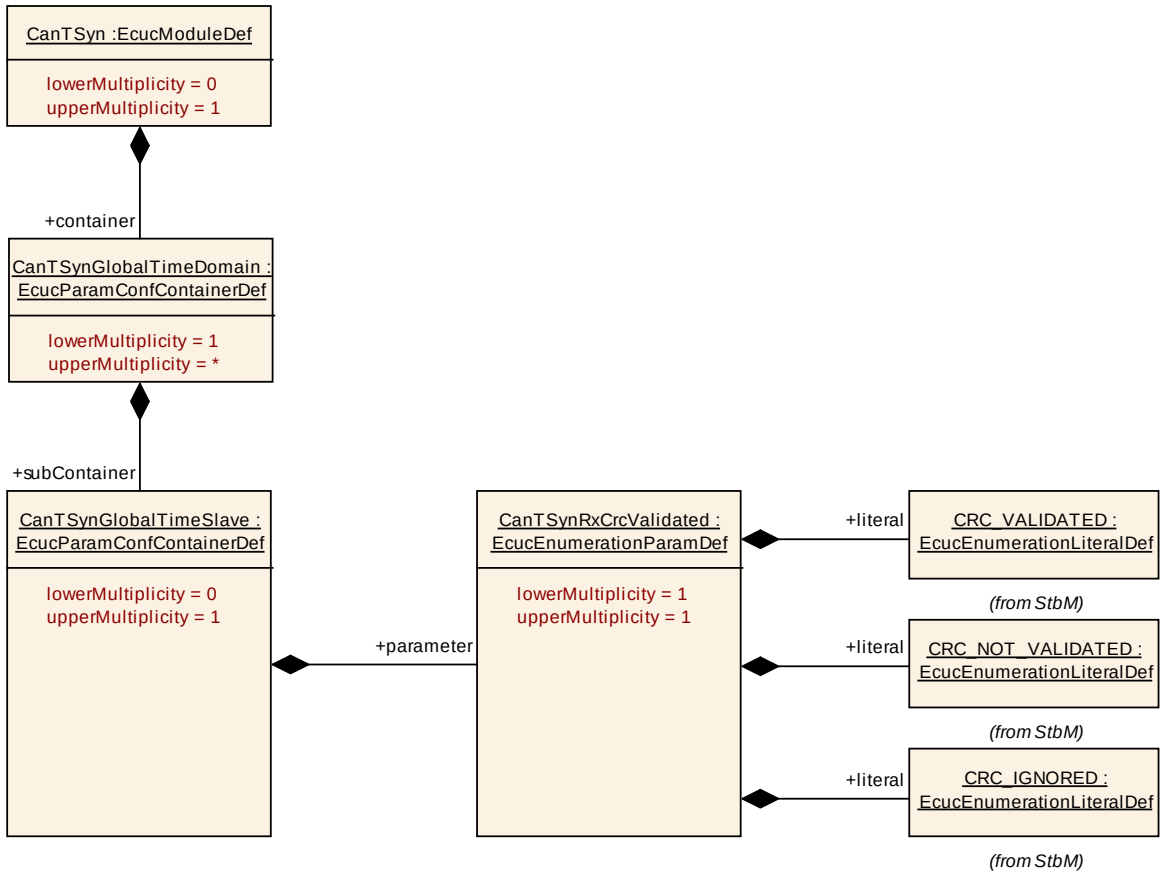
10.2.15 CanTSynGlobalTimeSlave

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.
Configuration Parameters	

SWS Item	**ECUC_CanTSyn_00021 :**	
Name	CanTSynRxCrcValidated	
Description	Definition of whether or not validation of the CRC is supported.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	CRC_IGNORED	The FrTSyn accepts time synchronization messages with all

		Types. The CRC will be ignored.	
	CRC_NOT_VALIDATED	The FrTSyn accepts time synchronization messages with Type equal to 0x10, 0x30 without validating the CRC. All other time synchronization messages are ignored.	
	CRC_VALIDATED	The FrTSyn accepts time synchronization messages with Type equal to 0x20, 0x40 with a correct CRC value. All other time synchronization messages are ignored.	
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***
CanTSynGlobalTimeSlavePdu	1	This container encloses the configuration of the PDU that is supposed to contain the global time information.



10.3 Published Information

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