Document Title	Specification of Time Synchronization over CAN
Document Category	Software specification (SWS)
Document UID	674
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Chapter 8 restructured • CanTSyn_GetVersionInfo removed
2025-11-27	R25-11	AUTOSAR Release Management	• Document split into a protocol spec part (PRS) and a SWS CanTSyn
2024-11-27	R24-11	AUTOSAR Release Management	• Offset Time Domains removed • Changed lower layer from CanIf to LSduR • Static link delay on CAN added • CRC calculation specification corrected
2023-11-23	R23-11	AUTOSAR Release Management	• Clarification of / refinement of sequence counter validation • Clarification of / refinement of Timesync message transmission and debouncing behavior • Incorporation of validation findings for "Secured Time Synchronization"
2022-11-24	R22-11	AUTOSAR Release Management	• Support for "Secured Time Synchronization" added • Support for rate corrected Sync reception delay • Minor content changes, clarifications
2021-11-25	R21-11	AUTOSAR Release Management	• CAN HW timestamping added • Hysteresis added for sequence counter validation

△

2020-11-30	R20-11	AUTOSAR Release Management	• Time Validation updated for gateways • Time out handling of Synchronized and Offset Time messages corrected • Post build variant value corrected for CanTSynGlobalTimeMasterConfirmationHandleId and CanTSynGlobalTimeSlaveHandleId
2019-11-28	R19-11	AUTOSAR Release Management	• Time Validation (draft) • Clarification regarding messages with stuck sequence counter • Clarification regarding cyclic operation entry after timebase startup • Clarification regarding transmission and reception of User Bytes • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Modifications to enhance the precision of Global Time Synchronization • Additional minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2017-12-08	4.3.1	AUTOSAR Release Management	• Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2016-11-30	4.3.0	AUTOSAR Release Management	• Offset message formats changed • Extended Offset message formats added • Immediate Time Synchronization message transmission • Various enhancements and corrections
2015-07-31	4.2.2	AUTOSAR Release Management	• CanTSyn_SetTransmissionMode changed to return "void" • Minor corrections / clarifications / editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	• Initial Release

Disclaimer

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 8
2 Acronyms and Abbreviations 9
3 Related documentation 10
3.1 Input documents & related standards and norms . . . 10
3.2 Related specification . . . 10
4 Constraints and assumptions 11
4.1 Limitations . . . 11
4.2 Applicability to car domains . . . 11
5 Dependencies to other modules 12
5.1 File structure . . . 14
5.1.1 Code file structure . . . 14
5.1.2 Header file structure . . . 14
6 Requirements Tracing 15
7 Functional specification 17
7.1 Overview . . . 17
7.2 Module Handling . . . 17
7.2.1 Interface to Upper Layer . . . 17
7.2.2 Interrupt Handling . . . 17
7.2.3 Initialization . . . 18
7.3 Message Format . . . 18
7.4 Acting as Time Master . . . 18
7.4.1 Time Synchronization Message Transmission . . . 19
7.4.1.1 Transmission mode . . . 20
7.4.1.2 Triggering of SYNC Message Transmission . . . 20
7.4.1.3 Debouncing of Time Synchronization Message Transmission . . . 25
7.4.2 Calculation and Assembling of Time Synchronization Messages . . . 26
7.4.2.1 Global Time Calculation . . . 26
7.4.2.2 User Data . . . 30
7.4.2.3 OVS Calculation . . . 30
7.4.2.4 SGW Calculation . . . 30
7.4.2.5 Sequence Counter Calculation . . . 31
7.4.2.6 CRC Calculation . . . 31
7.4.2.7 ICV Generation . . . 31
7.4.2.8 Message Assembling . . . 35
7.5 Acting as Time Slave . . . 35
7.5.1 SYNC and FUP message processing . . . 36
7.5.2 Validation and Disassembling of Time Synchronization Messages . . . 36
7.5.2.1 Global Time Calculation . . . 36

7.5.2.2 OVS Consideration . . . 41
7.5.2.3 SGW Calculation . . . 41
7.5.2.4 Sequence Counter Validation . . . 41
7.5.2.5 CRC Validation . . . 41
7.5.2.6 ICV Verification . . . 42
7.5.2.7 Message Disassembling . . . 46
7.6 Time Recording . . . 47
7.6.1 Global Time Precision Measurement . . . 47
7.6.2 Time Validation . . . 47
7.7 Security Events . . . 49
7.8 Error Classification . . . 51
7.8.1 Development Errors . . . 51
7.8.2 Runtime Errors . . . 52
7.8.3 Production Errors . . . 52
8 API specification 53
8.1 Imported types . . . 53
8.2 Type definitions . . . 54
8.2.1 CanTSyn_ConfigType . . . 54
8.2.2 CanTSyn_TransmissionModeType . . . 54
8.3 Function definitions . . . 54
8.3.1 CanTSyn_Init . . . 55
8.3.2 CanTSyn_SetTransmissionMode . . . 55
8.3.3 CanTSyn_RxIndication . . . 56
8.3.4 CanTSyn_TxConfirmation . . . 57
8.3.5 CanTSyn_IcvGenerationIndication . . . 58
8.3.6 CanTSyn_IcvVerificationIndication . . . 59
8.4 Scheduled functions . . . 60
8.4.1 CanTSyn_MainFunction . . . 60
8.5 Expected interfaces . . . 60
8.5.1 Mandatory interfaces . . . 60
8.5.2 Optional interfaces . . . 61
8.5.3 Callout Interfaces . . . 62
8.5.4 Configurable Interfaces . . . 62
8.6 Service Interfaces . . . 62
9 Sequence diagrams 63
9.1 Enable Egress Timestamping . . . 63
9.2 CAN Time Synchronization (Time Master) . . . 64
9.3 CAN Time Synchronization (Time Slave) . . . 66
9.4 CAN Secure Time Synchronization (Time Master, Time Slave) . . . 68

10 Configuration specification 69
10.1 How to read this chapter . . . 69
10.2 Containers and configuration parameters . . . 69
10.2.1 Variants . . . 69
10.2.2 CanTSyn . . . 69
10.2.3 CanTSynGeneral . . . 70
10.2.4 CanTSynSecurityEventRefs . . . 73
10.2.5 CanTSynGlobalTimeDomain . . . 75
10.2.6 CanTSynGlobalTimeSyncDataIDList . . . 78
10.2.7 CanTSynGlobalTimeSyncDataIDListElement . . . 79
10.2.8 CanTSynGlobalTimeFupDataIDList . . . 80
10.2.9 CanTSynGlobalTimeFupDataIDListElement . . . 81
10.2.10 CanTSynGlobalTimeMaster . . . 83
10.2.11 CanTSynGlobalTimeMasterPdu . . . 86
10.2.12 CanTSynGlobalTimeTxIcvGeneration . . . 88
10.2.13 CanTSynGlobalTimeSlave . . . 92
10.2.14 CanTSynGlobalTimeSlavePdu . . . 96
10.2.15 CanTSynGlobalTimeRxIcvVerification . . . 97
10.3 Constraints . . . 101
10.4 Published Information . . . 101
A Not applicable requirements 102
B Change history of AUTOSAR traceable items 103
B.1 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 103
B.1.1 Added Specification Items in R26-11 . . . 103
B.1.2 Changed Specification Items in R26-11 . . . 103
B.1.3 Deleted Specification Items in R26-11 . . . 103
B.1.4 Added Constraints in R26-11 . . . 103
B.1.5 Changed Constraints in R26-11 . . . 103
B.1.6 Deleted Constraints in R26-11 . . . 104
B.2 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 104
B.2.1 Added Specification Items in R25-11 . . . 104
B.2.2 Changed Specification Items in R25-11 . . . 104
B.2.3 Deleted Specification Items in R25-11 . . . 104
B.2.4 Added Constraints in R25-11 . . . 105
B.2.5 Changed Constraints in R25-11 . . . 105
B.2.6 Deleted Constraints in R25-11 . . . 105
B.3 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 105
B.3.1 Added Specification Items in R24-11 . . . 105
B.3.2 Changed Specification Items in R24-11 . . . 105

B.3.3 Deleted Specification Items in R24-11 . . . 106
B.3.4 Added Constraints in R24-11 . . . 106
B.3.5 Changed Constraints in R24-11 . . . 106
B.3.6 Deleted Constraints in R24-11 . . . 106
B.4 Traceable item history of this document according to AUTOSAR Release R23-11 . . . 107
B.4.1 Added Specification Items in R23-11 . . . 107
B.4.2 Changed Specification Items in R23-11 . . . 107
B.4.3 Deleted Specification Items in R23-11 . . . 107
B.4.4 Added Constraints in R23-11 . . . 107
B.4.5 Changed Constraints in R23-11 . . . 108
B.4.6 Deleted Constraints in R23-11 . . . 108
C Mapping Tables 109
C.1 Mapping of Protocol Inputs . . . 109
C.2 Mapping of Protocol Outputs . . . 109
C.3 Mapping of Protocol Timing Values . . . 110
C.4 Mapping of Protocol Parameters . . . 112
C.4.1 Mapping of Protocol Parameters of a Time Domain . . . 112
C.4.2 Mapping of Protocol Parameters of the Time Master of a Time Domain 112
C.4.3 Mapping of Protocol Parameters of a Time Slave of a Time Domain 113

1 Introduction and functional overview

The CanTSyn module implements the Time Synchronization over CAN protocol as specified in [1].

Figure 1.1 below shows the CanTSyn module and other time provider modules in context of the Layered SW Architecture of AUTOSAR.

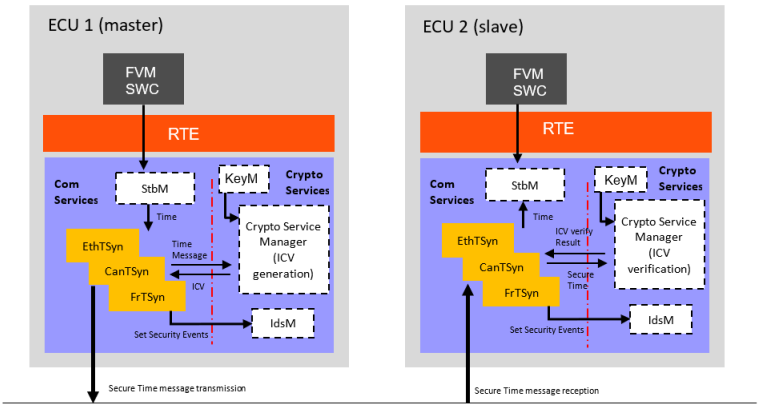


Figure 1.1: Timesync Modules in the AUTOSAR Layered SW Architecture

2 Acronyms and Abbreviations

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

Abbreviation	Description
GTM	Global Time Master
BswM	BSW Mode Manager module
<Bus>TSyn	Bus specific Time Synchronization module
CAN FD	Controller Area Network (CAN) - Flexible Data Rate
CanTSyn	Time Synchronization over CAN module
CRC	Cyclic Redundancy Checksum
CSM	Crypto Service Manager
Debounce Time	Minimum gap between two TX messages with the same PDU
Det	Default Error Tracer module
DLC	Data Length Code
DoS	Denial of Service
CanIf	CAN interface module
FUP message	Follow-Up message
FV	Freshness Value
FVM	Freshness Value Manager
ICV	Integrity Check Value
IdsM	Intrusion Detection System Manager
LSduR	L-SDU Router module
MAC	Message Authentication Code
OVS	Overflow Seconds value (field in FUP message)
SC	Sequence Counter in Time Synchronization messages
SGW	"Synchronized to Gateway" state of Time Synchronization
StbM	Synchronized Time-Base Manager
SYNC message	Time Synchronization message
Timesync	Time Synchronization

Table 2.1: Acronyms and Abbreviations

3 Related documentation

3.1 Input documents & related standards and norms

[1] Time Synchronization over CAN Protocol Specification
AUTOSAR_FO_PRS_TimeSyncOverCANProtocol

[2] Requirements on Time Synchronization
AUTOSAR_FO_RS_TimeSync

[3] Glossary
AUTOSAR_FO_TR_Glossary

[4] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[5] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[6] Specification of Synchronized Time-Base Manager
AUTOSAR_CP_SWS_SynchronizedTimeBaseManager

[7] Specification of CRC Library
AUTOSAR_CP_SWS_CRCLibrary

[8] Specification of Crypto Service Manager
AUTOSAR_CP_SWS_CryptoServiceManager

[9] Specification of Intrusion Detection System Manager
AUTOSAR_CP_SWS_IntrusionDetectionSystemManager

[10] System Template
AUTOSAR_CP_TPS_SystemTemplate

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [4, SWS BSW General], which is also valid for `CanTSyn`.

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

4 Constraints and assumptions

4.1 Limitations

1. The Time Base in the SYNC messages is limited to 32 bit, wherefore the maximum supported time value is 4294967295 seconds (2^{32}-1).
2. The Time Synchronization on CAN protocol assumes a local reference clock with a maximum tick duration of 2μs (i.e., this is equal to a local reference clock with a minimum frequency of 500 kHz)
3. The authentication protection mechanism of the time is not supported on classic CAN busses, due to below reasons.
    - The authentication protection mechanism is complex to achieve on classic CAN busses due to payload limitation and any solution incorporated will leave the security vulnerabilities (e.g., cryptographic attacks, DoS).
    - Today's ECUs in vehicle E/E architecture supports both classic CAN and CanFD channels.

4.2 Applicability to car domains

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

5 Dependencies to other modules

The Time Synchronization over CAN (CanTSyn) has interfaces towards the Synchronized Time-Base Manager (StbM), the CAN Interface (CanIf), the L-SDU Router (LSduR), the BSW Mode Manager (BswM), the Crypto Service Manager (CSM), the Intrusion Detection System Manager (IdsM) and the Default Error Tracer (Det).

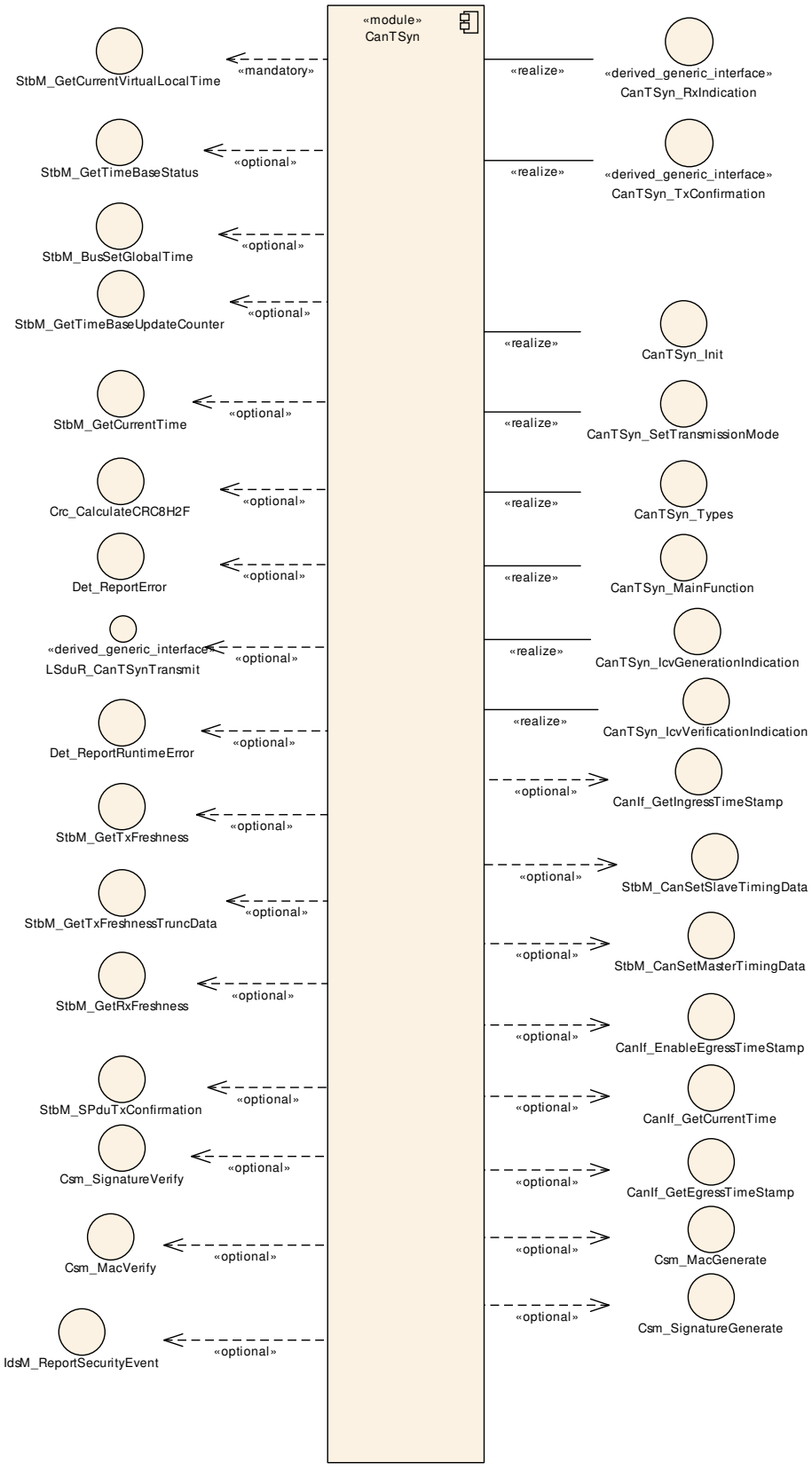


Figure 5.1: Module dependencies of the `CanTSyn` module

- StbM -

  - Get and set the current time value
  - Get FV from FVM
- LSduR - Receiving and transmitting messages
- CanIf - Accessing the clock unit in the CAN controller
- BswM - Coordination of network access (via `CanTSyn_SetTransmissionMode`)
- DET - Reporting of development errors
- CSM -
  - Generation of ICV for Time Master
  - Verification of ICV for Time Slave
- IdsM - Reporting of security events

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 [4].

5.1.2 Header file structure

For details, refer to the section 5.1.7 "Header file structure" of the SWS BSW General [4].

6 Requirements Tracing

The following tables reference the requirements specified in [2, RS TimeSync] and [5, SRS BSWGeneral] and links to the fulfillment of these. Please note that if column "Satisfied by" is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[RS_TS_00003]	Initialize Local Time Base with Configurable Value	[SWS_CanTSyn_00003]
[RS_TS_00004]	Initialize Global Time Base with Configurable Value	[SWS_CanTSyn_00003]
[RS_TS_00034]	Provide Measurement Data to Applications	[SWS_CanTSyn_00137] [SWS_CanTSyn_00138] [SWS_CanTSyn_00139] [SWS_CanTSyn_00140] [SWS_CanTSyn_00141] [SWS_CanTSyn_00142]
[RS_TS_20031]	Trigger Time Sync Transmission over CAN	[SWS_CanTSyn_00028] [SWS_CanTSyn_00043] [SWS_CanTSyn_00044] [SWS_CanTSyn_00117] [SWS_CanTSyn_00118] [SWS_CanTSyn_00120] [SWS_CanTSyn_00121] [SWS_CanTSyn_00122] [SWS_CanTSyn_00136] [SWS_CanTSyn_00220] [SWS_CanTSyn_00228] [SWS_CanTSyn_00229] [SWS_CanTSyn_00230] [SWS_CanTSyn_00238]
[RS_TS_20032]	Provide Time Base after Valid CAN Sync Message	[SWS_CanTSyn_00064] [SWS_CanTSyn_00133] [SWS_CanTSyn_00135]
[RS_TS_20033]	Protect Time Sync Protocol over CAN	[SWS_CanTSyn_00050]
[RS_TS_20034]	Detect Timeout and Integrity Errors in CAN Sync Protocol	[SWS_CanTSyn_00064] [SWS_CanTSyn_00080] [SWS_CanTSyn_00115] [SWS_CanTSyn_00133]
[RS_TS_20035]	Support Precise Time Measurement over CAN	[SWS_CanTSyn_00028] [SWS_CanTSyn_00030] [SWS_CanTSyn_00043] [SWS_CanTSyn_00044] [SWS_CanTSyn_00047] [SWS_CanTSyn_00050] [SWS_CanTSyn_00080] [SWS_CanTSyn_00090] [SWS_CanTSyn_00091] [SWS_CanTSyn_00092] [SWS_CanTSyn_00093] [SWS_CanTSyn_00095] [SWS_CanTSyn_00096] [SWS_CanTSyn_00099] [SWS_CanTSyn_00102] [SWS_CanTSyn_00103] [SWS_CanTSyn_00105] [SWS_CanTSyn_00106] [SWS_CanTSyn_00144] [SWS_CanTSyn_00145] [SWS_CanTSyn_00146] [SWS_CanTSyn_00147] [SWS_CanTSyn_00148] [SWS_CanTSyn_00149] [SWS_CanTSyn_00150] [SWS_CanTSyn_00151] [SWS_CanTSyn_00152] [SWS_CanTSyn_00153] [SWS_CanTSyn_00154]
[RS_TS_20037]	Support User-Specific Data in CAN Time Protocol	[SWS_CanTSyn_00014]
[RS_TS_20038]	CAN Time Sync Role Configuration	[SWS_CanTSyn_00108] [SWS_CanTSyn_00135]
[RS_TS_20070]	Support Hardware and Software Timestamping over CAN	[SWS_CanTSyn_00144] [SWS_CanTSyn_00147] [SWS_CanTSyn_00150] [SWS_CanTSyn_00152] [SWS_CanTSyn_00153]

▽

△

Requirement	Description	Satisfied by
[RS_TS_20073]	Secure Time Synchronization Protocol over CAN	[SWS_CanTSyn_00161] [SWS_CanTSyn_00163] [SWS_CanTSyn_00164] [SWS_CanTSyn_00165] [SWS_CanTSyn_00166] [SWS_CanTSyn_00167] [SWS_CanTSyn_00168] [SWS_CanTSyn_00169] [SWS_CanTSyn_00170] [SWS_CanTSyn_00171] [SWS_CanTSyn_00172] [SWS_CanTSyn_00173] [SWS_CanTSyn_00174] [SWS_CanTSyn_00175] [SWS_CanTSyn_00176] [SWS_CanTSyn_00177] [SWS_CanTSyn_00182] [SWS_CanTSyn_00188] [SWS_CanTSyn_00189] [SWS_CanTSyn_00190] [SWS_CanTSyn_00191] [SWS_CanTSyn_00193] [SWS_CanTSyn_00194] [SWS_CanTSyn_00195] [SWS_CanTSyn_00196] [SWS_CanTSyn_00197] [SWS_CanTSyn_00198] [SWS_CanTSyn_00199] [SWS_CanTSyn_00200] [SWS_CanTSyn_00207] [SWS_CanTSyn_00210] [SWS_CanTSyn_00211] [SWS_CanTSyn_00212] [SWS_CanTSyn_00213] [SWS_CanTSyn_00214] [SWS_CanTSyn_00215] [SWS_CanTSyn_00216] [SWS_CanTSyn_00217] [SWS_CanTSyn_00218] [SWS_CanTSyn_00219] [SWS_CanTSyn_91002] [SWS_CanTSyn_91003]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_CanTSyn_00097] [SWS_CanTSyn_00100] [SWS_CanTSyn_00134] [SWS_CanTSyn_00202] [SWS_CanTSyn_00203]
[SRS_BSW_00337]	Classification of development errors	[SWS_CanTSyn_00097] [SWS_CanTSyn_00100] [SWS_CanTSyn_00134] [SWS_CanTSyn_00202] [SWS_CanTSyn_00203]
[SRS_BSW_00385]	List possible error notifications	[SWS_CanTSyn_00089] [SWS_CanTSyn_91001]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_CanTSyn_00097] [SWS_CanTSyn_00100] [SWS_CanTSyn_00134] [SWS_CanTSyn_00202]
[SRS_BSW_00488]	Classification of security events	[SWS_CanTSyn_00204] [SWS_CanTSyn_92000] [SWS_CanTSyn_92001] [SWS_CanTSyn_92002] [SWS_CanTSyn_92003]
[SRS_BSW_00489]	Reporting of security events	[SWS_CanTSyn_00201] [SWS_CanTSyn_00204] [SWS_CanTSyn_92000] [SWS_CanTSyn_92001] [SWS_CanTSyn_92002] [SWS_CanTSyn_92003]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Overview

This chapter defines the AUTOSAR specific details of the implementation of the Time Synchronization over CAN protocol, which is specified in the protocol specification [1].

The API of the module is defined in chapter 8, while the configuration is defined in chapter 10.

Time Synchronization principles and common wording is described in the SWS Synchronized Time-Base Manager [6] and RS Time Synchronization [2].

7.2 Module Handling

This section specifies general aspects of the implementation of the CanTSyn module.

7.2.1 Interface to Upper Layer

[SWS_CanTSyn_00135]

Upstream requirements: RS_TS_20032, RS_TS_20038

⌈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.⌋

7.2.2 Interrupt Handling

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

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

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

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

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

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

7.2.3 Initialization

The Time Synchronization over CAN is initialized via CanTSyn_Init. Except for 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]

Upstream requirements: RS_TS_00003, RS_TS_00004

⌈A call to CanTSyn_Init initializes all internal variables and sets the Time Synchronization over CAN to the initialized state.
⌋

7.3 Message Format

Refer to the protocol specification [1] Chapter *"Message types"*.

7.4 Acting as Time Master

A Time Master is an entity which is the master for a 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 Time Base, i.e., the Time Base which all other nodes derive their Time Base from, then that Time Master is the system-wide (or Global) Time Master (refer to Figure 7.1). A Time Gateway typically consists of one Time Slaves port and 1..N Time Master ports. When mapping time entities to real ECUs it has to be noted, that an ECU could be the Time Master (or even Global Time Master) for one Time Base and Time Slave for another Time Base.

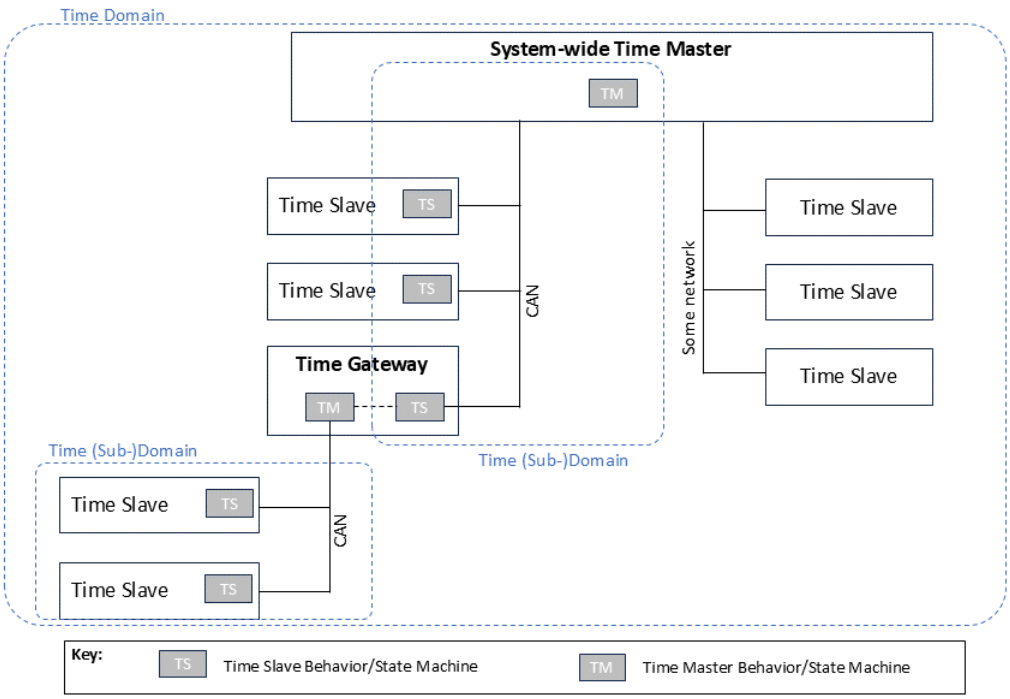


Figure 7.1: Network Topology Example

The CanTSyn supports a fan-out of a given Time Base on several CAN busses in parallel. Those busses all belong to the same Time Domain.

Note: A separate configuration container `CanTSynGlobalTimeDomain` needs to be configured for each CAN bus on which the CanTSyn transmits the Time Base for a given Time Domain - although those busses all belong to the same Time Domain.

The protocol specification [1] Chapter *"Time Master"* specifies the protocol state machine of a Time Master. The sections below specify details how the protocol state machine is realized in the CanTSyn module.

Also refer to the sequence diagrams Figure 9.2 and Figure 9.3 for the interaction with other BSW modules during transmission of a SYNC/FUP message sequence.

7.4.1 Time Synchronization Message Transmission

[SWS_CanTSyn_00136]

Upstream requirements: RS_TS_20031

⌈A Time Master shall transmit SYNC and FUP messages by calling `LSduR_CanTSynTransmit` with the PduId derived via `CanTSynGlobalTimePduRef` of the corresponding Time Domain.⌋

7.4.1.1 Transmission mode

[SWS_CanTSyn_00043]

Upstream requirements: RS_TS_20031, RS_TS_20035

⌈If `CanTSyn_SetTransmissionMode(Controller, Mode)` is called and parameter Mode equals `CANTSYN_TX_OFF`, all transmit requests from CanTSyn shall be omitted on this CAN channel.⌋

[SWS_CanTSyn_00044]

Upstream requirements: RS_TS_20031, RS_TS_20035

⌈If `CanTSyn_SetTransmissionMode(Controller, Mode)` is called and parameter Mode equals `CANTSYN_TX_ON`, all transmit requests from CanTSyn on this CAN channel shall be able to be transmitted.⌋

7.4.1.2 Triggering of SYNC Message Transmission

A Time Synchronization sequence of the Time Master is triggered in the `TM_Idle` state (refer to [1]) according to figure Figure 7.2

- either cyclically (refer to chapter “Cyclic Time Synchronization”) or
- or immediately,i.e., event triggered (refer to chapter “Immediate Time Synchronization”).

Triggering takes the debounce interval into consideration (refer to chapter “Debouncing of Time Synchronization Message Transmission”) to build and transmit the SYNC message.

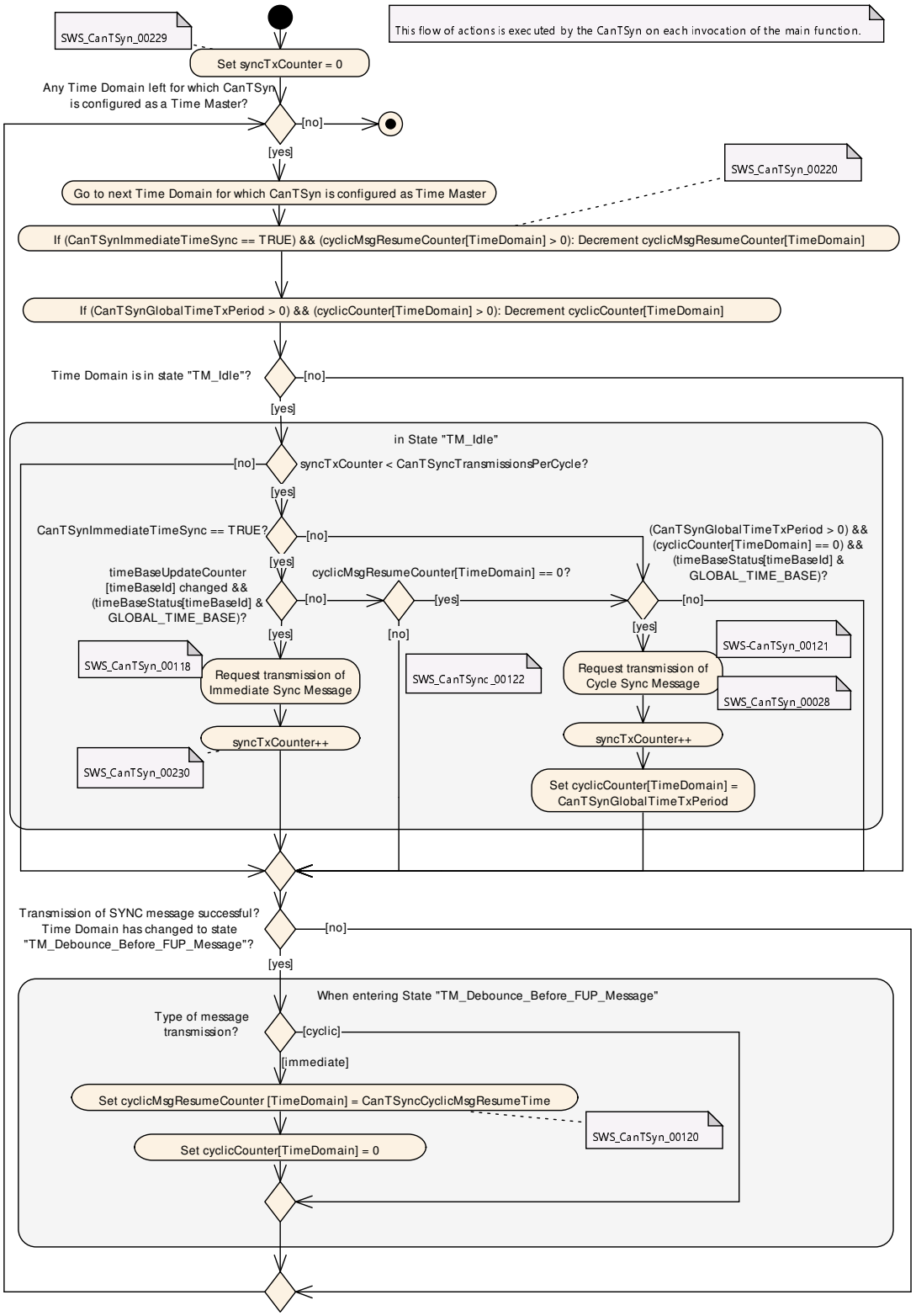


Figure 7.2: Time Master SYNC Message Transmission Request

7.4.1.2.1 Cyclic Time Synchronization

[SWS_CanTSyn_00028]

Upstream requirements: RS_TS_20031, RS_TS_20035

⌈For a Synchronized Time Domain (refer to CanTSynGlobalTimeDomain) if

- the Time Master is in state TM_Idle
- and the GLOBAL_TIME_BASE bit within the timeBaseStatus is set
- and CanTSynGlobalTimeTxPeriod is unequal to 0
- and cylic transmission period has elapsed (cyclicCounter is 0)
- and the associated cyclicMsgResumeCounter is equal to or less than 0
- and syncTxCounter (if configured) is less than CanTSynSyncTransmissionsPerCycle,

then a Time Master shall

- request a transmission of a SYNC message for the corresponding Time Domain
- and restart the cyclicCounter by setting it to the value of CanTSynGlobalTimeTxPeriod

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

Note: "earliest possible" means:

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

[SWS_CanTSyn_00238] Decrementing of cyclicCounter

Upstream requirements: RS_TS_20031

⌈If cyclicCounter is greater than 0, CanTSyn shall on each call of CanTSyn_MainFunction decrement the cyclicCounter by CanTSynMainFunctionPeriod.⌋

7.4.1.2.2 Immediate Time Synchronization

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

[SWS_CanTSyn_00117]

Upstream requirements: RS_TS_20031

⌈If `CanTSynImmediateTimeSync` is set to `TRUE` for a Time Base, CanTSyn shall check on each `CanTSyn_MainFunction` call by calling `StbM_GetTimeBaseUpdateCounter`, if the `timeBaseUpdateCounter` of the corresponding Time Base has changed.⌋

[SWS_CanTSyn_00118]

Upstream requirements: RS_TS_20031

⌈If

- the Time Master is in state `TM_Idle`
- and `CanTSynImmediateTimeSync` is set to `TRUE`
- and the `timeBaseUpdateCounter` of a Time Base has changed
- and the `GLOBAL_TIME_BASE` bit of the `timeBaseStatus` is set
- and `syncTxCounter` (if configured) is less than `CanTSynSyncTransmissionsPerCycle`,

CanTSyn shall trigger an immediate transmission of Time Synchronization messages for the corresponding Time Domain.⌋

Note: `timeBaseStatus` can be obtained by `StbM_GetTimeBaseStatus` or `StbM_GetCurrentTime`.

In addition, to the actual trigger condition for an immediate transmission (refer to [SWS_CanTSyn_00118] above) the parameter `CanTSynCyclicMsgResumeTime` needs to be considered for immediate transmission. Refer also to the trigger condition for cyclic Timesync message transmissions (refer to [SWS_CanTSyn_00028]).

Two main scenarios are relevant for configuration of `CanTSynCyclicMsgResumeTime`:

- With `CanTSynCyclicMsgResumeTime` and `CanTSynGlobalTimeTxPeriod` both being configured as zero, a single shot mode is achieved that is solely triggered by the change of the `timeBaseUpdateCounter`.
- With `CanTSynCyclicMsgResumeTime` greater than `CanTSynGlobalTimeTxPeriod` a hold-over scenario in a Time Gateway can be configured:
  - While Timesync messages are received from the Time Master side, the Timesync messages on the sub-busses are only triggered by immediate transmission, as cyclic transmission is suspended while `cyclicMsgResumeCounter` is running.
  - If no Timesync messages from the Time Master side are received anymore and a timeout is detected, cyclic transmission takes over, as cyclic trans-

mission is no longer suspended because cyclicMsgResumeCounter has elapsed.

– If reception of Timesync messages from the Time Master side resumes, the Timesync messages on the sub-busses are again triggered by immediate transmission, as cyclic transmission is again suspended because of the cyclicMsgResumeCounter.

[SWS_CanTSyn_00120]

Upstream requirements: RS_TS_20031

⌈If for a Time Domain

- CanTSynImmediateTimeSync is set to TRUE
- and CanTSynCyclicMsgResumeTime is greater than 0
- and an immediate transmission of a SYNC message has completed successfully, i.e., the Time Master transitisions from state TM_BuildSyncMsg to state TM_DebounceBeforeFupMsg (refer to [1])

then the CanTSyn shall

- set the counter cyclicMsgResumeCounter to CanTSynCyclicMsgResumeTime
- and set the cyclicCounter to 0

for the corresponding Time Domain.⌋

[SWS_CanTSyn_00122]

Upstream requirements: RS_TS_20031

⌈If for a Time Domain cyclicMsgResumeCounter is greater than 0, then the CanTSyn shall discard cyclic Timesync message transmission requests for that Time Domain.⌋

[SWS_CanTSyn_00220]

Upstream requirements: RS_TS_20031

⌈If for a Time Domain cyclicMsgResumeCounter is greater than 0, then the CanTSyn shall decrement upon each invocation of CanTSyn_MainFunction the cyclicMsgResumeCounter of the corresponding Time Domain by CanTSynMainFunctionPeriod.⌋

[SWS_CanTSyn_00121]

Upstream requirements: RS_TS_20031

⌈If the cyclicMsgResumeCounter is decremented to 0 or below, then the CanTSyn shall resume cyclic Timesync message transmission within the same CanTSyn_MainFunction call by requesting either a SYNC message transmission.⌋

Rationale: [SWS_CanTSyn_00121] is to ensure, that the first cyclic transmission is requested in the same main function call in which also `cyclicMsgResumeCounter` reaches 0 (refer to term "earliest possible" main function in [SWS_CanTSyn_00028]). If the message is actually really transmitted depends also on the `debounceCounter`.

7.4.1.3 Debouncing of Time Synchronization Message Transmission

7.4.1.3.1 Debouncing per PDU

The CanTSyn debounces CAN Tx PDUs of a Time Master to avoid bursts of Timesync messages on the bus (e.g. if immediate transmission is enabled). This mechanism also defines the minimum interval between a SYNC and the corresponding FUP message.

For each Tx PDU the CanTSyn maintains a `debounceCounter`. On each transmission of a Timesync message the debounce counter is (re-)loaded by the configured debounce time `CanTSynGlobalTimeDebounceTime`. The counter is decremented in each CanTSyn main cycle. Transmission of the same PDU can only be triggered, if the debounce counter has reached the value 0. Refer also to the overall sequence for the Timesync message transmission in Figure 7.2.

The CanTSyn does not support sharing of PDUs across domains and busses, i.e., the same PDU ID should not be used for different time domains.

However, the CanTSyn supports a separate mechanism to debounce the transmission of SYNC messages across CAN busses (refer to chapter “Debouncing across multiple domains and busses”).

7.4.1.3.2 Debouncing across multiple domains and busses

If SW timestamping is used the precision of the Global Time transmitted by the Time Master is affected by the transmission confirmation ISR delay/jitter. The effect increases if the Time Master transmits SYNC messages on various domains/buses, where the transmit confirmation interrupt is handled by the same core. In this case it is likely that several transmit confirmation interrupts are triggered almost simultaneously. As a result any but the first CanTSyn ISR execution gets delayed by the CanTSyn’s ISR delay of the ’first’ interrupt to be processed and its execution time but also by any other interrupt’s ISR which is then processed directly after the first CanTSyn ISR and before the second one (e.g., due to interrupt priority). To avoid such burst transmissions of SYNC messages, the transmissions will be distributed across CanTSyn main cycles, i.e., the number of simultaneous SYNC message transmissions per CanTSyn main cycle is limited. Refer also to the overall sequence for the Timesync message transmission in Figure 7.2.

[SWS_CanTSyn_00228]

Upstream requirements: RS_TS_20031

⌈If

- the CanTSyn module is configured as Time Master for at least one Time Domain
- and `CanTSynSyncTransmissionsPerCycle` is greater than 0

the CanTSyn module shall maintain a counter `syncTxCounter` that counts the SYNC message transmission requests per transmission SYNC message interval `CanTSynMainFunctionPeriod`.⌋

[SWS_CanTSyn_00229]

Upstream requirements: RS_TS_20031

⌈The CanTSyn module shall set the `syncTxCounter` to 0 at the start of each `CanTSyn_MainFunction`.⌋

[SWS_CanTSyn_00230]

Upstream requirements: RS_TS_20031

⌈The CanTSyn module shall increment the `syncTxCounter` by 1 at each call to `LSduR_CanTSynTransmit` for a SYNC message.⌋

7.4.2 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).

7.4.2.1 Global Time Calculation

For calculation of the Global Time by the Time Master also refer to the protocol specification

- [1] Chapter *"Set SYNC Message Seconds Field (SyncTimeSec)"*,
- [1] Chapter *"Set FUP Message Nanoseconds Field (SyncTimeNSec and OVS)"*.

In addition to the message fields (refer to chapter "Message Format")

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

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

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

Refer to Figure 9.2 and to sequence diagram Figure 9.3 for a better understanding of all steps of the Global Time Calculation sequence of the Time Master as specified in the requirements below.

[SWS_CanTSyn_00149]
Status: DRAFT
Upstream requirements: RS_TS_20035

⌈If a Synchronized Time Domain is in state TM_BuildSyncMsg, i.e., a cyclic or immediate transmission of a SYNC message is requested, the Time Master shall

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

⌋

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

[SWS_CanTSyn_00150]
Status: DRAFT
Upstream requirements: RS_TS_20035, RS_TS_20070

⌈Upon successful SYNC message TX confirmation for a Synchronized Time Domain and if no TX confirmation timeout has occurred (refer to the transition from state TM_TransmitSyncMsg to state TM_DebounceBeforeFupMsg in the Time Master's protocol state machine in [1]), the Time Master shall within the TX confirmation routine (CanTSyn_TxConfirmation)

- if HW timestamping is enabled,

  - Retrieve $T1_{CAN}$ as egress timestamp from CAN controller HW value via `CanIf_GetEgressTimeStamp`

- else
  - Retrieve $T1_{VLT}$ as egress timestamp by reading current Virtual Local Time value via `StbM_GetCurrentVirtualLocalTime`

⌋

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

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

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

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

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

[SWS_CanTSyn_00151]

Status: DRAFT
Upstream requirements: RS_TS_20035

⌈
If

- HW timestamping is disabled,

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

- calculate T4 = $T0_{SYNC_ns}$ + ($T1_{VLT}$ - $T0_{VLT}$)

⌋

[SWS_CanTSyn_00152]

Status: DRAFT
Upstream requirements: RS_TS_20035, RS_TS_20070

⌈
If

- HW timestamping is enabled and

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

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

- calculate T4 = $T0_{SYNC_ns}$ + $T1_{VLT}$ - $T0_{VLT}$,
  with $T1_{VLT}$ = $T1_{CAN}$

⌋

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

[SWS_CanTSyn_00153]

Status: DRAFT
Upstream requirements: RS_TS_20035, RS_TS_20070

⌈
If

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

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

1. establish a protection against interruptions
2. read $T_{currentTime_CAN}$ via `CanIf_GetCurrentTime`,
3. read $T1_{VLT}$ via `StbM_GetCurrentVirtualLocalTime`,
4. release the protection against interruptions and
5. calculate T4 = $T0_{SYNC_ns}$ + ($T1_{VLT}$ - $T0_{VLT}$) - ($T_{currentTime_CAN}$ - $T1_{CAN}$)

⌋

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

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

[SWS_CanTSyn_00154]
Status: DRAFT
Upstream requirements: RS_TS_20035

⌈If for a Synchronized Time Domain a FUP message is due (refer to [FO_PRS_CanTSyn_10047] of [1]), the Time Master shall

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

⌋

7.4.2.2 User Data

[SWS_CanTSyn_00014]
Upstream requirements: RS_TS_20037

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

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

7.4.2.3 OVS Calculation

Refer also to 6.4.2.6.4 "Calculate FUP Message Nanoseconds Field (SyncTimeNSec and OVS)" in [1] for how to set the `OVS` field of the FUP message.

[SWS_CanTSyn_00047]
Upstream requirements: RS_TS_20035

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

7.4.2.4 SGW Calculation

Refer also to the protocol specification [1] Chapter *"Set FUP Message Time Gateway Synchronization Status Field (SGW)"* for how to set the `SGW` field of the FUP message.

[SWS_CanTSyn_00030]

Upstream requirements: RS_TS_20035

⌈The SGW value (Time Gateway synchronization status) shall be retrieved from the Time Base synchronization status. If the SYNC_TO_GATEWAY bit within timeBase Status is not set the SGW value shall be SyncToGTM. Otherwise the SGW value shall be set to SyncToSubDomain.⌋

7.4.2.5 Sequence Counter Calculation

Refer also to the protocol specification [1] Chapter *"Set SYNC Message Sequence Counter Field (SC)"* and [1] Chapter *"Set FUP Message Sequence Counter Field (SC)"* for the specification of the Sequence Counter (SC) field calculation for the SYNC and FUP messages, respectively.

7.4.2.6 CRC Calculation

Refer also to the protocol specification [1] Chapter *"Set SYNC Message Checksum Field (CRC)"* and [1] Chapter *"Set FUP Message Checksum Field (CRC)"* for the specification of the calculation of the CRC field of the SYNC and FUP messages, respectively.

[SWS_CanTSyn_00050]

Upstream requirements: RS_TS_20033, RS_TS_20035

⌈The function Crc_CalculateCRC8H2F as defined in [7] shall be used to calculate the CRC if configured.⌋

7.4.2.7 ICV Generation

Refer also to the protocol specification [1] Chapter *"Set Freshness Value Field (FV and FVL)"* and [1] Chapter *"Set Integrity Check Value Fields (ICV and ICVL)"* for how the calculation of the Freshness Value and ICV integrates with protocol state machine of the Time Master.

[SWS_CanTSyn_00161]

Upstream requirements: RS_TS_20073

⌈If in state state TM_BuildFupMsg for the Time Doamin the CanTSyn shall calculate the ICV using

- the SYNC payload (byte 0 .. 7),
- the FUP payload (byte 0 .. 9)
- and the FV (if configured by CanTSynIcvGenerationFvIdRef).

⌋

Refer to the SWS StbM [6] Chapter *"Freshness Value"* for the configuration details of FV referenced in each Time Domain.

[SWS_CanTSyn_00163]

Upstream requirements: RS_TS_20073

⌈When the FV is referenced (refer CanTSynIcvGenerationFvIdRef) and the configured truncated FV length (StbMFreshnessValueTruncLength) == FV length (StbMFreshnessValueLength) in StbM, the Time Master shall call the StbM_GetTxFreshness Api to obtain the FV by using the StbMFreshnessValueId.⌋

[SWS_CanTSyn_00164]

Upstream requirements: RS_TS_20073

⌈When the FV is referenced (refer CanTSynIcvGenerationFvIdRef) and the configured truncated FV length (StbMFreshnessValueTruncLength) < FV length (StbMFreshnessValueLength), the Time Master shall call the StbM_GetTxFreshnessTruncData Api to obtain the FV and the truncated FV by using the StbMFreshnessValueId.⌋

[SWS_CanTSyn_00165]

Upstream requirements: RS_TS_20073

⌈When the FV is not referenced (refer CanTSynIcvGenerationFvIdRef), the Time Master shall not include the FV in the ICV generation and fill 0 for FVL field in message types 0x78 and 0x88.⌋

[SWS_CanTSyn_00166]

Upstream requirements: RS_TS_20073

⌈If StbM_GetTxFreshness returns E_OK, the Time Master shall construct the message types 0x78 and 0x88 with the full FV, set the FVL to StbMFreshnessValueLength and use the full FV in ICV generation.⌋

[SWS_CanTSyn_00167]

Upstream requirements: RS_TS_20073

⌈If StbM_GetTxFreshnessTruncData returns E_OK, the Time Master shall construct the message types 0x78 or 0x88 with truncated FV, set the FVL to StbMFreshnessValueTruncLength and use the full FV in ICV generation.⌋

[SWS_CanTSyn_00168]

Upstream requirements: RS_TS_20073

⌈If StbM_GetTxFreshness or StbM_GetTxFreshnessTruncData return a non-recoverable error code (i.e, E_NOT_OK), the Time Master shall:

- stop the ICV generation and accordingly fill 0 for FVL and ICVL fields in message types 0x78 and 0x88,

- call Det_ReportRuntimeError with the parameter ErrorId set to CANTSYN_E_FRESHNESSFAILURE,
- and report the security event SEV_TSYN_CAN_FRESHNESS_NOT_AVAILABLE to the IdsM.

⌋

Refer to the chapter 10.2.5 in [8] for the configuration details of CSM job used for ICV generation.

[SWS_CanTSyn_00169]

Upstream requirements: RS_TS_20073

⌈If CanTSynIcvGenerationBase for the Time Domain is configured to ICV_MAC, the Time Master shall call Csm_MacGenerate to generate the ICV value.⌋

[SWS_CanTSyn_00170]

Upstream requirements: RS_TS_20073

⌈If CanTSynIcvGenerationBase for the Time Domain is configured to ICV_SIGNATURE, the Time Master shall call Csm_SignatureGenerate to generate the ICV value.⌋

Note: The mode parameter is intentionally left open for the implementer to choose (i.e. CRYPTO_OPERATIONMODE_SINGLECALL would possibly be the best option since it does not require further calls to CSM).

The CSM job used to generate the ICV can be configured to synchronous or asynchronous behaviour.

[SWS_CanTSyn_00171]

Upstream requirements: RS_TS_20073

⌈If the CSM job used to generate ICV is configured in synchronous behaviour, the Time Master shall disable ICV generation timeout monitoring.⌋

[SWS_CanTSyn_00172]

Upstream requirements: RS_TS_20073

⌈If the CanTSynIcvGenerationTimeout is set to 0, the Time Master shall not do the ICV generation timeout monitoring.⌋

[SWS_CanTSyn_00173]

Upstream requirements: RS_TS_20073

⌈If CanTSynIcvGenerationTimeout is set to any value > 0 and Csm_MacGenerate or Csm_SignatureGenerate returns E_OK, the Time Master shall start the CanTSynIcvGenerationTimeout.⌋

[SWS_CanTSyn_00174]

Upstream requirements: RS_TS_20073

⌈If CanTSynIcvGenerationTimeout is set to any value > 0 and the CanTSyn_IcvGenerationIndication callback is called, the Time Master shall stop the running CanTSynIcvGenerationTimeout.⌋

[SWS_CanTSyn_00175]

Upstream requirements: RS_TS_20073

⌈If one of the following conditions is true:

- the authentication build counter has reached the configuration value CanTSynTxAuthenticationBuildAttempts,
- the verification of the ICV has returned a non-recoverable error such as returning E_NOT_OK or KEY_FAILURE,
- or CanTSynIcvGenerationTimeout expires before the notification of the CanTSyn_IcvGenerationIndication callback,

then the Time Master shall

- stop the ICV generation and accordingly fill 0 for ICVL field in message types 0x78 and 0x88,
- and report the security event SEV_TSYN_CAN_ICV_GENERATION_FAILED to the IdsM.

⌋

[SWS_CanTSyn_00176]

Upstream requirements: RS_TS_20073

⌈With the notification of the CanTSyn_IcvGenerationIndication callback and successful generation of ICV, the Time Master shall add the generated ICV to the ICV field in message types 0x78 and 0x88 and transmit the FUP.⌋

[SWS_CanTSyn_00177]

Upstream requirements: RS_TS_20073

⌈If the FV is referenced (refer CanTSynIcvGenerationFvIdRef), then the Time Master shall notify the successful transmission of messages of type 0x78 and 0x88 to FVM by calling StbM_SPduTxConfirmation.⌋

[SWS_CanTSyn_00210]

Status: DRAFT
Upstream requirements: RS_TS_20073

⌈For every transmission of messages of type 0x78 and 0x88 the Time Master shall maintain an authentication build counter (refer CanTSynTxAuthenticationBuildAttempts).⌋

[SWS_CanTSyn_00211]
Status: DRAFT
Upstream requirements: RS_TS_20073

⌈Upon the initial processing of messages of type `0x78` and `0x88` the Time Master shall set the authentication build counter to 0.⌋

[SWS_CanTSyn_00212]
Status: DRAFT
Upstream requirements: RS_TS_20073

⌈When `StbM_GetTxFreshness` or `StbM_GetTxFreshnessTruncData` return a recoverable error code (e.g., `STBM_E_BUSY`), then the Time Master shall increment the authentication build counter.⌋

[SWS_CanTSyn_00213]
Status: DRAFT
Upstream requirements: RS_TS_20073

⌈When `Csm_MacGenerate` or `Csm_SignatureGenerate` return a recoverable error code (e.g., `E_BUSY`, `QUEUE_FULL`), then the Time Master shall increment the authentication build counter.⌋

[SWS_CanTSyn_00214]
Status: DRAFT
Upstream requirements: RS_TS_20073

⌈If

- building the authenticated message has failed
- and the authentication build counter has not yet reached the configuration value `CanTSynTxAuthenticationBuildAttempts`,

then the Time Master shall retry the freshness attempt and ICV calculation in the next call of `CanTSyn_MainFunction`.⌋

7.4.2.8 Message Assembling

Refer to the protocol specification [1] Chapter *"State TM_BuildSyncMsg"* and [1] Chapter *"State TM_BuildFupMsg"* for how to build the SYNC and FUP messages.

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

For the overall behavior of the Time Slave refer to the protocol state machine in the protocol specification [1] Chapter *"Time Slave"*.

The sections below specify details how the protocol state machine is realized in the CanTSyn module.

[SWS_CanTSyn_00182]

Upstream requirements: RS_TS_20073

⌈If

- for a Time Domain Security Event reporting is enabled (`CanTSynEnableSecurityEventReporting` is `true`)
- and that Time Domain is in state `TS_WaitForFupMsg`,

when a SYNC message is received, then the Time Slave shall report the security event `SEV_TSYN_CAN_MSG_SEQUENCE_ERROR` to the `IdsM`.⌋

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

[SWS_CanTSyn_00207]

Upstream requirements: RS_TS_20073

⌈If

- for a Time Domain Security Event reporting is enabled (`CanTSynEnableSecurityEventReporting` is `true`)
- and for that Time Domain any SYNC or FUP message is received within `CanTSynGlobalTimeRxDebounceTime` after the previous Timesync message (i.e., while the Time Domain is in state `TS_DebounceBeforeFupMsg` or state `TS_DebounceBeforeSyncMsg`),

then the Time Slave shall report the security event `SEV_TSYN_CAN_MSG_SEQUENCE_ERROR` to the `IdsM`.⌋

7.5.2 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).

7.5.2.1 Global Time Calculation

For calculation of the Global Time by the Time Slave also refer to the protocol specification [1] Chapter *"Update of Time Slave"*.

In addition to the message fields (refer to chapter "Message Format")

- SyncTimeSec
- OVS and
- SyncTimeNSec,

which are actually received by the Time Slave on the bus from the Time Master, this chapter defines and uses the following internal variables for calculation of the Rx Time Tuple for Synchronized Time Domains:

- T0: Global Time (seconds portion) received from Time Master in SYNC message
- $T1_{VLT}$: Ingress timestamp of SYNC message as captured by HW in the CAN controller or by SW in CanTSyn_RxIndication.
- TG_{Rx} : Global Time component of the Rx Time Tuple.
- $T2_{VLT}$: Virtual Local Time component of the Rx Time Tuple (equivalent to TV_Rx in the StbM).
- $T3_{VLT}$: Current time read out from CAN controller hardware - used for correlation of StbM time and CAN HW clock.
- $T5_{VLT}$: Current virtual local time in StbM - used for correlation of StbM local time and CAN HW clock.
- T4: Correction value for T0 as received from the Time Master. It is calculated from values of OVS and SyncTimeNSec field in the FUP message.
- T_{SRD}: SYNC reception delay as difference between $T3_{VLT}$ and $T1_{VLT}$.
- T_{PD}: SYNC propagation delay as configured by CanTSynGlobalTime-PropagationDelay

Refer to

- the sequence diagrams in Figure 9.4 and Figure 9.5
- and the flow chart for evaluating Timesync messages in Figure 7.3 below

for a better understanding of all steps of the Global Time calculation sequence of the Time Slave as specified in the requirements below.

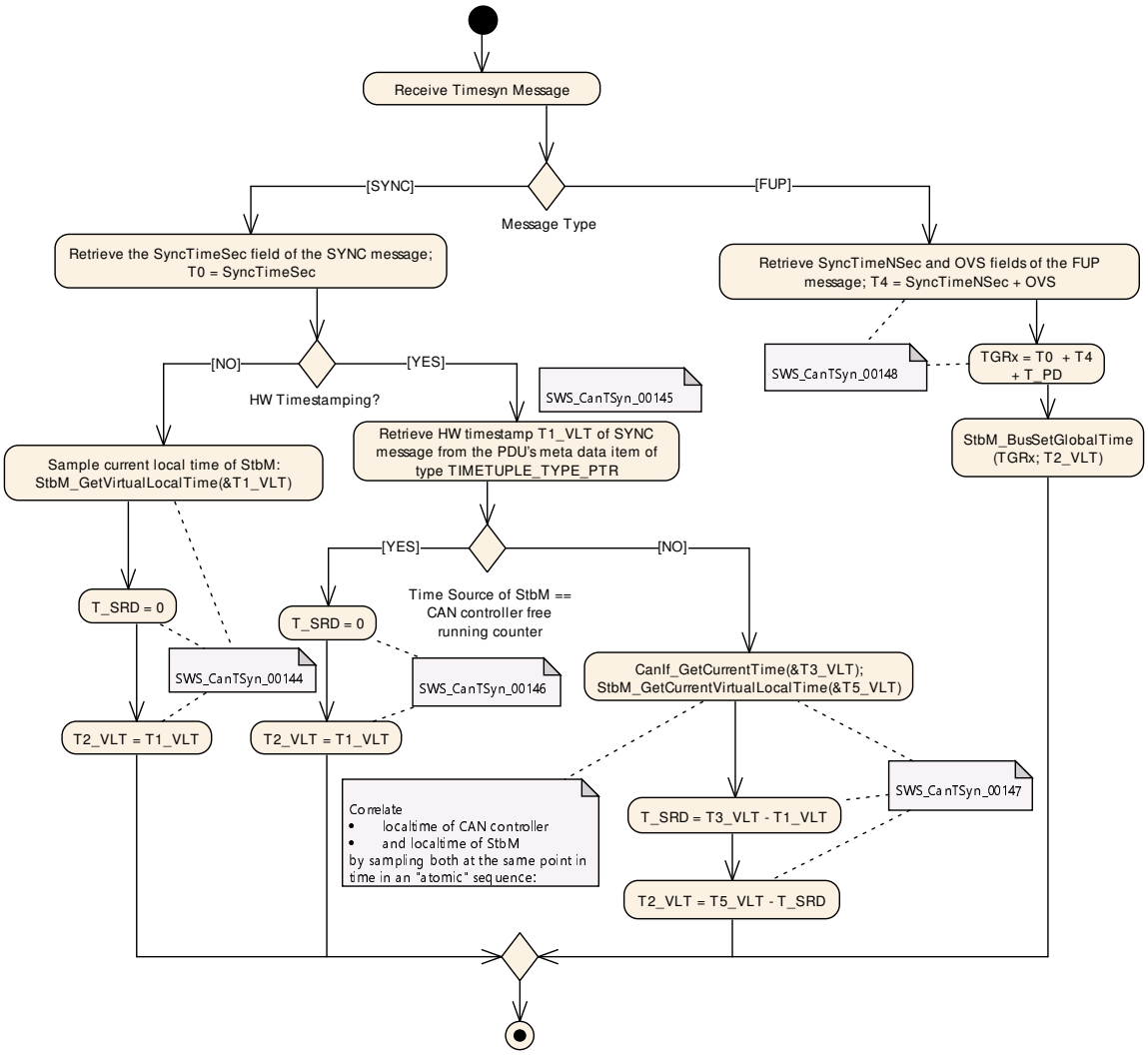


Figure 7.3: Evaluate Timesync message

[SWS_CanTSyn_00144]

Status: DRAFT
Upstream requirements: RS_TS_20035, RS_TS_20070

⌈For a Time Slave, on invocation of `CanTSyn_RxIndication` for a SYNC message, and if `CanTSynHardwareTimestampSupport` is set to `FALSE`, CanTSyn shall

1. immediately establish a protection against interruption
2. and directly afterwards retrieve the reference time $T1_{VLT}$ for the SYNC message via `StbM_GetCurrentVirtualLocalTime` from the StbM

   Note: Once $T1_{VLT}$ has been retrieved, protection against interruptions may be released

3. set the $T2_{VLT}$ part of the Rx Time Tuple to the value of $T1_{VLT}$ (i.e., $T2_{VLT} = T1_{VLT}$)
4. set the SYNC reception delay T_{SRD} to 0

5. retrieve T0 from the `SyncTimeSec` field of the SYNC message

⌋

Note: Immediate protection against interruptions means that there shall be no frame checks before (if called in context of the RX interrupt with interrupt nesting disabled, interrupt protection 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 $T1_{VLT}$. Once $T1_{VLT}$ has been sampled it is ok to remove the protection against interruptions and to make the necessary validations. This means that $T1_{VLT}$ will be taken even if the succeeding validations fail and thus making the snapshot superfluous.

[SWS_CanTSyn_00145]
Status: DRAFT
Upstream requirements: RS_TS_20035

⌈For a Time Slave, on invocation of `CanTSyn_RxIndication` for a SYNC message, and if `CanTSynHardwareTimestampSupport` is set to `TRUE`, CanTSyn shall

- retrieve $T1_{VLT}$ from the meta data item of type `TIMETUPLE_TYPE_PTR` of the PDU identified by parameter `RxPduId`
- convert $T1_{VLT}$ to a representation in ns
- retrieve T0 from the `SyncTimeSec` field of the SYNC message

⌋

[SWS_CanTSyn_00146]
Status: DRAFT
Upstream requirements: RS_TS_20035

⌈For a Time Slave, on invocation of `CanTSyn_RxIndication` for a SYNC message

- and if `CanTSynHardwareTimestampSupport` is set to `TRUE`
- and if the StbM uses the CAN controller hardware counter as Virtual Local Time source for the Time Base (refer to `StbMLocalTimeClock`),

CanTSyn shall

- set $T2_{VLT}$ part of the Rx Time Tuple to the value of $T1_{VLT}$ (i.e., $T2_{VLT} = T1_{VLT}$)
- and set the SYNC reception delay T_{SRD} to 0.

⌋

[SWS_CanTSyn_00147]
Status: DRAFT
Upstream requirements: RS_TS_20035, RS_TS_20070

⌈For a Time Slave, on invocation of `CanTSyn_RxIndication` for a SYNC message,

- and if CanTSynHardwareTimestampSupport is set to TRUE
- and if the StbM does not use the CAN controller hardware counter as Virtual Local Time source for the Time Base (refer to StbMLocalTimeClock)

CanTSyn shall correlate the CAN HW time and the Virtual Local Time of the StbM by applying the following sequence:

1. protect the following two steps against interruptions
2. retrieve the current time of the CAN controller hardware counter via CanIf_GetCurrentTime and convert it to the Virtual Local Time $T3_{VLT}$,
3. retrieve the current value of the Virtual Local Time of the corresponding Time Base in the StbM via StbM_GetCurrentVirtualLocalTime as $T5_{VLT}$,
4. calculate the SYNC reception delay T_{SRD} as ($T3_{VLT}$ - $T1_{VLT}$)
5. $T2_{VLT}$ shall be calculated as $T5_{VLT}$ - T_{SRD}

⌋

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

[SWS_CanTSyn_00148]

Upstream requirements: RS_TS_20035

⌈For a Time Slave, on invocation of CanTSyn_RxIndication for a FUP message, CanTSyn shall

1. retrieve the following data from the FUP message
   - the OVS field
   - and the SyncTimeNSec field
2. and calculate T4 = OVS + SyncTimeNSec

Either in the same Rx indication routine (CanTSyn_RxIndication) or in the subsequent CanTSyn_MainFunction invocation CanTSyn shall

1. calculate TG_{Rx} as (T0 + T4 + T_{PD}).
2. and forward the new Rx Time Tuple [TG_{Rx}; $T2_{VLT}$] to the StbM via StbM_BusSetGlobalTime

⌋

Note: In the above sequence protection against interruptions is important, because any interruption of the sequence of step 2 and step 3 would worsen the precision of the local instance of the Global Time, which depends on time tuple [TG_{Rx};$T2_{VLT}$].

Note: TG_{Rx} is a point in time in the past, i.e., at SYNC ingress. When StbM_BusSetGlobalTime is called, the StbM is able to derive from that the current Local Time of the Time Slave.

7.5.2.2 OVS Consideration

Refer to the protocol specification [1] Chapter *"Check FUP Message Nanoseconds Overflow Field (OVS)"* and [1] Chapter *"Update Local Time (from SyncTimeNSec and OVS) Fields"* for processing the OVS field of the FUP message.

7.5.2.3 SGW Calculation

Refer also to the protocol specification [1] Chapter *"Update Time Gateway Synchronization Status (from SGW) Field"* for processing the SGW field of the FUP message.

[SWS_CanTSyn_00133]

Upstream requirements: RS_TS_20032, RS_TS_20034

⌈If the SGW value (in the FUP message) is set to SyncToSubDomain, the SYNC_TO_GATEWAY bit within timeBaseStatus shall be set to TRUE. Otherwise, it shall be set to FALSE.⌋

7.5.2.4 Sequence Counter Validation

Refer to the protocol specification [1] Chapter *"Check SYNC Message Sequence Counter Field (SC)"* and [1] Chapter *"Check FUP Message Sequence Counter Field (SC)"* for the details of sequence counter check.

7.5.2.5 CRC Validation

Refer to the protocol specification [1] Chapter *"Check SYNC Message CRC Field (CRC)"* and [1] Chapter *"Check FUP Message CRC Field (CRC)"* for the specification of the CRC check.

[SWS_CanTSyn_00080]

Upstream requirements: RS_TS_20034, RS_TS_20035

⌈The function Crc_CalculateCRC8H2F as defined in [7] shall be used to validate the CRC if configured.⌋

7.5.2.6 ICV Verification

Refer to the protocol specification [1] Chapter *"ICV Secured FUP Messages"* for how calculation of the Freshness Value and ICV integrates with protocol state machine of the Time Slave.

The requirements in this chapter refine the protocol specification by

- adding a retry strategy when deriving the Freshness Value or ICV value
- allowing truncation of the ICV
- adding Security Events
- and defining the APIs for deriving the Freshness Value or ICV value

Refer to the chapter 7.3.13 in the SWS StbM [6] for the configuration details of `FV` referenced in each Time Domain.

[SWS_CanTSyn_00188]

Upstream requirements: RS_TS_20073

⌈When

- the `FV` is referenced (refer `CanTSynIcvVerificationFvIdRef`)
- and `FVL` > 0,

in a received FUP message,
then the Time Slave shall call the `StbM_GetRxFreshness` Api to obtain the Freshness Value by using

- the `StbMFreshnessValueId` from the reference `CanTSynIcvVerificationFvIdRef`
- the `StbMTruncatedFreshnessValue` as received in the `FV` field of the FUP message
- the `StbMTruncatedFreshnessValueLength` as received in the FVL field of the FUP message
- the `StbMAuthVerifyAttempts` as the number of failed verification attempts for the current message (ICV verification attempt counter)
- the `StbMFreshnessValueLength` from the reference `CanTSynIcvVerificationFvIdRef`

⌋

[SWS_CanTSyn_00189]

Upstream requirements: RS_TS_20073

⌈When the `FVL` is 0 in the received FUP, then the Time Slave shall not include the `FV` in the ICV verification.⌋

[SWS_CanTSyn_00190]

Upstream requirements: RS_TS_20073

⌈When

- the FV is not referenced (refer CanTSynIcvVerificationFvIdRef)
- and FVL > 0,

in the received FUP,
then the Time Slave shall stop the ICV verification and consider ICV verification as failed.⌋

[SWS_CanTSyn_00191]

Upstream requirements: RS_TS_20073

⌈If StbM_GetRxFreshness returns E_OK, the Time Slave shall use the FV in ICV verification.⌋

[SWS_CanTSyn_00193]

Upstream requirements: RS_TS_20073

⌈If StbM_GetRxFreshness returned a non-recoverable error code (i.e, E_NOT_OK), then the Time Slave shall

- consider the ICV verification of a received FUP message as failed
- stop the ICV verification,
- drop the received FUP message,
- call Det_ReportRuntimeError with the parameter ErrorId set to CANTSYN_E_FRESHNESSFAILURE,
- and report the security event SEV_TSYN_CAN_FRESHNESS_NOT_AVAILABLE to the IdsM.

⌋

Refer to the chapter 10.2.5 in [8] for the configuration details of CSM job used for ICV verification.

[SWS_CanTSyn_00194]

Upstream requirements: RS_TS_20073

⌈If CanTSynIcvVerificationBase for the Time Domain is configured to ICV_MAC, the Time Slave shall call Csm_MacVerify to verify the ICV value.⌋

[SWS_CanTSyn_00195]

Upstream requirements: RS_TS_20073

⌈If CanTSynIcvVerificationBase for the Time Domain is configured to ICV_SIGNATURE, the Time Slave shall call Csm_SignatureVerify to verify the ICV value.⌋

Note: The mode parameter is intentionally left open for the implementer to choose (i.e. CRYPTO_OPERATIONMODE_SINGLECALL would possibly be the best option since it does not require further calls to CSM).

The CSM job used to generate the ICV can be configured to synchronous or asynchronous behaviour.

[SWS_CanTSyn_00196]

Upstream requirements: RS_TS_20073

⌈The ICV verification timeout observation is disabled, when the CSM job to verify ICV is configured in synchronous behaviour. In this case, the CanTSynIcvVerificationTimeout shall be set to 0.⌋

[SWS_CanTSyn_00197]

Upstream requirements: RS_TS_20073

⌈If Csm_MacVerify or Csm_SignatureVerify returns E_OK, the Time Slave shall start the CanTSynIcvVerificationTimeout.⌋

[SWS_CanTSyn_00198]

Upstream requirements: RS_TS_20073

⌈The CanTSynIcvVerificationTimeout shall be stopped with the notification of the CanTSyn_IcvVerificationIndication callback.⌋

[SWS_CanTSyn_00199]

Upstream requirements: RS_TS_20073

⌈When Csm_MacVerify or Csm_SignatureVerify return a recoverable error code (e.g., E_BUSY, QUEUE_FULL), then the Time Slave shall

- consider the verification of received FUP message as failed
- and increment the authentication build counter for this FUP message.

⌋

[SWS_CanTSyn_00200]

Upstream requirements: RS_TS_20073

⌈If one of the following conditions is true:

- the authentication build counter has reached the configuration value CanTSynRxAuthenticationBuildAttempts,

- the ICV verification attempt counter has reached the configuration value CanTSynIcvVerificationAttempts
- the verification of the ICV has returned a non-recoverable error such as returning E_NOT_OK or KEY_FAILURE,
- CanTSynIcvVerificationTimeout expires before the notification of the CanTSyn_IcvVerificationIndication callback,

then the Time Slave shall

- stop the ICV verification and consider the ICV verification as failed,
- and report the security event SEV_TSYN_CAN_ICV_VERIFICATION_FAILED to the IdsM.

⌋

[SWS_CanTSyn_00215]
Status: DRAFT
Upstream requirements: RS_TS_20073

⌈For every reception of messages that require ICV verification, the Time Slave shall maintain an authentication build counter (refer to CanTSynRxAuthenticationBuildAttempts).⌋

[SWS_CanTSyn_00216]
Status: DRAFT
Upstream requirements: RS_TS_20073

⌈Upon the initial processing of messages that require ICV verification (i.e., upon the first attempt of a freshness and ICV check for each received message), the Time Slave shall set the authentication build counter to 0.⌋

[SWS_CanTSyn_00217]
Status: DRAFT
Upstream requirements: RS_TS_20073

⌈When StbM_GetRxFreshness returns a recoverable error code (e.g., STBM_E_BUSY), then the Time Slave shall increment the authentication build counter and shall not execute verification of the ICV.⌋

[SWS_CanTSyn_00218]
Status: DRAFT
Upstream requirements: RS_TS_20073

⌈If

- the verification of the authenticated message has failed
- and the authentication build counter has not yet reached the configuration value CanTSynRxAuthenticationBuildAttempts,

then the Time Slave shall retry the freshness attempt and ICV verification in the next call of `CanTSyn_MainFunction`.⌋

[SWS_CanTSyn_00219]
Status: DRAFT
Upstream requirements: RS_TS_20073

⌈If the verification of the ICV could be successfully executed but the verification failed (e.g. the MAC verification has failed or the key was invalid), then the Time Slave shall

- increment the ICV verification attempt counter
- and set the authentication build counter to 0.

⌋

Note: Resetting the authentication build counter will prevent dropping the authentication process too early even though ICV verification attempts are still possible.

7.5.2.7 Message Disassembling

Refer to the protocol specification [1] Chapter *"State TS ReceiveSyncMsg"* and [1] Chapter *"State TS_ReceiveFupMsg"* for the overall flow of the message verification process of a SYNC or FUP message on Time Slave side.

[SWS_CanTSyn_00064]
Upstream requirements: RS_TS_20032, RS_TS_20034

⌈For a valid pair of SYNC and FUP messages with successfully validated set of values `SyncTimeSec`, `OVS` and `SyncTimeNSec` a new Rx Time Tuple [TG_{Rx};$T2_{VLT}$] (refer to [6]), consisting of the Global Time value and the associated value of the Virtual Local Time, shall be calculated (refer to [SWS_CanTSyn_00146], [SWS_CanTSyn_00147], [SWS_CanTSyn_00148]) and forwarded to the `StbM` module via `StbM_BusSetGlobalTime`.⌋

[SWS_CanTSyn_00115]
Upstream requirements: RS_TS_20034

⌈On an invocation of `StbM_BusSetGlobalTime` the parameter `pathDelay` of the `measureDataPtr` structure shall be set to the value of the propagation delay as configured by `CanTSynGlobalTimePropagationDelay`.⌋

7.6 Time Recording

7.6.1 Global Time Precision Measurement

The CanTSyn provides all relevant information for Global Time Precision Measurement via `StbM_BusSetGlobalTime` to the StbM.

7.6.2 Time Validation

[SWS_CanTSyn_00137]

Upstream requirements: RS_TS_00034

⌈The `CanTSyn` shall support Time Validation, if `CanTSynTimeValidationSupport` set to `TRUE`.⌋

[SWS_CanTSyn_00138]

Upstream requirements: RS_TS_00034

⌈
If

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

`CanTSyn` shall do time recording for Time Validation for that Time Domain
⌋

[SWS_CanTSyn_00139]

Upstream requirements: RS_TS_00034

⌈
If

- time recording for Time Validation is enabled for a Time Domain (refer to [SWS_CanTSyn_00137] and [SWS_CanTSyn_00138])
- and `CanTSyn` is configured as Time Slave for that Time Domain,

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

`StbM_CanSetSlaveTimingData` shall be called after `StbM_BusSetGlobalTime`.
⌋

Note: `StbM_BusSetGlobalTime` shall be called first, because it updates the Synclocal Time Tuple (refer to [6]), which is required by `StbM_CanSetSlaveTimingData`.

[SWS_CanTSyn_00140]

Upstream requirements: RS_TS_00034

⌈Upon invocation of `StbM_CanSetSlaveTimingData` `CanTSyn` shall pass following values

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

to the function by the parameter `measureDataPtr`.

Struct members

- `measureDataPtr→referenceLocalTimestamp` and
- `measureDataPtr→referenceGlobalTimestamp`

shall be passed as 0.
⌋

Note: The `CanTSyn` passes 0 to avoid undefined values. The structure members `referenceLocalTimestamp` and `referenceGlobalTimestamp` will be set by the `StbM` via `StbM_CanSetSlaveTimingData` internally (refer to [SWS_StbM_00471] in [6]).

[SWS_CanTSyn_00141]

Upstream requirements: RS_TS_00034

⌈
If

- time recording for Time Validation is enabled for a Time Domain (refer to [SWS_CanTSyn_00137] and [SWS_CanTSyn_00138])
- and `CanTSyn` is configured as Time Master for that Time Domain

`CanTSyn` shall call `StbM_CanSetMasterTimingData` upon successful transmission of a SYNC message).
⌋

[SWS_CanTSyn_00142]

Upstream requirements: RS_TS_00034

⌈Upon invocation of `StbM_CanSetMasterTimingData` `CanTSyn` shall pass the following data

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

to the function by the parameter `measureDataPtr`.
⌋

7.7 Security Events

[SWS_CanTSyn_00201]

Upstream requirements: SRS_BSW_00489

⌈If security event reporting has been enabled for the CanTSyn module (`CanTSynEnableSecurityEventReporting` = TRUE) the respective security events shall be reported to the IdsM [9] via the interfaces defined in BSWGeneral [4].⌋

The following table lists the security events which are standardized for the CanTSyn together with their trigger conditions.

[SWS_CanTSyn_00204] Security events for CanTSyn

Upstream requirements: SRS_BSW_00488, SRS_BSW_00489

⌈

Name	*Description*	*ID*
`SEV_TSYN_CAN_ICV_GENERATION_FAILED`	ICV generation for a FUP message has failed.	66
`SEV_TSYN_CAN_ICV_VERIFICATION_FAILED`	ICV verification of a FUP message has failed.	67
`SEV_TSYN_CAN_FRESHNESS_NOT_AVAILABLE`	Failed to get freshness value from FvM.	68
`SEV_TSYN_CAN_MSG_SEQUENCE_ERROR`	Failed to receive correct sequence of SYNC and FUP from the TimeMaster within (CanTSynGlobalTimeFollowUp Timeout).	69

⌋

The following tables specify the context data which shall be reported for the respective security events.

[SWS_CanTSyn_92000] Security event context data definition: SEV_TSYN_CAN_ICV_GENERATION_FAILED

Upstream requirements: SRS_BSW_00488, SRS_BSW_00489

⌈

SEV Name	***SEV_TSYN_CAN_ICV_GENERATION_FAILED***	
ID	66	
Description	ICV generation for a FUP message has failed.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
GlobalTimeDomainId	uint8	

⌋

[SWS_CanTSyn_92001] Security event context data definition: SEV_TSYN_CAN_ICV_VERIFICATION_FAILED

Upstream requirements: SRS_BSW_00488, SRS_BSW_00489

⌈

SEV Name	***SEV_TSYN_CAN_ICV_VERIFICATION_FAILED***	
ID	67	
Description	ICV verification of a FUP message has failed.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
GlobalTimeDomainId	uint8	

⌋

[SWS_CanTSyn_92002] Security event context data definition: SEV_TSYN_CAN_FRESHNESS_NOT_AVAILABLE

Upstream requirements: SRS_BSW_00488, SRS_BSW_00489

⌈

SEV Name	***SEV_TSYN_CAN_FRESHNESS_NOT_AVAILABLE***	
ID	68	
Description	Failed to get freshness value from FvM.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
GlobalTimeDomainId	uint8	

⌋

[SWS_CanTSyn_92003] Security event context data definition: SEV_TSYN_CAN_MSG_SEQUENCE_ERROR

Upstream requirements: SRS_BSW_00488, SRS_BSW_00489

⌈

SEV Name	*SEV_TSYN_CAN_MSG_SEQUENCE_ERROR*	
ID	69	
Description	Failed to receive correct sequence of SYNC and FUP from the TimeMaster within (CanTSyn GlobalTimeFollowUpTimeout).	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
GlobalTimeDomainId	uint8	

⌋

7.8 Error Classification

Section 7.2 "Error Handling" of the document "General Specification of Basic Software Modules" [4] describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme for the error types that may occur in BSW modules.

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

7.8.1 Development Errors

[SWS_CanTSyn_00089] Definition of development errors in module CanTSyn

Upstream requirements: SRS_BSW_00385

⌈

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

⌋

7.8.2 Runtime Errors

[SWS_CanTSyn_91001] Definition of runtime errors in module CanTSyn

Upstream requirements: SRS_BSW_00385

⌈

Type of error	*Related error code*	*Error value*
No FV available from the FVM	CANTSYN_E_FRESHNESSFAILURE	0x01

⌋

7.8.3 Production Errors

There are no production errors.

8 API specification

8.1 Imported types

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

[SWS_CanTSyn_00090] Definition of imported datatypes of module CanTSyn

Upstream requirements: RS_TS_20035

⌈

Module	*Header File*	*Imported Type*
Can	Can_GeneralTypes.h	Can_TimeStampType (draft)
Comtype	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
	ComStack_Types.h	RateDeviationStatusType (draft)
	ComStack_Types.h	RateDeviationWithStatusType (draft)
Csm	Rte_Csm_Type.h	Crypto_OperationModeType
	Rte_Csm_Type.h	Crypto_ResultType
	Rte_Csm_Type.h	Crypto_VerifyResultType
IdsM	IdsM.h	IdsM_SecurityEventIdType
	Rte_IdsM_Type.h	IdsM_TimestampDataType
StbM	Rte_StbM_Type.h	StbM_CanTimeMasterMeasurementType
	Rte_StbM_Type.h	StbM_CanTimeSlaveMeasurementType
	Rte_StbM_Type.h	StbM_SynchronizedTimeBaseType
	Rte_StbM_Type.h	StbM_TimeBaseStatusType
	Rte_StbM_Type.h	StbM_TimeStampShortType
	Rte_StbM_Type.h	StbM_TimeStampType
	Rte_StbM_Type.h	StbM_TimeTupleType
	Rte_StbM_Type.h	StbM_UserDataType
	StbM.h	StbM_MeasurementType
	StbM.h	StbM_VirtualLocalTimeType
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type definitions

8.2.1 CanTSyn_ConfigType

[SWS_CanTSyn_00091] Definition of datatype CanTSyn_ConfigType

Upstream requirements: RS_TS_20035

⌈

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

⌋

8.2.2 CanTSyn_TransmissionModeType

[SWS_CanTSyn_00092] Definition of datatype CanTSyn_TransmissionModeType

Upstream requirements: RS_TS_20035

⌈

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

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 CanTSyn_Init

[SWS_CanTSyn_00093] Definition of API function CanTSyn_Init

Upstream requirements: RS_TS_20035

⌈

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	

⌋

CANTSYN_E_INIT_FAILED is reported as specified by [SWS_BSW_00050] in [4]. See section 7.2.3 for details.

8.3.2 CanTSyn_SetTransmissionMode

[SWS_CanTSyn_00095] Definition of API function CanTSyn_SetTransmissionMode

Upstream requirements: RS_TS_20035

⌈

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	

⌋

[SWS_CanTSyn_00134]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337, SRS_BSW_00386

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

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

⌋

8.3.3 CanTSyn_RxIndication

[SWS_CanTSyn_00096] Definition of callback function CanTSyn_RxIndication

Upstream requirements: RS_TS_20035

⌈

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

⌋

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]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337, SRS_BSW_00386

⌈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)

⌋

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.

8.3.4 CanTSyn_TxConfirmation

[SWS_CanTSyn_00099] Definition of callback function CanTSyn_TxConfirmation

Upstream requirements: RS_TS_20035

⌈

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 successfully. 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	

⌋

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

[SWS_CanTSyn_00100]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337, SRS_BSW_00386

⌈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`

⌋

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.

8.3.5 CanTSyn_IcvGenerationIndication

[SWS_CanTSyn_91002] Definition of API function CanTSyn_IcvGenerationIndication

Upstream requirements: RS_TS_20073

⌈

Service Name	CanTSyn_IcvGenerationIndication	
Syntax	`void CanTSyn_IcvGenerationIndication ( uint32 jobId, Crypto_ResultType result )`	
Service ID [hex]	0x07	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	jobId	JobID of the operation that caused the callback.
	result	Contains the result of the cryptographic operation.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	By this API service the CanTSyn gets an indication and the result of ICV generation.	
Available via	CanTSyn.h	

⌋

[SWS_CanTSyn_00202]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337, SRS_BSW_00386

⌈The function `CanTSyn_IcvGenerationIndication` 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:

- jobId is invalid (`CANTSYN_E_PARAM`)

⌋

8.3.6 CanTSyn_IcvVerificationIndication

[SWS_CanTSyn_91003] Definition of API function CanTSyn_IcvVerificationIndication

Upstream requirements: RS_TS_20073

⌈

Service Name	CanTSyn_IcvVerificationIndication	
Syntax	`void CanTSyn_IcvVerificationIndication ( uint32 jobId, Crypto_ResultType result )`	
Service ID [hex]	0x08	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	jobId	JobID of the operation that caused the callback.
	result	Contains the result of the cryptographic operation.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	By this API service the CanTSyn gets an indication and the result of ICV verification.	
Available via	CanTSyn.h	

⌋

[SWS_CanTSyn_00203]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈The function CanTSyn_IcvVerificationIndication() 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:

- jobId is invalid (`CANTSYN_E_PARAM`)

⌋

8.4 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.

8.4.1 CanTSyn_MainFunction

[SWS_CanTSyn_00102] Definition of scheduled function CanTSyn_MainFunction

Upstream requirements: RS_TS_20035

⌈

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

⌋

[SWS_CanTSyn_00103]

Upstream requirements: RS_TS_20035

⌈The frequency of invocations of `CanTSyn_MainFunction` is determined by the configuration parameter `CanTSynMainFunctionPeriod`.⌋

8.5 Expected interfaces

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

8.5.1 Mandatory interfaces

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

[SWS_CanTSyn_00105] Definition of mandatory interfaces required by module CanTSyn

Upstream requirements: RS_TS_20035

⌈

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

⌋

8.5.2 Optional interfaces

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

[SWS_CanTSyn_00106] Definition of optional interfaces requested by module CanTSyn

Upstream requirements: RS_TS_20035

⌈

API Function	*Header File*	*Description*
CanIf_EnableEgressTimeStamp (draft)	CanIf.h	This service calls the corresponding CAN Driver service to activate egress time stamping on a dedicated message object. **Tags:** atp.Status=draft
CanIf_GetCurrentTime (draft)	CanIf.h	This service calls the corresponding CAN Driver service to retrieve the current time value out of the HW registers. **Tags:** atp.Status=draft
CanIf_GetEgressTimeStamp (draft)	CanIf.h	This service calls the corresponding CAN Driver service to read back the egress time stamp on a dedicated message object. It needs to be called within the TxConfirmation() function. **Tags:** atp.Status=draft
CanIf_GetIngressTimeStamp (draft)	CanIf.h	This service calls the corresponding CAN Driver service to reads back the ingress time stamp on a dedicated message object. It needs to be called within the RxIndication() function. **Tags:** atp.Status=draft
Crc_CalculateCRC8H2F	Crc.h	This service makes a CRC8 calculation with the Polynomial 0x2F on Crc_Length
Csm_MacGenerate	Csm.h	Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.
Csm_MacVerify	Csm.h	Verifies the given MAC by comparing if the MAC is generated with the given data.
Csm_SignatureGenerate	Csm.h	Uses the given data to perform the signature calculation and stores the signature in the memory location pointed by the result pointer.
Csm_SignatureVerify	Csm.h	Verifies the given signature by checking if it was generated with the given data.
Det_ReportError	Det.h	Service to report development errors.
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.
IdsM_ReportSecurityEvent	IdsM.h	This API is the interface to report security events to the IdsM.
LSduR_CanTSynTransmit (draft)	LSduR_CanTSyn.h	Requests transmission of a PDU.

▽

△

API Function	*Header File*	*Description*
StbM_BusSetGlobalTime	StbM.h	Allows the Time Base Provider Modules to forward the Rx Time Tuple to the StbM.
StbM_CanSetMasterTimingData	StbM_CanTSyn.h	Provides CAN Timesyn module specific data for a Time Master to the StbM.
StbM_CanSetSlaveTimingData	StbM_CanTSyn.h	Allows the CanTSyn Module to forward CAN specific details to the StbM.
StbM_GetCurrentTime	StbM.h	Returns a time tuple (Local time, Global time and Timebase status) and user data details 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_GetRxFreshness	StbM.h	This interface is used by the StbM to query the current freshness value.
StbM_GetTimeBaseStatus	StbM.h	Returns detailed status information for a Synchronized (or Pure Local) 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.
StbM_GetTxFreshness	StbM.h	This API returns the freshness value from the Most Significant Bits in the first byte, of the Freshness array, in big endian format.
StbM_GetTxFreshnessTruncData	StbM.h	This interface is used by the StbM to obtain the current freshness value. The interface function provides also the truncated freshness transmitted in the secured time sync message.
StbM_SPduTxConfirmation	StbM.h	This interface is used by the StbM to indicate that the Secured Time Synchronization Message has been initiated for transmission.

⌋

8.5.3 Callout Interfaces

There are no callout interfaces defined.

8.5.4 Configurable Interfaces

There are no configurable interfaces defined.

8.6 Service Interfaces

There are no service interfaces defined.

9 Sequence diagrams

9.1 Enable Egress Timestamping

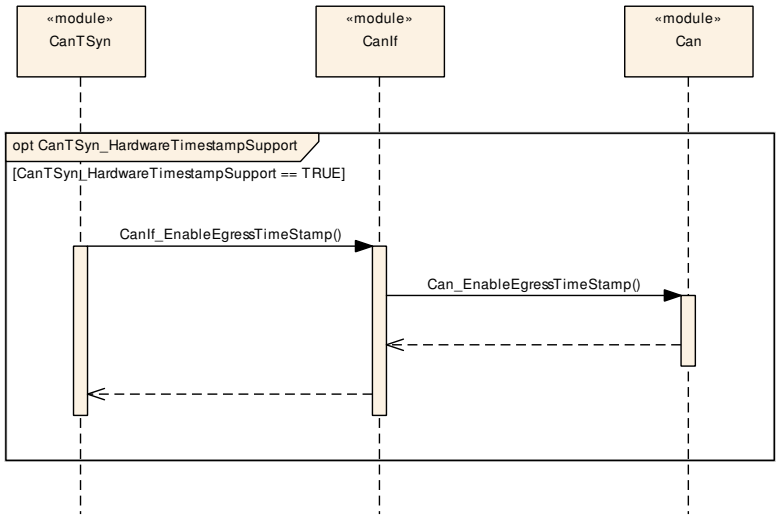


Figure 9.1: Enable Egress Timestamping

9.2 CAN Time Synchronization (Time Master)

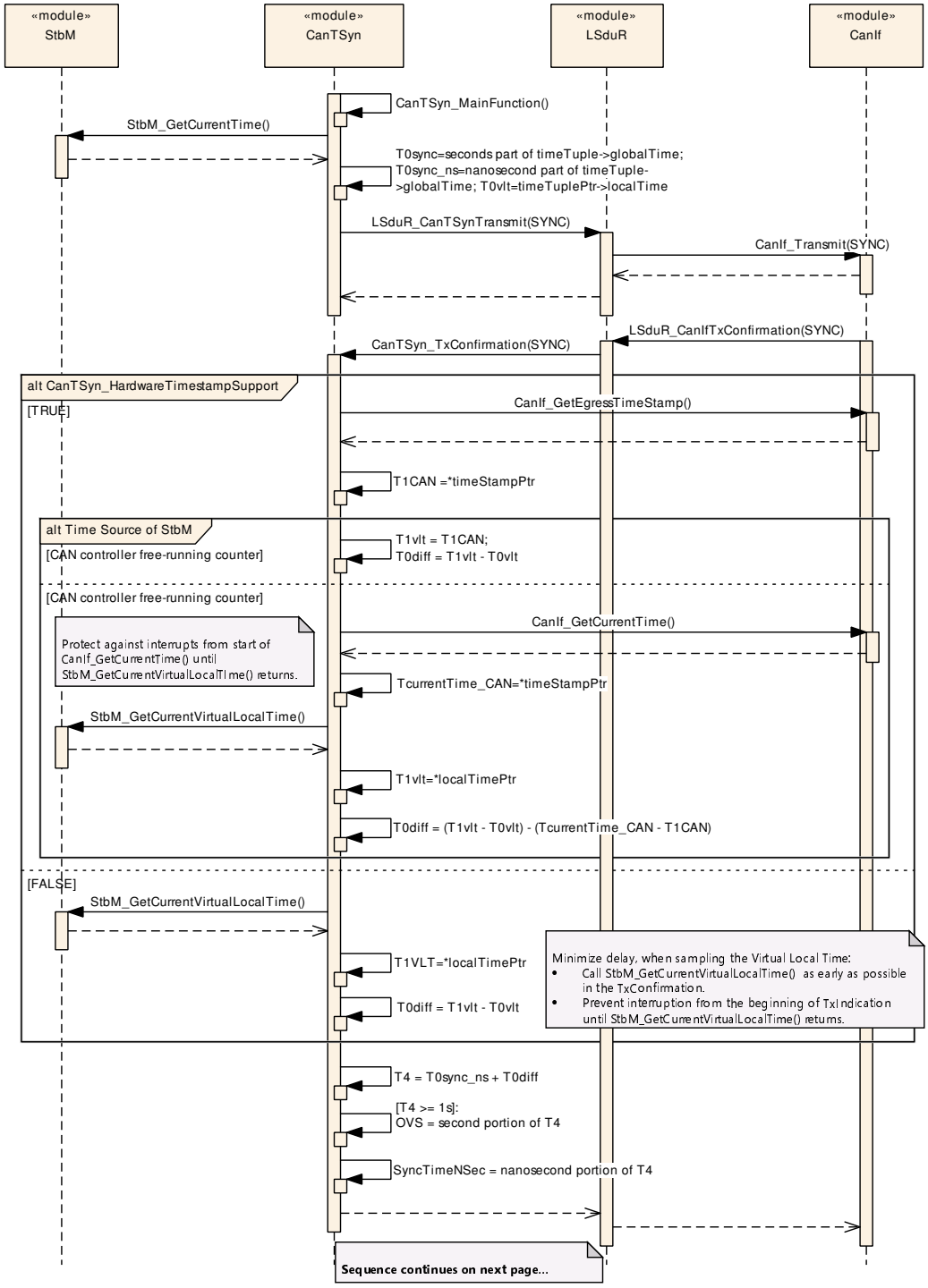


Figure 9.2: CAN Time Synchronization (Time Master), Part 1

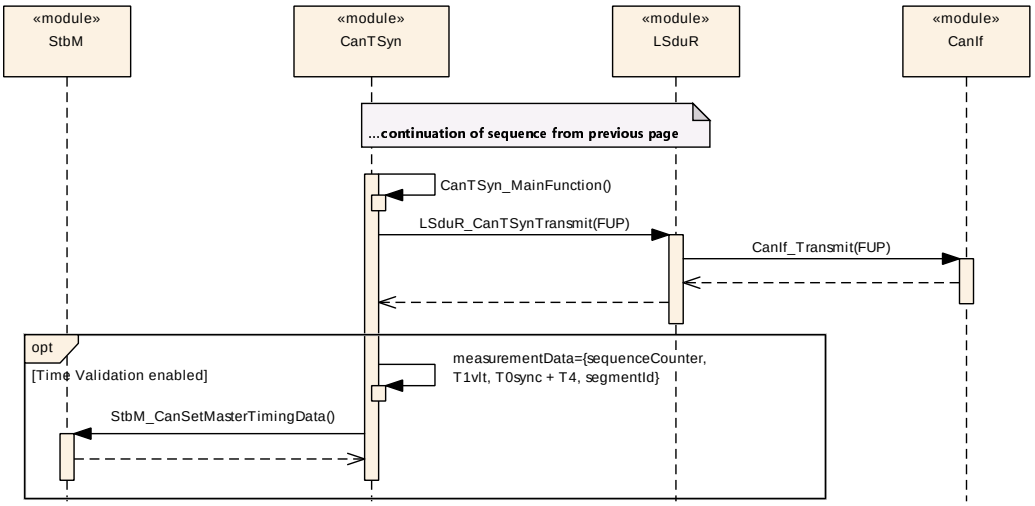


Figure 9.3: CAN Time Synchronization (Time Master), Part 2

9.3 CAN Time Synchronization (Time Slave)

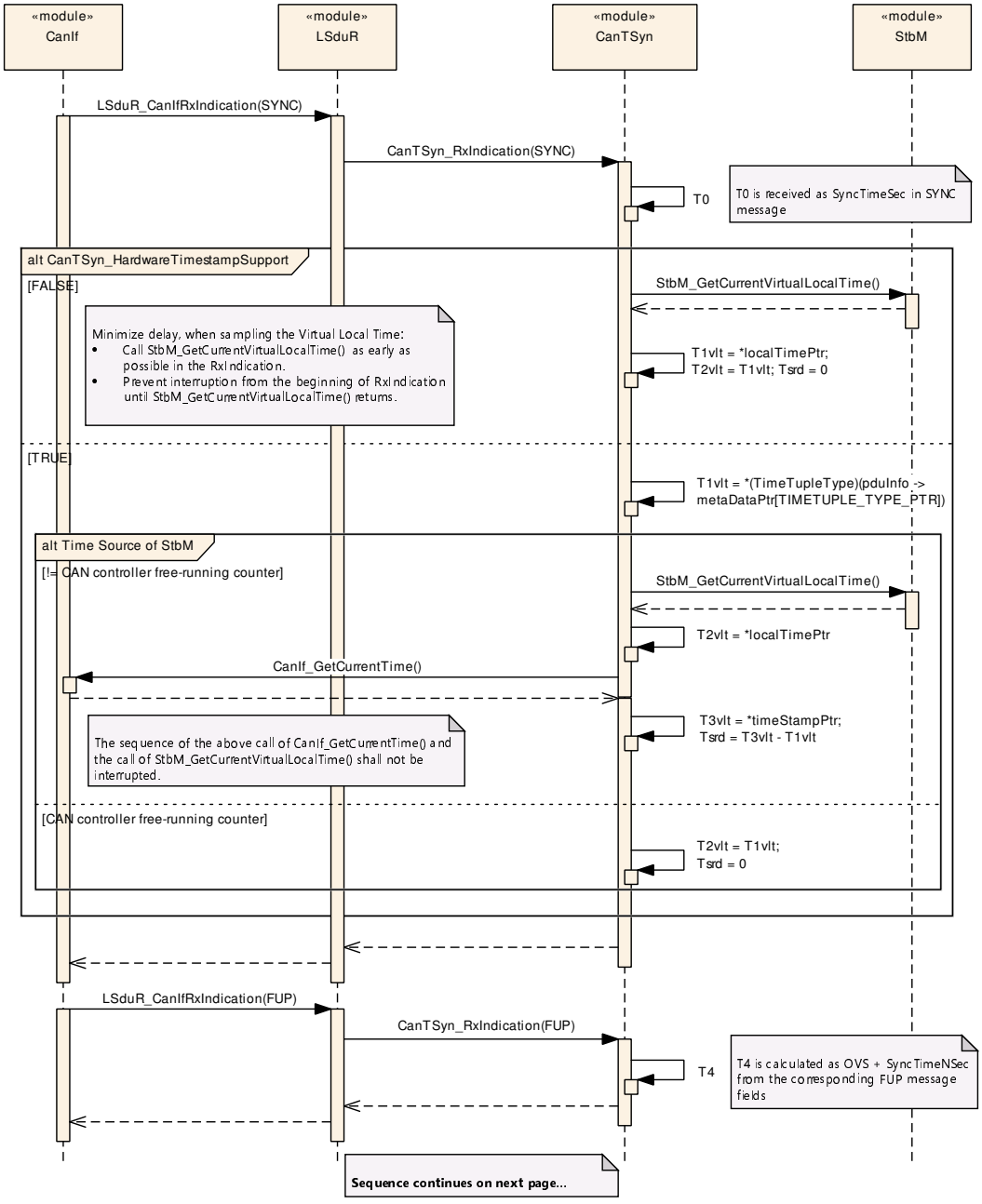


Figure 9.4: CAN Time Synchronization (Time Slave), Part 1

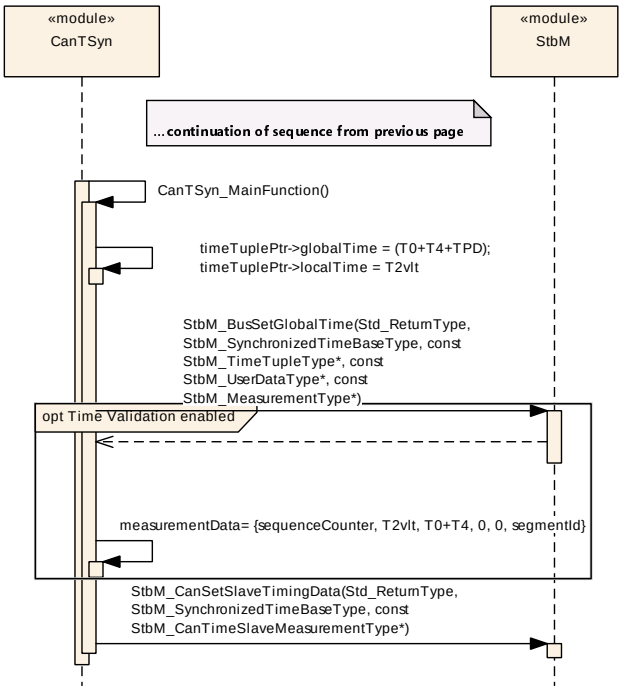


Figure 9.5: CAN Time Synchronization (Time Slave), Part 2

9.4 CAN Secure Time Synchronization (Time Master, Time Slave)

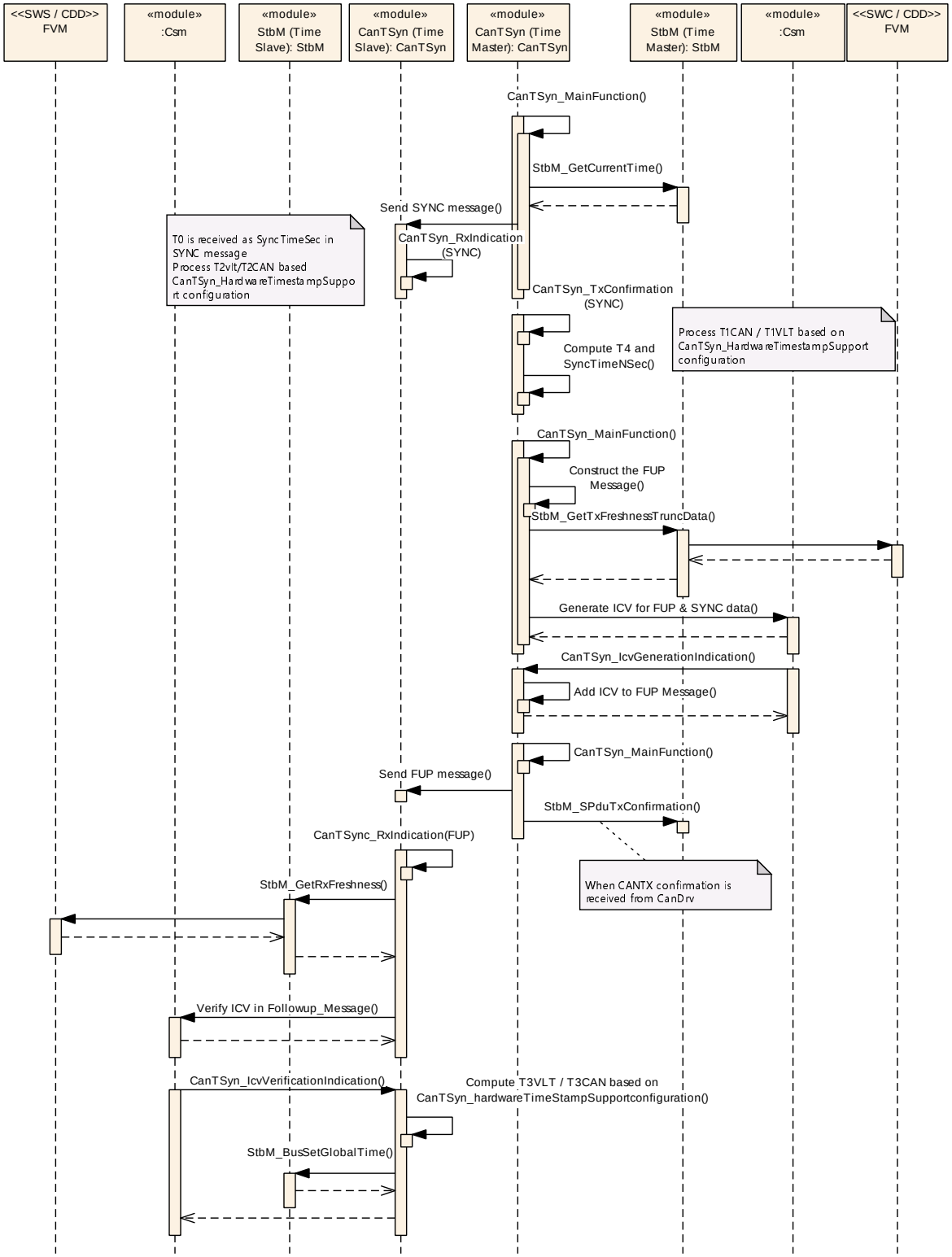


Figure 9.6: Secure Time Synchronization Sequence

10 Configuration specification

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

Chapter Chapter 10.2 specifies the structure (containers) and the parameters of the module `CanTSyn`.

Chapter Chapter 10.4 specifies published information of the module `CanTSyn`.

10.1 How to read this chapter

For details refer to [4] Chapter 10.1 *"Introduction to configuration specification"*.

10.2 Containers and configuration parameters

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

10.2.1 Variants

[SWS_CanTSyn_00108]

Upstream requirements: RS_TS_20038

⌈The Time Synchronization over CAN shall support the configuration for Time Master, Time Slave and Time Gateway.⌋

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

[ECUC_CanTSyn_00001] Definition of EcucModuleDef CanTSyn ⌈

Module Name	CanTSyn
Description	Configuration of the Synchronized Time-base Manager (StbM) module with respect to global time handling on CAN.
Post-Build Selectable Support	true
Post-Build Loadable Support	false

Included Containers		
Container Name	**Multiplicity**	**Description**
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

[ECUC_CanTSyn_00003] Definition of EcucParamConfContainerDef CanTSyn General ⌈

Container Name	CanTSynGeneral
Parent Container	CanTSyn
Upstream mapping defined	–
Description	This container holds the general parameters of the CAN-specific Synchronized Time-base Manager
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanTSynDevErrorDetect	1	[ECUC_CanTSyn_00002]
CanTSynEnableSecurityEventReporting	1	[ECUC_CanTSyn_00055]
CanTSynHardwareTimestampSupport	1	[ECUC_CanTSyn_00054]
CanTSynMainFunctionPeriod	1	[ECUC_CanTSyn_00019]
CanTSynSyncTransmissionsPerCycle	0..1	[ECUC_CanTSyn_00088]
CanTSynTimeValidationSupport	1	[ECUC_CanTSyn_00050]

Included Containers		
Container Name	**Multiplicity**	**Description**
CanTSynSecurityEventRefs	0..1	Container for the references to IdsMEvent elements representing the security events that the CanTSyn module shall report to the IdsM in case the coresponding security related event occurs (and if CanTSynEnableSecurityEventReporting is set to "true"). The standardized security events in this container can be extended by vendor-specific security events.

⌋

[ECUC_CanTSyn_00002] Definition of EcucBooleanParamDef CanTSynDevErrorDetect ⌈

Parameter Name	CanTSynDevErrorDetect
Parent Container	CanTSynGeneral
Upstream mapping defined	–
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
Pre-Compile only	true

⌋

[ECUC_CanTSyn_00055] Definition of EcucBooleanParamDef CanTSynEnableSecurityEventReporting ⌈

Parameter Name	CanTSynEnableSecurityEventReporting
Parent Container	CanTSynGeneral
Upstream mapping defined	–
Description	Switches the reporting of security events to the IdsM: - true: reporting is enabled. - false: reporting is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_CanTSyn_00054] Definition of EcucBooleanParamDef CanTSynHardwareTimestampSupport

Status: DRAFT

⌈

Parameter Name	CanTSynHardwareTimestampSupport
Parent Container	CanTSynGeneral
Upstream mapping defined	–
Description	Activate/Deactivate the hardware time stamping functionality of the CAN hardware. True: Timestamp is retrieved from the CAN hardware False: Timestamp is retrieved from the StbM **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_CanTSyn_00019] Definition of EcucFloatParamDef CanTSynMainFunctionPeriod ⌈

Parameter Name	CanTSynMainFunctionPeriod	
Parent Container	CanTSynGeneral	
Upstream mapping defined	–	
Description	Schedule period of the main function CanTSyn_MainFunction. Unit: [s].	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	]0 .. INF[	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_CanTSyn_00088] Definition of EcucIntegerParamDef CanTSynSyncTransmissionsPerCycle

Status: DRAFT

⌈

Parameter Name	CanTSynSyncTransmissionsPerCycle	
Parent Container	CanTSynGeneral	
Upstream mapping defined	–	
Description	Maximum number of SYNC message transmissions within one CanTSyn_MainFunction **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	0	
Pre-Compile only	true	

⌋

[ECUC_CanTSyn_00050] Definition of EcucBooleanParamDef CanTSynTimeValidationSupport ⌈

Parameter Name	CanTSynTimeValidationSupport
Parent Container	CanTSynGeneral
Upstream mapping defined	–
Description	Switches support for Time Validation on or off. • true: Time Validation is enabled. • false: Time Validation is disabled
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

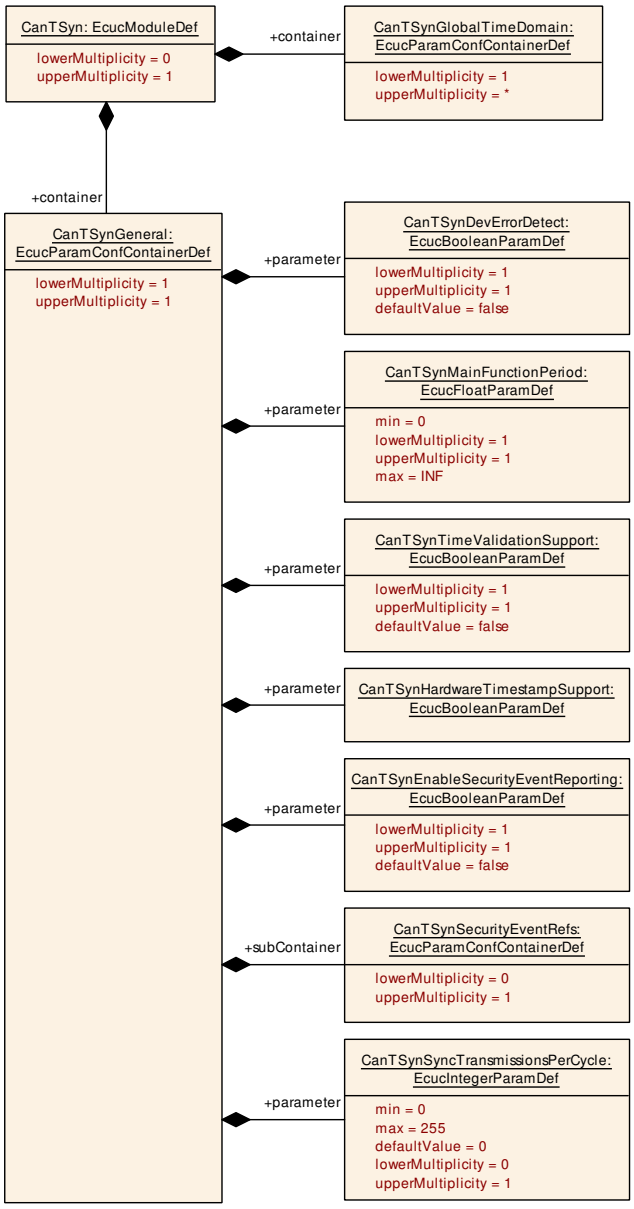


Figure 10.1: CanTSynGeneral

10.2.4 CanTSynSecurityEventRefs

[ECUC_CanTSyn_00056] Definition of EcucParamConfContainerDef CanTSynSecurityEventRefs ⌈

Container Name	CanTSynSecurityEventRefs
Parent Container	CanTSynGeneral
Upstream mapping defined	–

▽

△

Description	Container for the references to IdsMEvent elements representing the security events that the CanTSyn module shall report to the IdsM in case the coresponding security related event occurs (and if CanTSynEnableSecurityEventReporting is set to "true"). The standardized security events in this container can be extended by vendor-specific security events.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SEV_TSYN_CAN_FRESHNESS_NOT_AVAILABLE	0..1	[ECUC_CanTSyn_00059]
SEV_TSYN_CAN_ICV_GENERATION_FAILED	0..1	[ECUC_CanTSyn_00057]
SEV_TSYN_CAN_ICV_VERIFICATION_FAILED	0..1	[ECUC_CanTSyn_00058]
SEV_TSYN_CAN_MSG_SEQUENCE_ERROR	0..1	[ECUC_CanTSyn_00083]

No Included Containers

⌋

[ECUC_CanTSyn_00059] Definition of EcucReferenceDef SEV_TSYN_CAN_FRESHNESS_NOT_AVAILABLE ⌈

Parameter Name	SEV_TSYN_CAN_FRESHNESS_NOT_AVAILABLE
Parent Container	CanTSynSecurityEventRefs
Upstream mapping defined	–
Description	FV not available from FVM. Context data provides the respective domain ID.
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_CanTSyn_00057] Definition of EcucReferenceDef SEV_TSYN_CAN_ICV_GENERATION_FAILED ⌈

Parameter Name	SEV_TSYN_CAN_ICV_GENERATION_FAILED
Parent Container	CanTSynSecurityEventRefs
Upstream mapping defined	–
Description	ICV generation for Follow_Up message failed. Context data provides the respective domain ID
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_CanTSyn_00058] Definition of EcucReferenceDef SEV_TSYN_CAN_ICV_VERIFICATION_FAILED ⌈

Parameter Name	SEV_TSYN_CAN_ICV_VERIFICATION_FAILED
Parent Container	CanTSynSecurityEventRefs
Upstream mapping defined	–
Description	ICV verification for Follow_Up message failed. Context data provides the respective domain ID.
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_CanTSyn_00083] Definition of EcucReferenceDef SEV_TSYN_CAN_MSG_SEQUENCE_ERROR ⌈

Parameter Name	SEV_TSYN_CAN_MSG_SEQUENCE_ERROR
Parent Container	CanTSynSecurityEventRefs
Upstream mapping defined	–
Description	Failed to receive correct sequence of SYNC and FUP from the TimeMaster within (CanTSynGlobalTimeFollowUpTimeout).
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

10.2.5 CanTSynGlobalTimeDomain

[ECUC_CanTSyn_00004] Definition of EcucParamConfContainerDef CanTSynGlobalTimeDomain ⌈

Container Name	CanTSynGlobalTimeDomain
Parent Container	CanTSyn
Upstream mapping defined	[10, CP_TPS_SystemTemplate]
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.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanTSynEnableTimeValidation	0..1	[ECUC_CanTSyn_00051]
CanTSynGlobalTimeDomainId	1	[ECUC_CanTSyn_00005]
CanTSynGlobalTimeNetworkSegmentId	0..1	[ECUC_CanTSyn_00052]
CanTSynSynchronizedTimeBaseRef	1	[ECUC_CanTSyn_00022]

Included Containers		
Container Name	**Multiplicity**	**Description**
CanTSynGlobalTimeFupData IDList	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 a Time Master for a Time Domain (refer to parent container). If CanTSynGlobalTimeMaster container exists, the local ECU acts as a Time Master for the Time Domain.
CanTSynGlobalTimeSlave	0..1	Configuration of a Time Slave for a Time Domain (refer to parent container). If CanTSynGlobalTimeSlave container exists, the local ECU acts as a Time Slave for the Time Domain.
CanTSynGlobalTimeSyncData IDList	0..1	The DataIDList for SYNC messages ensures the identification of data elements due to CRC calculation and message authentication process.

⌋

[ECUC_CanTSyn_00051] Definition of EcucBooleanParamDef CanTSynEnable TimeValidation ⌈

Parameter Name	CanTSynEnableTimeValidation
Parent Container	CanTSynGlobalTimeDomain
Upstream mapping defined	–
Description	Enables/disables time recording for Time Validation for a specific Time Domain. Dependency: Only valid if CanTSynTimeValidationSupport is TRUE. Value set according to parameter StbMEnableTimeValidation of the referenced Time Base in the StbM.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_CanTSyn_00005] Definition of EcucIntegerParamDef CanTSynGlobal TimeDomainId ⌈

Parameter Name	CanTSynGlobalTimeDomainId	
Parent Container	CanTSynGlobalTimeDomain	
Upstream mapping defined	[10, CP_TPS_SystemTemplate]	
Description	The global time domain ID.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 15	

▽

△

Default value	–	
Post-Build Value	-	Selectable

⌋

[ECUC_CanTSyn_00052] Definition of EcucIntegerParamDef CanTSynGlobalTimeNetworkSegmentId ⌈

Parameter Name	CanTSynGlobalTimeNetworkSegmentId	
Parent Container	CanTSynGlobalTimeDomain	
Upstream mapping defined	[10, CP_TPS_SystemTemplate]	
Description	This represents the numerical identifier of the network on system level scope where this Global Time has been communicated on.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Post-Build Value	-	Selectable

⌋

[ECUC_CanTSyn_00022] Definition of EcucReferenceDef CanTSynSynchronizedTimeBaseRef ⌈

Parameter Name	CanTSynSynchronizedTimeBaseRef
Parent Container	CanTSynGlobalTimeDomain
Upstream mapping defined	–
Description	Mandatory reference to the required synchronized time-base.
Multiplicity	1
Type	Symbolic name reference to StbMSynchronizedTimeBase
Pre-Compile only	true

⌋

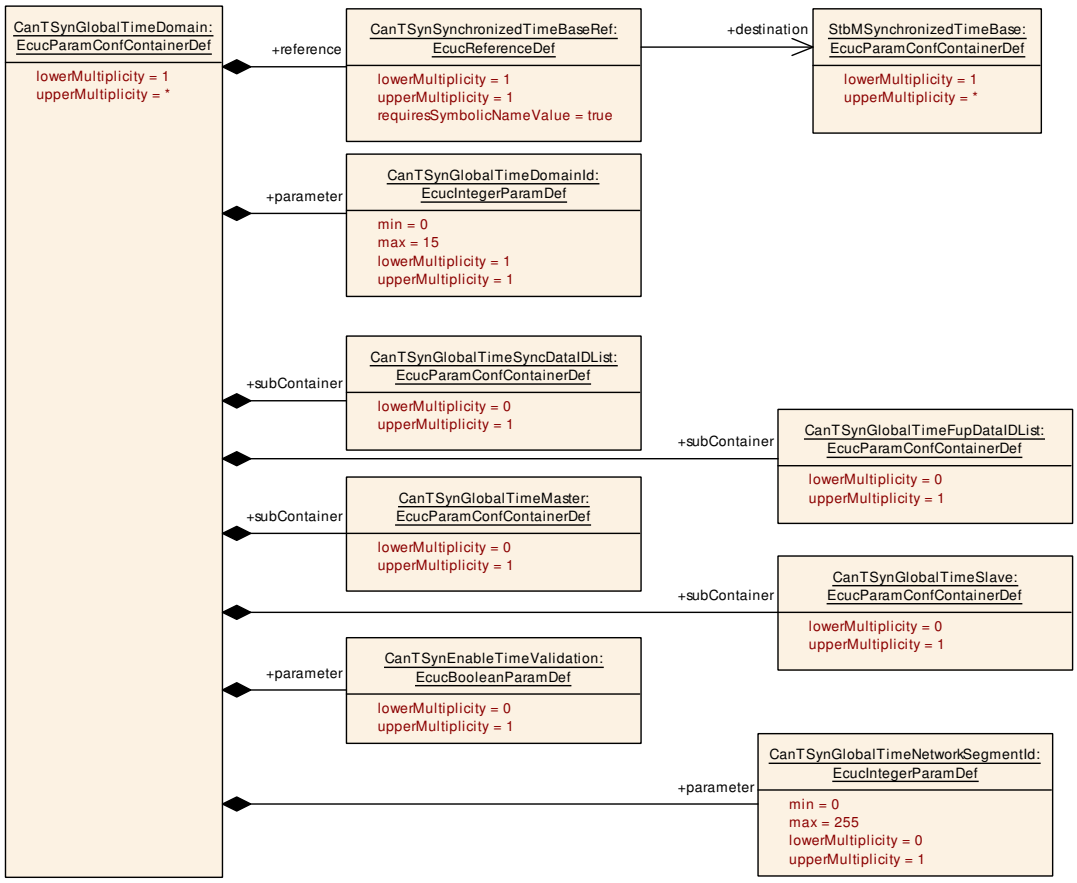


Figure 10.2: CanTSynGlobalTimeDomain

10.2.6 CanTSynGlobalTimeSyncDataIDList

[ECUC_CanTSyn_00024] Definition of EcucParamConfContainerDef CanTSynGlobalTimeSyncDataIDList ⌈

Container Name	CanTSynGlobalTimeSyncDataIDList	
Parent Container	CanTSynGlobalTimeDomain	
Upstream mapping defined	[10, CP_TPS_SystemTemplate]	
Description	The DataIDList for SYNC messages ensures the identification of data elements due to CRC calculation and message authentication process.	
Multiplicity	0..1	
Post-Build Multiplicity	-	Selectable
Configuration Parameters		

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CanTSynGlobalTimeSyncDataIDListElement	16	Element of the DataIDList for SYNC messages ensures the identification of data elements due to CRC calculation and message authentication process.

⌋

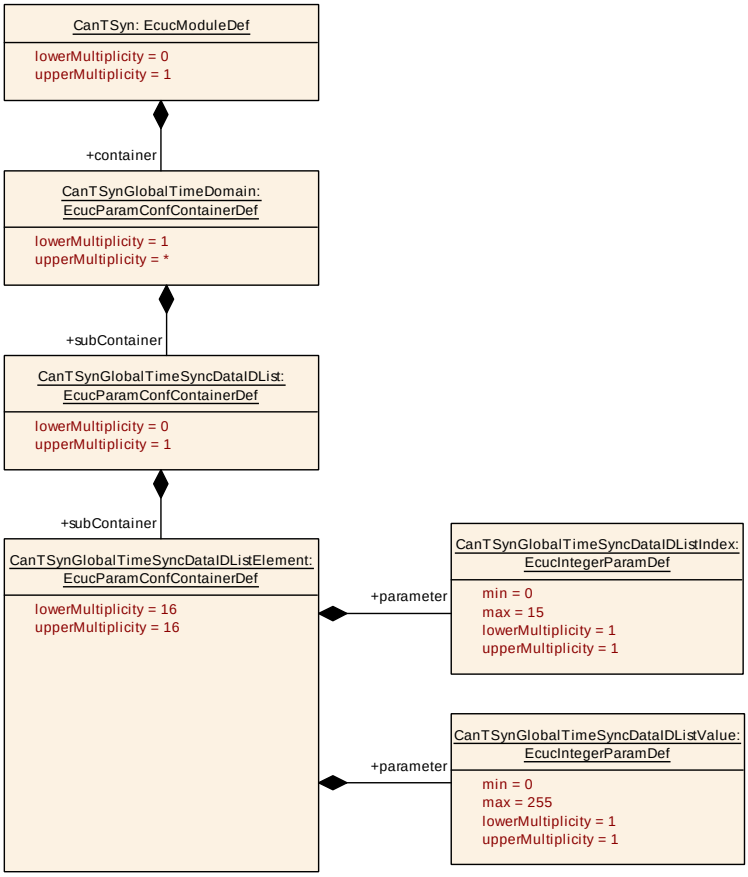


Figure 10.3: CanTSynGlobalTimeSyncDataIDList

10.2.7 CanTSynGlobalTimeSyncDataIDListElement

[ECUC_CanTSyn_00028] Definition of EcucParamConfContainerDef CanTSynGlobalTimeSyncDataIDListElement ⌈

Container Name	CanTSynGlobalTimeSyncDataIDListElement
Parent Container	CanTSynGlobalTimeSyncDataIDList
Upstream mapping defined	[10, CP_TPS_SystemTemplate]
Description	Element of the DataIDList for SYNC messages ensures the identification of data elements due to CRC calculation and message authentication process.
Multiplicity	16
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanTSynGlobalTimeSyncDataIDListIndex	1	[ECUC_CanTSyn_00029]
CanTSynGlobalTimeSyncDataIDListValue	1	[ECUC_CanTSyn_00030]

No Included Containers

⌋

[ECUC_CanTSyn_00029] Definition of EcucIntegerParamDef CanTSynGlobalTimeSyncDataIDListIndex ⌈

Parameter Name	CanTSynGlobalTimeSyncDataIDListIndex	
Parent Container	CanTSynGlobalTimeSyncDataIDListElement	
Upstream mapping defined	–	
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 Value	-	Selectable

⌋

[ECUC_CanTSyn_00030] Definition of EcucIntegerParamDef CanTSynGlobalTimeSyncDataIDListValue ⌈

Parameter Name	CanTSynGlobalTimeSyncDataIDListValue	
Parent Container	CanTSynGlobalTimeSyncDataIDListElement	
Upstream mapping defined	–	
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 Value	-	Selectable

⌋

10.2.8 CanTSynGlobalTimeFupDataIDList

[ECUC_CanTSyn_00025] Definition of EcucParamConfContainerDef CanTSynGlobalTimeFupDataIDList ⌈

Container Name	CanTSynGlobalTimeFupDataIDList
Parent Container	CanTSynGlobalTimeDomain
Upstream mapping defined	[10, CP_TPS_SystemTemplate]
Description	The DataIDList for FUP messages ensures the identification of data elements due to CRC calculation and message authentication process.

▽

△

Multiplicity	0..1	
Post-Build Multiplicity	-	Selectable
Configuration Parameters		

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CanTSynGlobalTimeFupDataIDListElement	16	Element of the DataIDList for FUP messages ensures the identification of data elements due to CRC calculation and message authentication process.

⌋

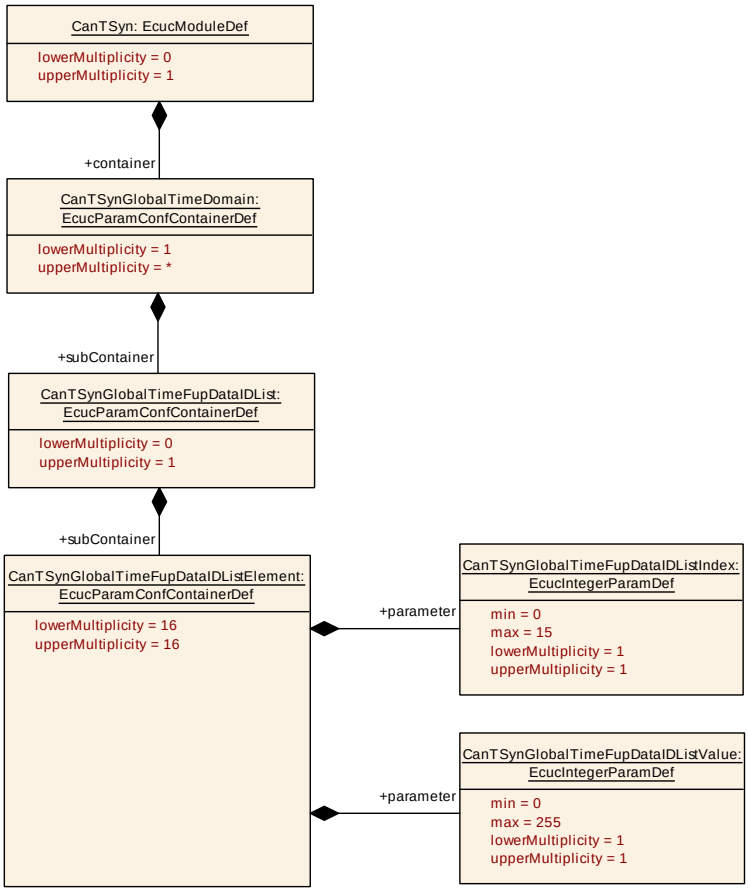


Figure 10.4: CanTSynGlobalTimeFupDataIDList

10.2.9 CanTSynGlobalTimeFupDataIDListElement

[ECUC_CanTSyn_00031] Definition of EcucParamConfContainerDef CanTSynGlobalTimeFupDataIDListElement ⌈

Container Name	CanTSynGlobalTimeFupDataIDListElement
Parent Container	CanTSynGlobalTimeFupDataIDList
Upstream mapping defined	[10, CP_TPS_SystemTemplate]
Description	Element of the DataIDList for FUP messages ensures the identification of data elements due to CRC calculation and message authentication process.
Multiplicity	16
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanTSynGlobalTimeFupDataIDListIndex	1	[ECUC_CanTSyn_00032]
CanTSynGlobalTimeFupDataIDListValue	1	[ECUC_CanTSyn_00033]

No Included Containers

⌋

[ECUC_CanTSyn_00032] Definition of EcucIntegerParamDef CanTSynGlobalTimeFupDataIDListIndex ⌈

Parameter Name	CanTSynGlobalTimeFupDataIDListIndex	
Parent Container	CanTSynGlobalTimeFupDataIDListElement	
Upstream mapping defined	–	
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 Value	-	Selectable

⌋

[ECUC_CanTSyn_00033] Definition of EcucIntegerParamDef CanTSynGlobalTimeFupDataIDListValue ⌈

Parameter Name	CanTSynGlobalTimeFupDataIDListValue	
Parent Container	CanTSynGlobalTimeFupDataIDListElement	
Upstream mapping defined	–	
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 Value	-	Selectable

⌋

10.2.10 CanTSynGlobalTimeMaster

[ECUC_CanTSyn_00007] Definition of EcucParamConfContainerDef CanTSyn GlobalTimeMaster ⌈

Container Name	CanTSynGlobalTimeMaster	
Parent Container	CanTSynGlobalTimeDomain	
Upstream mapping defined	[10, CP_TPS_SystemTemplate]	
Description	Configuration of a Time Master for a Time Domain (refer to parent container). If Can TSynGlobalTimeMaster container exists, the local ECU acts as a Time Master for the Time Domain.	
Multiplicity	0..1	
Post-Build Multiplicity	-	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanTSynCyclicMsgResumeTime	1	[ECUC_CanTSyn_00044]
CanTSynGlobalTimeDebounceTime	1	[ECUC_CanTSyn_00045]
CanTSynGlobalTimeTxCrcSecured	1	[ECUC_CanTSyn_00015]
CanTSynGlobalTimeTxIcvSecured	1	[ECUC_CanTSyn_00111]
CanTSynGlobalTimeTxPeriod	1	[ECUC_CanTSyn_00017]
CanTSynImmediateTimeSync	1	[ECUC_CanTSyn_00043]

Included Containers		
Container Name	**Multiplicity**	**Description**
CanTSynGlobalTimeMasterPdu	1	This container encloses the configuration of the PDU that is supposed to contain the global time information.
CanTSynGlobalTimeTxIcv Generation	0..1	This container collects configuration that shall be used for ICV generation.

⌋

[ECUC_CanTSyn_00044] Definition of EcucFloatParamDef CanTSynCyclicMsg ResumeTime ⌈

Parameter Name	CanTSynCyclicMsgResumeTime	
Parent Container	CanTSynGlobalTimeMaster	
Upstream mapping defined	[10, CP_TPS_SystemTemplate]	
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 Value	-	Selectable

⌋

[ECUC_CanTSyn_00045] Definition of EcucFloatParamDef CanTSynGlobalTimeDebounceTime ⌈

Parameter Name	CanTSynGlobalTimeDebounceTime	
Parent Container	CanTSynGlobalTimeMaster	
Upstream mapping defined	[10, CP_TPS_SystemTemplate]	
Description	This represents the configuration of a TX debounce time for SYNC and FUP messages compared to a message before with the same PDU. Unit: seconds	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 4]	
Default value	–	
Post-Build Value	-	Selectable

⌋

[ECUC_CanTSyn_00015] Definition of EcucEnumerationParamDef CanTSynGlobalTimeTxCrcSecured ⌈

Parameter Name	CanTSynGlobalTimeTxCrcSecured	
Parent Container	CanTSynGlobalTimeMaster	
Upstream mapping defined	[10, CP_TPS_SystemTemplate]	
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 Value	-	Selectable

⌋

[ECUC_CanTSyn_00111] Definition of EcucEnumerationParamDef CanTSynGlobalTimeTxIcvSecured

Status: DRAFT

⌈

Parameter Name	CanTSynGlobalTimeTxIcvSecured	
Parent Container	CanTSynGlobalTimeMaster	
Upstream mapping defined	–	
Description	This parameter controls whether or not ICV generation shall be supported. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ICV_NOT_SUPPORTED	The Timesync module shall not generate the ICV. **Tags:** atp.Status=draft

▽

△

	ICV_SUPPORTED	The Timesync module shall generate the ICV. **Tags:** atp.Status=draft
Default value	ICV_NOT_SUPPORTED	
Pre-Compile only	true	

⌋

[ECUC_CanTSyn_00017] Definition of EcucFloatParamDef CanTSynGlobalTimeTxPeriod ⌈

Parameter Name	CanTSynGlobalTimeTxPeriod	
Parent Container	CanTSynGlobalTimeMaster	
Upstream mapping defined	[10, CP_TPS_SystemTemplate]	
Description	This represents configuration of the TX period. Unit: seconds	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Post-Build Value	-	Selectable

⌋

[ECUC_CanTSyn_00043] Definition of EcucBooleanParamDef CanTSynImmediateTimeSync ⌈

Parameter Name	CanTSynImmediateTimeSync	
Parent Container	CanTSynGlobalTimeMaster	
Upstream mapping defined	–	
Description	Enables/Disables the cyclic polling of StbM_GetTimeBaseUpdateCounter() within CanTSyn_MainFunction().	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Value	-	Selectable

⌋

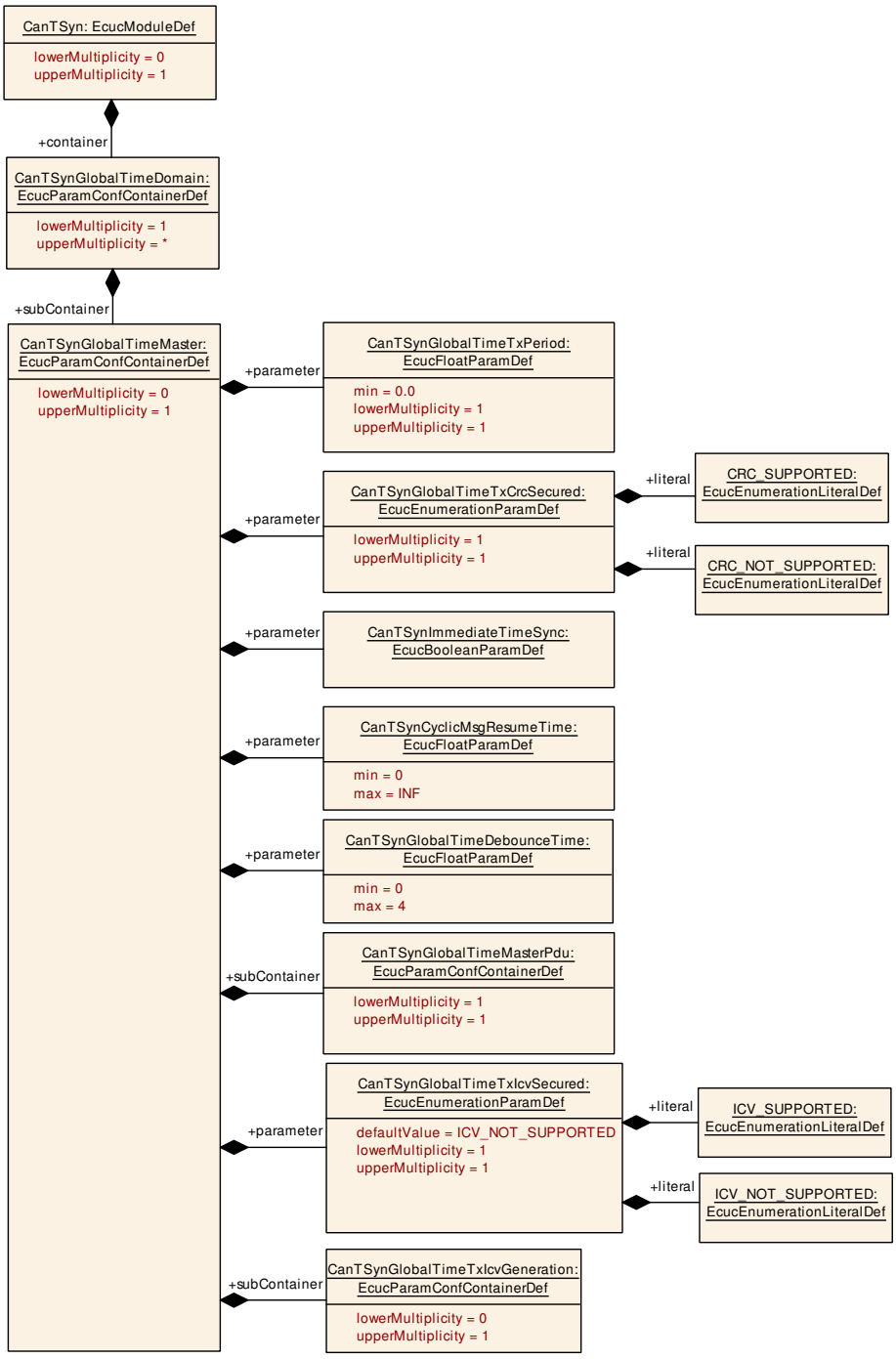


Figure 10.5: CanTSynGlobalTimeMaster

10.2.11 CanTSynGlobalTimeMasterPdu

[ECUC_CanTSyn_00009] Definition of EcucParamConfContainerDef CanTSyn GlobalTimeMasterPdu ⌈

Container Name	CanTSynGlobalTimeMasterPdu
Parent Container	CanTSynGlobalTimeMaster
Upstream mapping defined	–
Description	This container encloses the configuration of the PDU that is supposed to contain the global time information.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
CanTSynGlobalTimeMasterConfirmationHandleId	1	[ECUC_CanTSyn_00008]
CanTSynGlobalTimePduRef	1	[ECUC_CanTSyn_00027]

No Included Containers

⌋

[ECUC_CanTSyn_00008] Definition of EcucIntegerParamDef CanTSynGlobalTimeMasterConfirmationHandleId ⌈

Parameter Name	CanTSynGlobalTimeMasterConfirmationHandleId	
Parent Container	CanTSynGlobalTimeMasterPdu	
Upstream mapping defined	–	
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	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_CanTSyn_00027] Definition of EcucReferenceDef CanTSynGlobalTimePduRef ⌈

Parameter Name	CanTSynGlobalTimePduRef	
Parent Container	CanTSynGlobalTimeMasterPdu, CanTSynGlobalTimeSlavePdu	
Upstream mapping defined	[10, CP_TPS_SystemTemplate]	
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 Value	-	Selectable

⌋

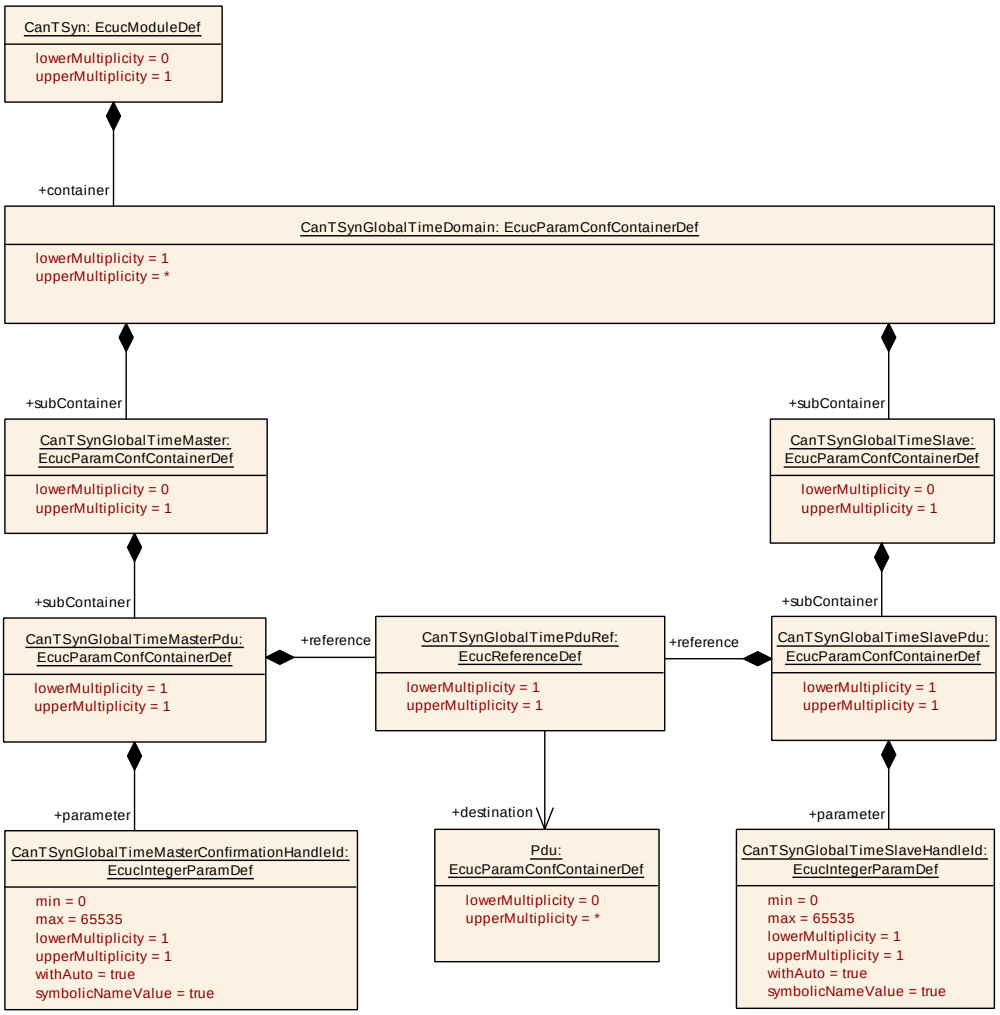


Figure 10.6: CanTSynGlobalTimePdu

10.2.12 CanTSynGlobalTimeTxIcvGeneration

[ECUC_CanTSyn_00060] Definition of EcucParamConfContainerDef CanTSynGlobalTimeTxIcvGeneration ⌈

Container Name	CanTSynGlobalTimeTxIcvGeneration
Parent Container	CanTSynGlobalTimeMaster
Upstream mapping defined	[10, CP_TPS_SystemTemplate]
Description	This container collects configuration that shall be used for ICV generation.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanTSynIcvGenerationBase	1	[ECUC_CanTSyn_00062]
CanTSynIcvGenerationTimeout	1	[ECUC_CanTSyn_00065]
CanTSynIcvTxLength	1	[ECUC_CanTSyn_00063]
CanTSynTxAuthenticationBuildAttempts	1	[ECUC_CanTSyn_00086]
CanTSynIcvGenerationFvIdRef	0..1	[ECUC_CanTSyn_00061]
CanTSynIcvGenerationJobRef	1	[ECUC_CanTSyn_00064]

No Included Containers

⌋

[ECUC_CanTSyn_00062] Definition of EcucEnumerationParamDef CanTSynIcvGenerationBase ⌈

Parameter Name	CanTSynIcvGenerationBase	
Parent Container	CanTSynGlobalTimeTxIcvGeneration	
Upstream mapping defined	–	
Description	Symmetric or asymmetric cryptography selection for the ICV generation	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ICV_MAC	Symmetric cryptography selection for the ICV generation.
	ICV_SIGNATURE	Asymmetric cryptography selection for the ICV generation.
Pre-Compile only	true	

⌋

[ECUC_CanTSyn_00065] Definition of EcucFloatParamDef CanTSynIcvGenerationTimeout ⌈

Parameter Name	CanTSynIcvGenerationTimeout	
Parent Container	CanTSynGlobalTimeTxIcvGeneration	
Upstream mapping defined	–	
Description	Timeout of ICV generation (respective CSM job completion in asynchronous behavior). A value of 0 disables the ICV timeout monitoring. Unit: Seconds	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF[	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_CanTSyn_00063] Definition of EcucIntegerParamDef CanTSynIcvTx Length ⌈

Parameter Name	CanTSynIcvTxLength	
Parent Container	CanTSynGlobalTimeTxIcvGeneration	
Upstream mapping defined	–	
Description	Length of ICV to be transmitted within Follow_Up Message on the bus (in bytes).	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 54	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_CanTSyn_00086] Definition of EcucIntegerParamDef CanTSynTxAuthenticationBuildAttempts ⌈

Parameter Name	CanTSynTxAuthenticationBuildAttempts	
Parent Container	CanTSynGlobalTimeTxIcvGeneration	
Upstream mapping defined	–	
Description	This parameter specifies the number of authentication build attempts that are to be carried out when the generation of the ICV failed for a given FUP message. If zero is set, then only one ICV build attempt is done.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	0	
Pre-Compile only	true	

⌋

[ECUC_CanTSyn_00061] Definition of EcucReferenceDef CanTSynIcvGeneration FvIdRef ⌈

Parameter Name	CanTSynIcvGenerationFvIdRef
Parent Container	CanTSynGlobalTimeTxIcvGeneration
Upstream mapping defined	[10, CP_TPS_SystemTemplate]
Description	This represents the reference to the FV taken to generate the ICV generation.
Multiplicity	0..1
Type	Symbolic name reference to StbMFreshnessValue
Pre-Compile only	true

⌋

[ECUC_CanTSyn_00064] Definition of EcucReferenceDef CanTSyncvGenerationJobRef ⌈

Parameter Name	CanTSyncvGenerationJobRef
Parent Container	CanTSynGlobalTimeTxIcvGeneration
Upstream mapping defined	[10, CP_TPS_SystemTemplate]
Description	This represents the reference to the CSM job to fetch the CSM job ID.
Multiplicity	1
Type	Symbolic name reference to CsmJob
Pre-Compile only	true

⌋

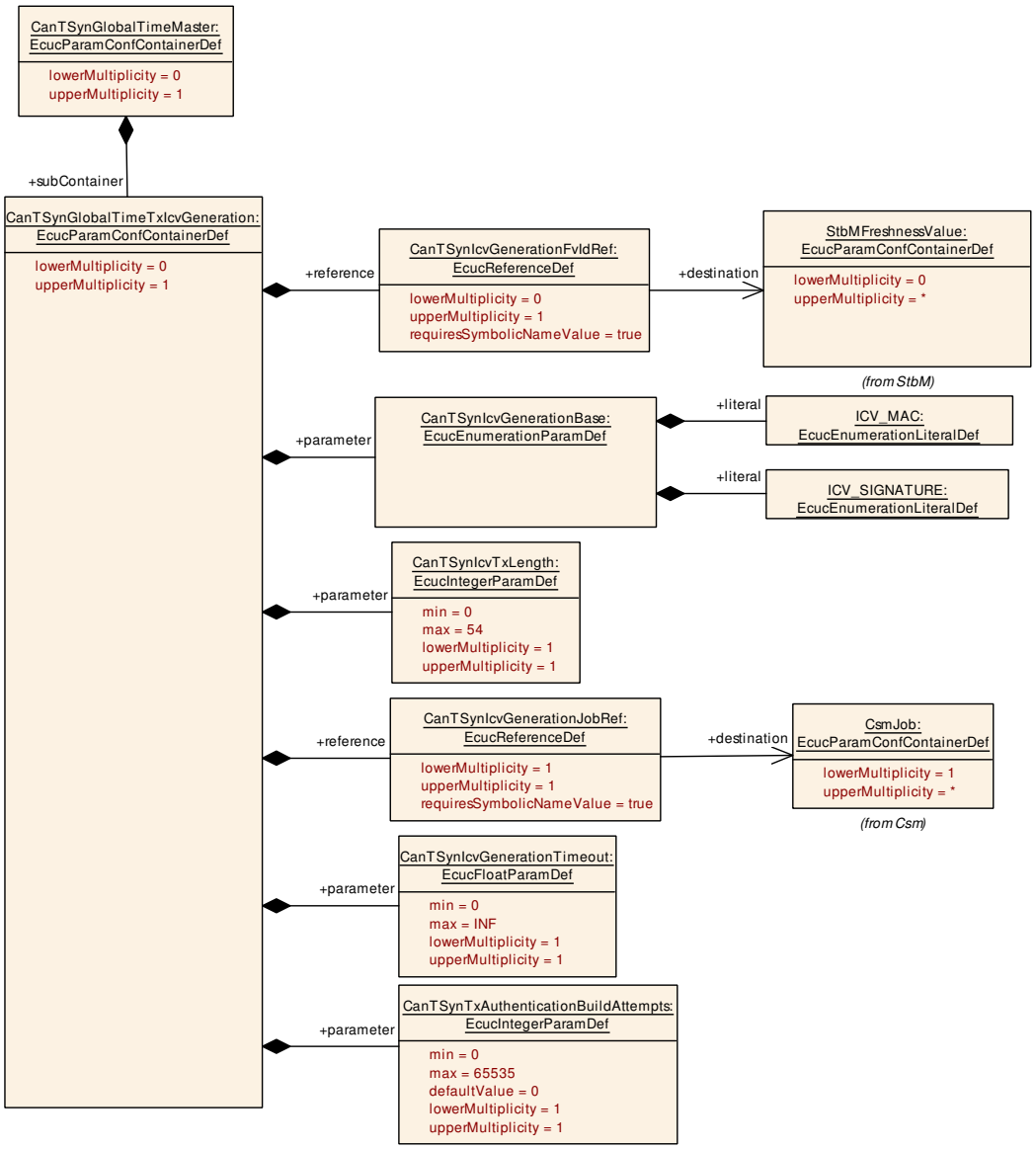


Figure 10.7: CanTSynGlobalTimeTxIcvGeneration

10.2.13 CanTSynGlobalTimeSlave

[ECUC_CanTSyn_00012] Definition of EcucParamConfContainerDef CanTSyn GlobalTimeSlave ⌈

Container Name	CanTSynGlobalTimeSlave	
Parent Container	CanTSynGlobalTimeDomain	
Upstream mapping defined	[10, CP_TPS_SystemTemplate]	
Description	Configuration of a Time Slave for a Time Domain (refer to parent container). If Can TSynGlobalTimeSlave container exists, the local ECU acts as a Time Slave for the Time Domain.	
Multiplicity	0..1	
Post-Build Multiplicity	-	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanTSynGlobalTimeFollowUpTimeout	1	[ECUC_CanTSyn_00006]
CanTSynGlobalTimeRxDebounceTime	1	[ECUC_CanTSyn_00068]
CanTSynGlobalTimeSequenceCounterHysteresis	1	[ECUC_CanTSyn_00053]
CanTSynGlobalTimeSequenceCounterJumpWidth	1	[ECUC_CanTSyn_00011]
CanTSynRxCrcValidated	1	[ECUC_CanTSyn_00021]
CanTSynRxIcvVerificationType	1	[ECUC_CanTSyn_00075]

Included Containers		
Container Name	**Multiplicity**	**Description**
CanTSynGlobalTimeRxIcv Verification	0..1	This container collects configuration required for ICV verification.
CanTSynGlobalTimeSlavePdu	1	This container encloses the configuration of the PDU that is supposed to contain the global time information. Supported MetaData entry: TIMETUPLE_TYPE_PTR

⌋

[ECUC_CanTSyn_00006] Definition of EcucFloatParamDef CanTSynGlobalTime FollowUpTimeout ⌈

Parameter Name	CanTSynGlobalTimeFollowUpTimeout	
Parent Container	CanTSynGlobalTimeSlave	
Upstream mapping defined	[10, CP_TPS_SystemTemplate]	
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 Value	-	Selectable

⌋

[ECUC_CanTSyn_00068] Definition of EcucFloatParamDef CanTSynGlobalTimeRxDebounceTime ⌈

Parameter Name	CanTSynGlobalTimeRxDebounceTime	
Parent Container	CanTSynGlobalTimeSlave	
Upstream mapping defined	–	
Description	This represents the configuration of a RX debounce time for the Sync and FUP. Unit: seconds	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 4]	
Default value	–	
Post-Build Value	-	Selectable

⌋

[ECUC_CanTSyn_00053] Definition of EcucIntegerParamDef CanTSynGlobalTimeSequenceCounterHysteresis ⌈

Parameter Name	CanTSynGlobalTimeSequenceCounterHysteresis	
Parent Container	CanTSynGlobalTimeSlave	
Upstream mapping defined	–	
Description	CanTSynGlobalTimeSequenceCounterHysteresis specifies the number of consecutive valid message pairs that are required by the Time Slave while being in Timeout state until a Time Tuple is forwarded to the StbM.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 15	
Default value	0	
Post-Build Value	-	Selectable

⌋

[ECUC_CanTSyn_00011] Definition of EcucIntegerParamDef CanTSynGlobalTimeSequenceCounterJumpWidth ⌈

Parameter Name	CanTSynGlobalTimeSequenceCounterJumpWidth	
Parent Container	CanTSynGlobalTimeSlave	
Upstream mapping defined	[10, CP_TPS_SystemTemplate]	
Description	The SequenceCounterJumpWidth specifies the maximum allowed gap of the Sequence Counter between two SYNC messages.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 15	
Default value	0	
Post-Build Value	-	Selectable

⌋

[ECUC_CanTSyn_00021] Definition of EcucEnumerationParamDef CanTSynRx CrcValidated ⌈

Parameter Name	CanTSynRxCrcValidated	
Parent Container	CanTSynGlobalTimeSlave	
Upstream mapping defined	[10, CP_TPS_SystemTemplate]	
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 Value	-	Selectable

⌋

[ECUC_CanTSyn_00075] Definition of EcucEnumerationParamDef CanTSynRx IcvVerificationType ⌈

Parameter Name	CanTSynRxIcvVerificationType	
Parent Container	CanTSynGlobalTimeSlave	
Upstream mapping defined	–	
Description	This parameter controls whether or not ICV verification shall be supported.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ICV_IGNORED	The Timesync module accepts Time Synchronization messages, which are ICV secured (without actually validating the ICV) and those which are not ICV secured. That means, the Timesync module ignores the ICV.
	ICV_NOT_VERIFIED	The Timesync module accepts only Time Synchronization messages, which are not ICV secured. All other Time Synchronization messages are ignored.
	ICV_OPTIONAL	The Timesync module accepts only Time Synchronization messages which are not ICV secured and Time Synchronization messages which are ICV secured and have the correct ICV. All other Time Synchronization messages are ignored.

▽

△

	ICV_VERIFIED	The Timesync module accepts only Time Synchronization messages, which are ICV secured and have the correct ICV. All other Time Synchronization messages are ignored.
Pre-Compile only	true	

⌋

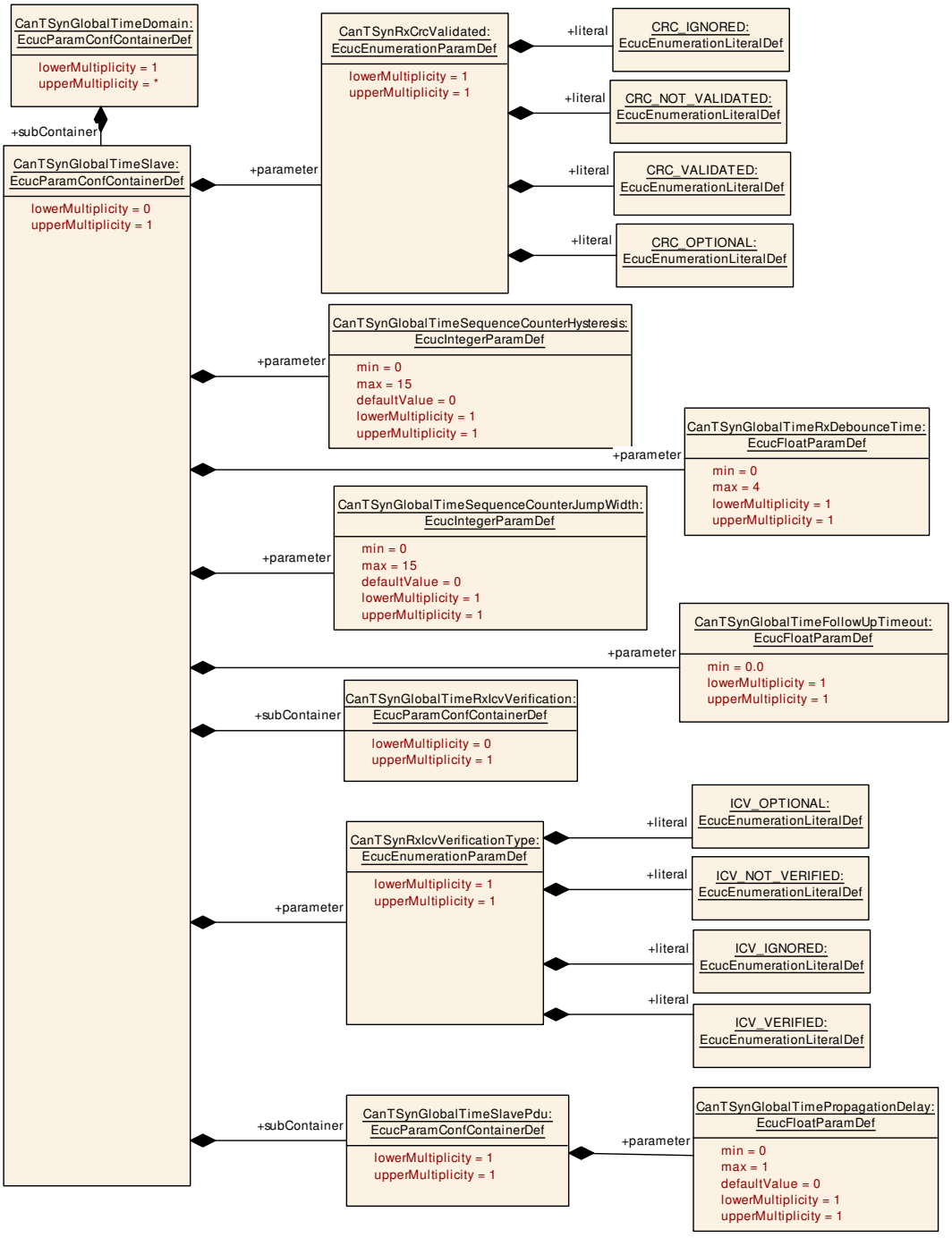


Figure 10.8: CanTSynGlobalTimeSlave

10.2.14 CanTSynGlobalTimeSlavePdu

[ECUC_CanTSyn_00014] Definition of EcucParamConfContainerDef CanTSyn GlobalTimeSlavePdu ⌈

Container Name	CanTSynGlobalTimeSlavePdu
Parent Container	CanTSynGlobalTimeSlave
Upstream mapping defined	–
Description	This container encloses the configuration of the PDU that is supposed to contain the global time information. Supported MetaData entry: TIMETUPLE_TYPE_PTR
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanTSynGlobalTimePropagationDelay	1	[ECUC_CanTSyn_00089]
CanTSynGlobalTimeSlaveHandleId	1	[ECUC_CanTSyn_00013]
CanTSynGlobalTimePduRef	1	[ECUC_CanTSyn_00027]

No Included Containers

⌋

[ECUC_CanTSyn_00089] Definition of EcucFloatParamDef CanTSynGlobalTime PropagationDelay ⌈

Parameter Name	CanTSynGlobalTimePropagationDelay	
Parent Container	CanTSynGlobalTimeSlavePdu	
Upstream mapping defined	–	
Description	Physical Layer specific propagation delay of a SYNC message to be compensated upon SYNC reception. Unit: seconds.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 1]	
Default value	0	
Post-Build Value	-	Selectable

⌋

[ECUC_CanTSyn_00013] Definition of EcucIntegerParamDef CanTSynGlobal TimeSlaveHandleId ⌈

Parameter Name	CanTSynGlobalTimeSlaveHandleId
Parent Container	CanTSynGlobalTimeSlavePdu
Upstream mapping defined	–
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	–	
With Auto	true	
Pre-Compile only	true	

⌋

For parameter table [ECUC_CanTSyn_00027] CanTSynGlobalTimePduRef, see definition below container CanTSynGlobalTimeMasterPdu.

10.2.15 CanTSynGlobalTimeRxIcvVerification

[ECUC_CanTSyn_00076] Definition of EcucParamConfContainerDef CanTSynGlobalTimeRxIcvVerification ⌈

Container Name	CanTSynGlobalTimeRxIcvVerification
Parent Container	CanTSynGlobalTimeSlave
Upstream mapping defined	[10, CP_TPS_SystemTemplate]
Description	This container collects configuration required for ICV verification.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CanTSynIcvRxLength	1	[ECUC_CanTSyn_00079]
CanTSynIcvVerificationAttempts	1	[ECUC_CanTSyn_00082]
CanTSynIcvVerificationBase	1	[ECUC_CanTSyn_00078]
CanTSynIcvVerificationTimeout	1	[ECUC_CanTSyn_00081]
CanTSynRxAuthenticationBuildAttempts	1	[ECUC_CanTSyn_00084]
CanTSynIcvVerificationFvIdRef	0..1	[ECUC_CanTSyn_00077]
CanTSynIcvVerificationJobRef	1	[ECUC_CanTSyn_00080]

No Included Containers

⌋

[ECUC_CanTSyn_00079] Definition of EcucIntegerParamDef CanTSynIcvRxLength ⌈

Parameter Name	CanTSynIcvRxLength
Parent Container	CanTSynGlobalTimeRxIcvVerification
Upstream mapping defined	–
Description	Length of ICV to be used for verification of received ICV within FUP Message in Bytes.

▽

△

Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 54	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_CanTSyn_00082] Definition of EcucIntegerParamDef CanTSynIcvVerificationAttempts ⌈

Parameter Name	CanTSynIcvVerificationAttempts	
Parent Container	CanTSynGlobalTimeRxIcvVerification	
Upstream mapping defined	[10, CP_TPS_SystemTemplate]	
Description	This parameter specifies the number of ICV verification attempts that are to be carried out when the verification of the ICV failed for a given FUP message. If zero is set, then only one ICV verification attempt is done.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	0	
Pre-Compile only	true	

⌋

[ECUC_CanTSyn_00078] Definition of EcucEnumerationParamDef CanTSynIcvVerificationBase ⌈

Parameter Name	CanTSynIcvVerificationBase	
Parent Container	CanTSynGlobalTimeRxIcvVerification	
Upstream mapping defined	–	
Description	Symmetric or asymmetric cryptography selection for the ICV generation	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ICV_MAC	Symmetric cryptography selection for the ICV verification.
	ICV_SIGNATURE	Asymmetric cryptography selection for the ICV verification.
Pre-Compile only	true	

⌋

[ECUC_CanTSyn_00081] Definition of EcucFloatParamDef CanTSynIcvVerificationTimeout ⌈

Parameter Name	CanTSynIcvVerificationTimeout
Parent Container	CanTSynGlobalTimeRxIcvVerification
Upstream mapping defined	–

▽

△

Description	Timeout of ICV generation (respective CSM job completion in asynchronous behavior). A value of 0 disables the ICV timeout monitoring. Unit: Seconds	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF[	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_CanTSyn_00084] Definition of EcucIntegerParamDef CanTSynRxAuthenticationBuildAttempts ⌈

Parameter Name	CanTSynRxAuthenticationBuildAttempts	
Parent Container	CanTSynGlobalTimeRxIcvVerification	
Upstream mapping defined	–	
Description	This parameter specifies the number of authentication build attempts that are to be carried out when the verification of the ICV failed for a given FUP message. If zero is set, then only one ICV verification attempt is done.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	0	
Pre-Compile only	true	

⌋

[ECUC_CanTSyn_00077] Definition of EcucReferenceDef CanTSynIcvVerificationFvIdRef ⌈

Parameter Name	CanTSynIcvVerificationFvIdRef
Parent Container	CanTSynGlobalTimeRxIcvVerification
Upstream mapping defined	[10, CP_TPS_SystemTemplate]
Description	This represents the reference to the FV taken to generate the ICV generation.
Multiplicity	0..1
Type	Symbolic name reference to StbMFreshnessValue
Pre-Compile only	true

⌋

[ECUC_CanTSyn_00080] Definition of EcucReferenceDef CanTSynIcvVerificationJobRef ⌈

Parameter Name	CanTSynIcvVerificationJobRef
Parent Container	CanTSynGlobalTimeRxIcvVerification
Upstream mapping defined	[10, CP_TPS_SystemTemplate]
Description	This represents the reference to the CSM job to fetch the CSM job ID.
Multiplicity	1

▽

△

Type	Symbolic name reference to CsmJob
Pre-Compile only	true

⌋

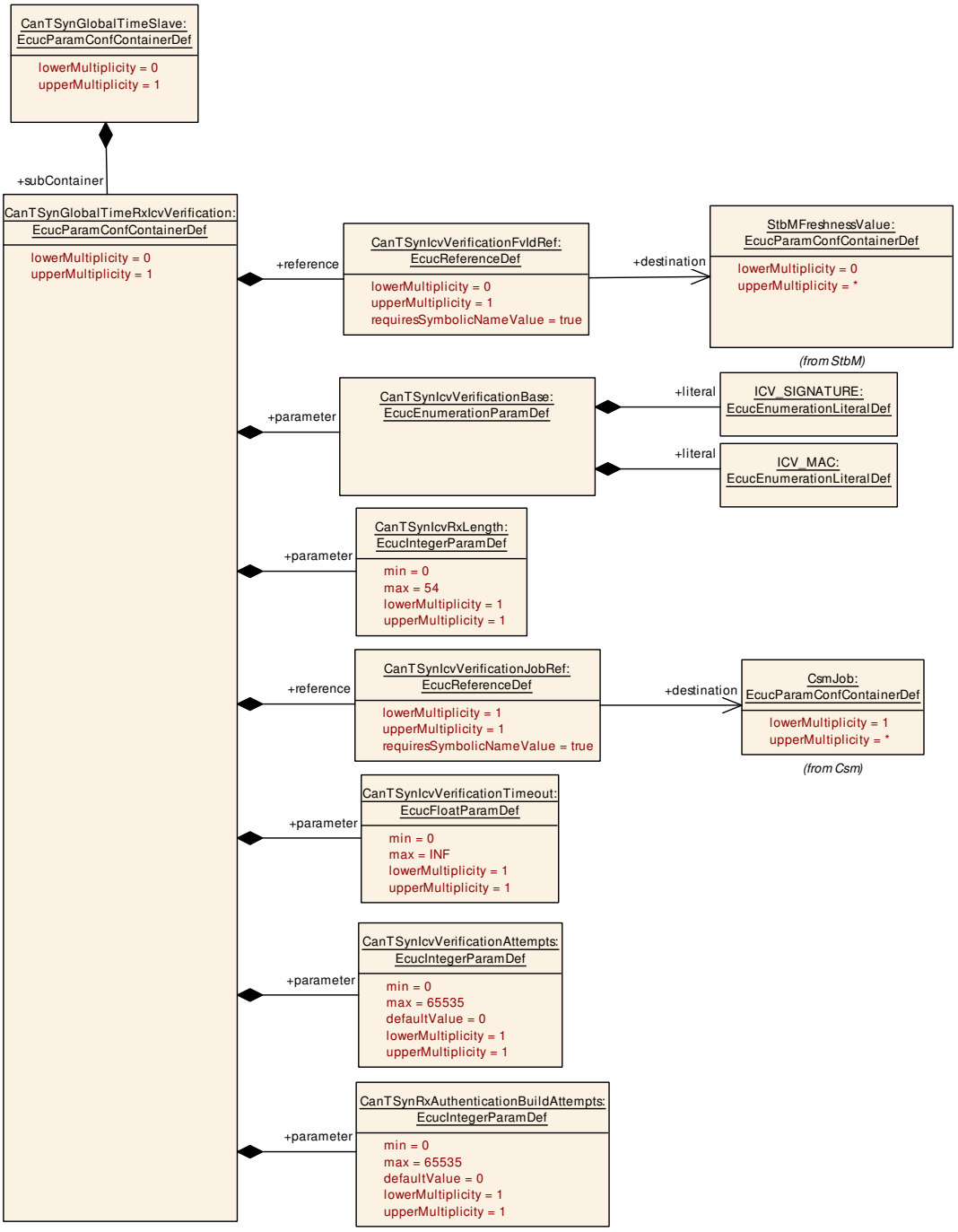


Figure 10.9: CanTSynGlobalTimerxIcvVerification

10.3 Constraints

[SWS_CanTSyn_CONSTR_00001] ⌈If the CSM job used to generate ICV is configured in synchronous behaviour, the CanTSynIcvGenerationTimeout shall be set to 0.⌋

[SWS_CanTSyn_CONSTR_00002] ⌈If the CSM job used to verify ICV is configured in synchronous behavior, the CanTSynIcvVerificationTimeout shall be set to 0.⌋

[SWS_CanTSyn_CONSTR_00003] **CanTSynGlobalTimeMasterPdu and CanTSynGlobalTimeSlavePdu constraint for keeping the local buffer**

Status: DRAFT

⌈The configuration of CanTSynGlobalTimeMasterPdu and CanTSynGlobalTimeSlavePdu shall refer to PDUs where KeepLocalPduBuffer is set to FALSE.

Otherwise the configuration shall be rejected as invalid.⌋

10.4 Published Information

For details refer to [4] Chapter 10.3 *"Published Information"*.

A Not applicable requirements

[SWS_CanTSyn_NA_00999]

Upstream requirements: RS_TS_00002, RS_TS_00005, RS_TS_00006, RS_TS_00007, RS_TS_00008, RS_TS_00009, RS_TS_00010, RS_TS_00011, RS_TS_00014, RS_TS_00015, RS_TS_00016, RS_TS_00017, RS_TS_00018, RS_TS_00019, RS_TS_00021, RS_TS_00024, RS_TS_00025, RS_TS_00026, RS_TS_00029, RS_TS_00030, RS_TS_00031, RS_TS_00032, RS_TS_00033, RS_TS_00035, RS_TS_00037, RS_TS_00038, RS_TS_00039, RS_TS_00040, RS_TS_00041, RS_TS_00042, RS_TS_00043, RS_TS_20039, RS_TS_20040, RS_TS_20041, RS_TS_20042, RS_TS_20043, RS_TS_20045, RS_TS_20046, RS_TS_20047, RS_TS_20048, RS_TS_20051, RS_TS_20052, RS_TS_20053, RS_TS_20054, RS_TS_20058, RS_TS_20059, RS_TS_20060, RS_TS_20061, RS_TS_20062, RS_TS_20066, RS_TS_20069, RS_TS_20071, RS_TS_20072, RS_TS_20074, RS_TS_20075

⌈These requirements on Time Synchronization from the RS Time Synchronization [2] are not applicable to `CanTSyn`, because they are allocated either to other network specific Time Sync modules (CAN, FlexRay) or to the Synchronized Time-Base Manager (StbM) module.⌋

[SWS_CanTSyn_NA_00998]

Upstream requirements: SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00375, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00422, SRS_BSW_00425, SRS_BSW_00432, SRS_BSW_00461, SRS_BSW_00469, SRS_BSW_00471, SRS_BSW_00472

⌈These requirements of the [5, SRS BSWGeneral] are not traced to CanTSyn spec items, because they refer to functionality that is not supported by the CanTSyn.⌋

[SWS_CanTSyn_NA_00997]

Upstream requirements: SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00395, SRS_BSW_00403, SRS_BSW_00478

⌈These requirements of the [5, SRS BSWGeneral] are not traced to CanTSyn spec items, because they apply only to EcuC elements.⌋

Note: EcuC elements do not support trace links

[SWS_CanTSyn_NA_00996]

Upstream requirements: SRS_BSW_00369, SRS_BSW_00383, SRS_BSW_00384, SRS_BSW_00419

⌈These requirements of the [5, SRS BSWGeneral] are not traced to CanTSyn spec items, because they do not apply to specific spec items of the CanTSyn.⌋

Note: These requirements of the [5, SRS BSWGeneral] are generic in nature and would affect all or very many requirements (e.g. non-functional requirement like naming conventions) or cannot be traced to any requirement at all (but just explanatory chapters of the SWS).

B Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These constraints and specification items do not appear as hyperlinks in the document.

B.1 Traceable item history of this document according to AUTOSAR Release R26-11

B.1.1 Added Specification Items in R26-11

none

B.1.2 Changed Specification Items in R26-11

[ECUC_CanTSyn_00001] [ECUC_CanTSyn_00002] [ECUC_CanTSyn_00003]
[ECUC_CanTSyn_00004] [ECUC_CanTSyn_00008] [ECUC_CanTSyn_00009]
[ECUC_CanTSyn_00013] [ECUC_CanTSyn_00014] [ECUC_CanTSyn_00019]
[ECUC_CanTSyn_00022] [ECUC_CanTSyn_00028] [ECUC_CanTSyn_00031]
[ECUC_CanTSyn_00050] [ECUC_CanTSyn_00051] [ECUC_CanTSyn_00054]
[ECUC_CanTSyn_00055] [ECUC_CanTSyn_00056] [ECUC_CanTSyn_00057]
[ECUC_CanTSyn_00058] [ECUC_CanTSyn_00059] [ECUC_CanTSyn_00060]
[ECUC_CanTSyn_00061] [ECUC_CanTSyn_00062] [ECUC_CanTSyn_00063]
[ECUC_CanTSyn_00064] [ECUC_CanTSyn_00065] [ECUC_CanTSyn_00075]
[ECUC_CanTSyn_00076] [ECUC_CanTSyn_00077] [ECUC_CanTSyn_00078]
[ECUC_CanTSyn_00079] [ECUC_CanTSyn_00080] [ECUC_CanTSyn_00081]
[ECUC_CanTSyn_00082] [ECUC_CanTSyn_00083] [ECUC_CanTSyn_00084]
[ECUC_CanTSyn_00086] [ECUC_CanTSyn_00088] [ECUC_CanTSyn_00111]
[SWS_CanTSyn_00090]

B.1.3 Deleted Specification Items in R26-11

[ECUC_CanTSyn_00023] [ECUC_CanTSyn_00042] [SWS_CanTSyn_00094]

B.1.4 Added Constraints in R26-11

none

B.1.5 Changed Constraints in R26-11

none

B.1.6 Deleted Constraints in R26-11

none

B.2 Traceable item history of this document according to AUTOSAR Release R25-11

B.2.1 Added Specification Items in R25-11

[SWS_CanTSyn_00238]

B.2.2 Changed Specification Items in R25-11

[ECUC_CanTSyn_00003] [ECUC_CanTSyn_00004] [ECUC_CanTSyn_00005] [ECUC_CanTSyn_00007] [ECUC_CanTSyn_00009] [ECUC_CanTSyn_00012] [ECUC_CanTSyn_00014] [ECUC_CanTSyn_00024] [ECUC_CanTSyn_00025] [ECUC_CanTSyn_00028] [ECUC_CanTSyn_00031] [ECUC_CanTSyn_00042] [ECUC_CanTSyn_00056] [ECUC_CanTSyn_00060] [ECUC_CanTSyn_00076] [SWS_CanTSyn_00028] [SWS_CanTSyn_00090] [SWS_CanTSyn_00106] [SWS_CanTSyn_00118] [SWS_CanTSyn_00120] [SWS_CanTSyn_00149] [SWS_CanTSyn_00150] [SWS_CanTSyn_00161] [SWS_CanTSyn_00168] [SWS_CanTSyn_00175] [SWS_CanTSyn_00182] [SWS_CanTSyn_00193] [SWS_CanTSyn_00200] [SWS_CanTSyn_00207]

B.2.3 Deleted Specification Items in R25-11

[SWS_CanTSyn_00007] [SWS_CanTSyn_00008] [SWS_CanTSyn_00010] [SWS_CanTSyn_00011] [SWS_CanTSyn_00012] [SWS_CanTSyn_00013] [SWS_CanTSyn_00015] [SWS_CanTSyn_00016] [SWS_CanTSyn_00017] [SWS_CanTSyn_00018] [SWS_CanTSyn_00025] [SWS_CanTSyn_00026] [SWS_CanTSyn_00027] [SWS_CanTSyn_00029] [SWS_CanTSyn_00031] [SWS_CanTSyn_00032] [SWS_CanTSyn_00033] [SWS_CanTSyn_00048] [SWS_CanTSyn_00049] [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_00075] [SWS_CanTSyn_00076] [SWS_CanTSyn_00084] [SWS_CanTSyn_00085] [SWS_CanTSyn_00086] [SWS_CanTSyn_00087] [SWS_CanTSyn_00088] [SWS_CanTSyn_00109] [SWS_CanTSyn_00110] [SWS_CanTSyn_00124] [SWS_CanTSyn_00125] [SWS_CanTSyn_00155] [SWS_CanTSyn_00156] [SWS_CanTSyn_00159] [SWS_CanTSyn_00178] [SWS_CanTSyn_00179] [SWS_CanTSyn_00180] [SWS_CanTSyn_00181] [SWS_CanTSyn_00183] [SWS_CanTSyn_00206] [SWS_CanTSyn_00221] [SWS_CanTSyn_00222] [SWS_CanTSyn_00223] [SWS_CanTSyn_00224] [SWS_CanTSyn_00225] [SWS_CanTSyn_00226] [SWS_

CanTSyn_00227] [SWS_CanTSyn_00231] [SWS_CanTSyn_00232] [SWS_CanTSyn_00233] [SWS_CanTSyn_00234] [SWS_CanTSyn_00235] [SWS_CanTSyn_00236] [SWS_CanTSyn_00237]

B.2.4 Added Constraints in R25-11

none

B.2.5 Changed Constraints in R25-11

none

B.2.6 Deleted Constraints in R25-11

none

B.3 Traceable item history of this document according to AUTOSAR Release R24-11

B.3.1 Added Specification Items in R24-11

[ECUC_CanTSyn_00089] [SWS_CanTSyn_00234] [SWS_CanTSyn_00235] [SWS_CanTSyn_00236] [SWS_CanTSyn_00237] [SWS_CanTSyn_92000] [SWS_CanTSyn_92001] [SWS_CanTSyn_92002] [SWS_CanTSyn_92003]

B.3.2 Changed Specification Items in R24-11

[ECUC_CanTSyn_00012] [ECUC_CanTSyn_00014] [SWS_CanTSyn_00010] [SWS_CanTSyn_00031] [SWS_CanTSyn_00048] [SWS_CanTSyn_00049] [SWS_CanTSyn_00054] [SWS_CanTSyn_00055] [SWS_CanTSyn_00056] [SWS_CanTSyn_00085] [SWS_CanTSyn_00086] [SWS_CanTSyn_00106] [SWS_CanTSyn_00115] [SWS_CanTSyn_00133] [SWS_CanTSyn_00136] [SWS_CanTSyn_00140] [SWS_CanTSyn_00145] [SWS_CanTSyn_00148] [SWS_CanTSyn_00149] [SWS_CanTSyn_00154] [SWS_CanTSyn_00159] [SWS_CanTSyn_00161] [SWS_CanTSyn_00165] [SWS_CanTSyn_00166] [SWS_CanTSyn_00167] [SWS_CanTSyn_00168] [SWS_CanTSyn_00175] [SWS_CanTSyn_00176] [SWS_CanTSyn_00177] [SWS_CanTSyn_00188] [SWS_CanTSyn_00189] [SWS_CanTSyn_00190] [SWS_CanTSyn_00193] [SWS_CanTSyn_00199] [SWS_CanTSyn_00204] [SWS_CanTSyn_00210] [SWS_CanTSyn_00211] [SWS_CanTSyn_00221] [SWS_CanTSyn_00222] [SWS_CanTSyn_00223] [SWS_CanTSyn_00224] [SWS_CanTSyn_00225] [SWS_

CanTSyn_00226] [SWS_CanTSyn_00227] [SWS_CanTSyn_00230] [SWS_CanTSyn_00232] [SWS_CanTSyn_00233]

B.3.3 Deleted Specification Items in R24-11

[ECUC_CanTSyn_00026] [ECUC_CanTSyn_00034] [ECUC_CanTSyn_00035] [ECUC_CanTSyn_00036] [ECUC_CanTSyn_00037] [ECUC_CanTSyn_00038] [ECUC_CanTSyn_00039] [ECUC_CanTSyn_00041] [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_00078] [SWS_CanTSyn_00079] [SWS_CanTSyn_00111] [SWS_CanTSyn_00112] [SWS_CanTSyn_00113] [SWS_CanTSyn_00114] [SWS_CanTSyn_00116] [SWS_CanTSyn_00119] [SWS_CanTSyn_00123] [SWS_CanTSyn_00126] [SWS_CanTSyn_00127] [SWS_CanTSyn_00128] [SWS_CanTSyn_00129] [SWS_CanTSyn_00130] [SWS_CanTSyn_00131] [SWS_CanTSyn_00132] [SWS_CanTSyn_00143] [SWS_CanTSyn_00157] [SWS_CanTSyn_00158] [SWS_CanTSyn_00160] [SWS_CanTSyn_00162] [SWS_CanTSyn_00184] [SWS_CanTSyn_00185] [SWS_CanTSyn_00186] [SWS_CanTSyn_00187] [SWS_CanTSyn_00205] [SWS_CanTSyn_00208] [SWS_CanTSyn_00209]

B.3.4 Added Constraints in R24-11

[SWS_CanTSyn_CONSTR_00003]

B.3.5 Changed Constraints in R24-11

none

B.3.6 Deleted Constraints in R24-11

none

B.4 Traceable item history of this document according to AUTOSAR Release R23-11

B.4.1 Added Specification Items in R23-11

[SWS_CanTSyn_00207] [SWS_CanTSyn_00208] [SWS_CanTSyn_00209] [SWS_CanTSyn_00210] [SWS_CanTSyn_00211] [SWS_CanTSyn_00212] [SWS_CanTSyn_00213] [SWS_CanTSyn_00214] [SWS_CanTSyn_00215] [SWS_CanTSyn_00216] [SWS_CanTSyn_00217] [SWS_CanTSyn_00218] [SWS_CanTSyn_00219] [SWS_CanTSyn_00220] [SWS_CanTSyn_00221] [SWS_CanTSyn_00222] [SWS_CanTSyn_00223] [SWS_CanTSyn_00224] [SWS_CanTSyn_00225] [SWS_CanTSyn_00226] [SWS_CanTSyn_00227] [SWS_CanTSyn_00228] [SWS_CanTSyn_00229] [SWS_CanTSyn_00230] [SWS_CanTSyn_00231] [SWS_CanTSyn_00232] [SWS_CanTSyn_00233]

B.4.2 Changed Specification Items in R23-11

[SWS_CanTSyn_00010] [SWS_CanTSyn_00015] [SWS_CanTSyn_00016] [SWS_CanTSyn_00017] [SWS_CanTSyn_00018] [SWS_CanTSyn_00028] [SWS_CanTSyn_00031] [SWS_CanTSyn_00033] [SWS_CanTSyn_00038] [SWS_CanTSyn_00041] [SWS_CanTSyn_00042] [SWS_CanTSyn_00064] [SWS_CanTSyn_00068] [SWS_CanTSyn_00069] [SWS_CanTSyn_00070] [SWS_CanTSyn_00078] [SWS_CanTSyn_00079] [SWS_CanTSyn_00086] [SWS_CanTSyn_00090] [SWS_CanTSyn_00111] [SWS_CanTSyn_00112] [SWS_CanTSyn_00114] [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_00126] [SWS_CanTSyn_00127] [SWS_CanTSyn_00128] [SWS_CanTSyn_00129] [SWS_CanTSyn_00139] [SWS_CanTSyn_00141] [SWS_CanTSyn_00143] [SWS_CanTSyn_00147] [SWS_CanTSyn_00148] [SWS_CanTSyn_00155] [SWS_CanTSyn_00156] [SWS_CanTSyn_00157] [SWS_CanTSyn_00158] [SWS_CanTSyn_00159] [SWS_CanTSyn_00160] [SWS_CanTSyn_00168] [SWS_CanTSyn_00175] [SWS_CanTSyn_00177] [SWS_CanTSyn_00182] [SWS_CanTSyn_00183] [SWS_CanTSyn_00193] [SWS_CanTSyn_00200] [SWS_CanTSyn_00204] [SWS_CanTSyn_00205] [SWS_CanTSyn_NA_00999]

B.4.3 Deleted Specification Items in R23-11

[SWS_CanTSyn_00192]

B.4.4 Added Constraints in R23-11

[SWS_CanTSyn_CONSTR_00002]

B.4.5 Changed Constraints in R23-11

none

B.4.6 Deleted Constraints in R23-11

none

C Mapping Tables

C.1 Mapping of Protocol Inputs

CanTSyn Protocol Value	CanTSyn Module Reference	Mapping Description
APP_TMRequestSyncTx		SYNC message transmission request according to [SWS_CanTSyn_00028] and [SWS_CanTSyn_00118]
APP_TMUserData	StbM_UserDataType.userByte1..3	User Data byte array of the Time Master as derived by StbM_GetCurrentTime via parameter userData.
APP_TMGatewayStatus	SYNC_TO_GATEWAY bit of timeBaseStatus	"Synchronized to Gateway" status flag of the Time Domain as derived by by the Time Master via StbM_GetCurrentTime and transmitted in the SGW bit
APP_TMMasterTimeAvailable	GLOBAL_TIME_BASE bit of timeBaseStatus	
DLL_EventSyncTxSuccess		Event that indicates a successful transmission of SYNC message
APP_TMEventSyncTxFailure		Event that indicates a failed transmission of SYNC message
EXT_TMEventFvCalcFailure		Event that indicates a Freshness Value calculation failure
EXT_TMEventFvCalcSuccess		Event that indicates a Freshness Value calculation succeeded
EXT_TMEventIcvCalcSuccess		Event that indicates a successful ICV calculation
EXT_TMEventIcvCalcFailure		Event that indicates a ICV calculation failure
EXT_TMFreshnessValue		
EXT_TSFreshnessValue	StbMTruncatedFreshnessValue parameter of StbM_GetRxFreshness	
EXT_TMIntegrityCheckValue	Csm_SignatureGenerate or Csm_MacGenerate	
EXT_TSIntegrityCheckValue	Csm_SignatureVerify or Csm_MacVerify	

Table C.1: Mapping of Protocol Inputs

C.2 Mapping of Protocol Outputs

CanTSyn Protocol Value	CanTSyn Module Reference	Mapping Description
APP_TSUserData	StbM_UserDataType.userByte1..3	User Data byte array of the Time Slave as forwarded by StbM_BusSetGlobalTime

CanTSyn Protocol Value	CanTSyn Module Reference	Mapping Description
APP_TSUserDataLength	StbM_UserDataType.userDataLength	Length of the User Data byte array APP_TSUserData of the Time Slave as forwarded by StbM_BusSetGlobalTime
APP_TSGatewayStatus	SYNC_TO_GATEWAY bit of timeBaseStatus	"Synchronized to Gateway" status flag of the Time Slave as received in the SGW bit and forwarded by StbM_BusSetGlobalTime.
DLL_EventSyncTxSuccess	CanTSyn_TxConfirmation	A return value E_OK for the API (if received within the 2 sec) indicates a successful transmission of a SYNC message
DLL_EventSyncTxFailure	CanTSyn_TxConfirmation	A return value E_NOT_OK for the API (or a timeout after 2 sec) indicates a transmission failure for a SYNC message
DLL_TransmitSyncMsg	LSduR_CanTSynTransmit	Transmission request for a SYNC message to the LSduR module
DLL_TransmitFupMsg	LSduR_CanTSynTransmit	Transmission request for a FUP message to the LSduR module

Table C.2: Mapping of Protocol Outputs

C.3 Mapping of Protocol Timing Values

CanTSyn Protocol Value	CanTSyn Module Reference	Mapping Description
$\Delta t_{FupRxTimeout}$		The timeout interval when waiting for a FUP message after a corresponding SYNC message has been successfully received. Configured by parameter CanTSynGlobalTimeFollowUpTimeout.
$\Delta t_{LinkDelay}$	T_{PD}	SYNC propagation delay as configured by CanTSynGlobalTimePropagationDelay CanTSynGlobalTimePropagationDelay
$\Delta t_{RxDebounceInterval}$		Time interval between the reception of 2 consecutive Time Synchronization messages. Configured by parameter CanTSynGlobalTimeRxDebounceTime

CanTSyn Protocol Value	CanTSyn Module Reference	Mapping Description
$\Delta t_{SlaveSyncLossTimeout}$		Timeout interval between 2 valid FUP messages on Time Slave side. Equivalent to the timeout interval between 2 invocation of StbM_BusSetGlobalTime. Configured by StbMSyncLoss-Timeout
$\Delta t_{SlaveRxOffset}$	(T2 - T1) in Chapter 7.5.2.1	Time offset from SYNC ingress to actually re-synchronizing the Time Slave's Local Time (which can only be done after the FUP message has been received)
$\Delta t_{SyncTxOffset}$	T4 in Chapter 7.4.2.1	Time offset from transmission request for SYNC message to actual egress of the SYNC mesage.
$\Delta t_{SyncTxTimeout}$	2 sec	The timeout interval when waiting for the transmission confirmation after transmitting a SYNC message. In the CanTSyn this is not configurable and set to 2 sec.
$\Delta t_{TxDebounceInterval}$	debounceCounter	Inhibit time interval between the transmission of Time Synchronition messages to avoid bursts on the CAN bus. Configured by parameter CanTSyn-GlobalTimeDebounceTime
$T_{SyncEgress}$	$T1_{VLT/CAN}$ in Chapter 7.4.2.1	Egress Timestamp of SYNC message on Time Master side (either derived by HW or SW time stamping)
$T_{SyncIngress}$	$T1_{VLT}$ in Chapter 7.5.2.1	Ingress Timestamp of SYNC message on Time Slave side
$t_{TMGlobalTime}$	TL in the StbM	Current Local Time (= Local Instance of the Global time), which is maintained in the StbM on Time Master side
$T_{TMGlobalTime}$	$T0_{SYNC}$ in Chapter 7.4.2.1	Timestamp of the Local Time sampled by the Time Master by StbM_GetCurrentTime before transmission of the SYNC message
$T_{TMGlobalTime_Sec}$	SyncTimeSec	Seconds part of $T0_{SYNC}$
$T_{TMGlobalTime_NSec}$	$T0_{SYNC_ns}$ in Chapter 7.4.2.1	Nanoseconds part of $T0_{SYNC}$
$t_{TMLocalTime}$	TV in the StbM	Current Virtual Local Time of a Time Master
$T_{TMLocalTime}$	$T0_{VLT}$ in Chapter 7.4.2.1	Timestamp of the Virtual Local Time sampled by the Time Master at the same time as $T0_{SYNC}$
$t_{TSGlobalTime}$	TL in the StbM	Current Local Time (= Local Instance of the Global time), which is maintained in the StbM on Time Slave side

CanTSyn Protocol Value	CanTSyn Module Reference	Mapping Description
$T_{TSGlobalTime}$	TG_{Rx} in Chapter 7.5.2.1	Time calculated by the Time Slave from the information SYNC and FUP message content to update Local Instance of the Global Time in the StbM via StbM_BusSetGlobalTime
$T_{TSGlobalTime_Sec}$	SyncTimeSec + OVS	Seconds part of TG_{Rx}
$T_{TSGlobalTime_NSec}$	SyncTimeSec + T_{PD}	Nanoseconds part of TG_{Rx}
$t_{TSLocalTime}$	TV in the StbM	(current) Virtual Local Time which is maintained in the StbM on Time Slave side
$T_{TSLocalTime_ReSync}$	TV_{Sync} in StbM	Virtual Local Time of the Time Slave, when Time Slave is re-synchronized to the Time Master

Table C.3: Mapping of Protocol Timing Values

C.4 Mapping of Protocol Parameters

C.4.1 Mapping of Protocol Parameters of a Time Domain

Protocol Parameter	Module Parameter	Mapping Description
Cfg_TDDomainId	CanTSynGlobalTimeDomainId	
Cfg_TDCanId	CanTSynGlobalTimePduRef and CanTSynGlobalTimePduRef	
Cfg_TDSyncMsgDataIDList	CanTSynGlobalTimeSyncDataIDList	
Cfg_TDFupMsgDataIDList	CanTSynGlobalTimeFupDataIDList	

Table C.4: Mapping of Protocol Parameters of a Time Domain

C.4.2 Mapping of Protocol Parameters of the Time Master of a Time Domain

Protocol Parameter	Module Parameter	Mapping Description
Cfg_TMCrcProtected	CanTSynGlobalTimeTxCrcSecured	
Cfg_TMIcvSecured	CanTSynGlobalTimeTxIcvSecured	
Cfg_TMDebounceInterval	CanTSynGlobalTimeDebounceTime	
	Not configurable	Set to a fixed value of 2 sec. This is to avoid an overflow of the OVS value in the FUP message. Range: 0 .. 3. Unit: sec

Protocol Parameter	Module Parameter	Mapping Description
Cfg_TMIcvLength	CanTSynIcvTxLength	
Cfg_TMIcvGeneration-Algorithm	CanTSynIcvGeneration-Base	
Cfg_TMFreshnessValue-Support	CanTSynIcvGenerationFv-IdRef	
Cfg_TMFreshnessValue-Length	StbMFreshnessValue-Length	

Table C.5: Mapping of Protocol Parameters of the Time Master of a Time Domain

C.4.3 Mapping of Protocol Parameters of a Time Slave of a Time Domain

Protocol Parameter	Module Parameter	Mapping Description
Cfg_TSCrcVerified	CanTSynRxCrcValidated	
Cfg_TSIcvVerified	CanTSynRxIcv-VerificationType	
Cfg_TSSequenceCounter-JumpWidth	CanTSynGlobalTime-SequenceCounterJump-Width	
Cfg_TSSequenceCounter-Hysteresis	CanTSynGlobalTime-SequenceCounter-Hysteresis	
Cfg_TSDebounceInterval	CanTSynGlobalTimeRx-DebounceTime	
Cfg_TSFupRxTimeout	CanTSynGlobalTime-FollowUpTimeout	
Cfg_TSSyncLossTimeout	StbMSyncLossTimeout	
Cfg_TSLinkDelay	CanTSynGlobalTime-PropagationDelay	
Cfg_TMFreshnessValue-Support	CanTSynIcvVerification-FvIdRef	
Cfg_TMFreshnessValue-Length	StbMFreshnessValue-Length	
Cfg_TMIcvGeneration-Algorithm	CanTSynIcvGeneration-Base	

Table C.6: Mapping of Protocol Parameters of a Time Slave of a Time Domain