Document Title	Specification of Time Synchronization over Ethernet
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	676

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.1

Document Change History

Date	Release	Changed by	Change Description
2017-12-08	4.3.1	AUTOSAR Release Management	• Clarification of handling of unexpected Sub-TLVs • Clarification for configuration parameter • Clarification of handling FUP messages
2016-11-30	4.3.0	AUTOSAR Release Management	• Resident time compensation for switches added • AUTOSAR specific TLV added • Interface to StbM and EthIf reworked (incl. support for immediate Timesync message transmission) • Various enhancements and corrections (e.g. postbuild configuration)
2015-07-31	4.2.2	AUTOSAR Release Management	• <Bus>TSyn_SetTransmissionMode changed to return "void" • Call of StbM_BusSetGlobalTime() added - sequence diagrams corrected • 'const' added to input arguments passed by pointer
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

The EthTSyn module handles the distribution of time information over Ethernet.

The Ethernet mechanism is based on existing PTP (Precision Time Protocol) mechanisms that are described in standards like IEEE1588 (IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems) and IEEE802.1AS (Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks).

IEEE802.1AS, also known as gPTP (generalized Precision Time Protocol), can be seen as a profile (or subset) for using IEEE1588.

However, neither IEEE1588 nor IEEE802.1AS have been developed considering automotive requirements. Therefore, the Time Synchronization over Ethernet uses the current mechanisms as defined in IEEE802.1AS with specific extensions and/or restrictions.

Automotive Ethernet networks deviate from commercial Ethernet networks in terms of the following items:

- Role and functions of ECUs is known and defined a priori
- The network is static, i.e. components like ECUs, switches and characteristics like cable length, don’t change during ”operation” or even after switching off and switching on the vehicle. Components of course may be unavailable (due to failure situations or by purpose) but mostly only change when the vehicle is at a service facility.

Therefore, dynamic mechanisms like determining the Global Time Master (denoted as grandmaster in IEEE802.1AS) by the best master clock algorithm (BMCA) during operation are not required.
It is also possible to omit the cyclic measurement of link delays on Ethernet links due to the static nature of the automotive network and restrict mechanisms that belonging to dynamic network topology.

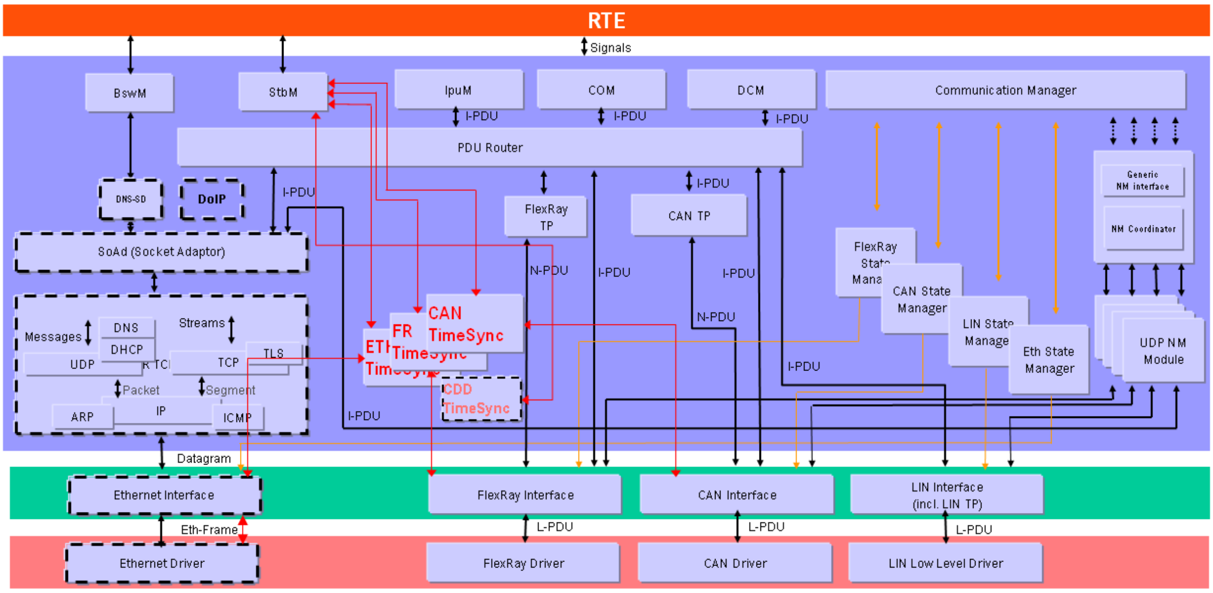


Figure 1: Clustering of Timesync modules

2 Acronyms, Abbreviations and Definitions

This section lists module local Abbreviations and Definitions. For a complete set of Synchronized Time Base related terms refer to the corresponding chapter in [5].

Abbreviation / Acronym:	Description
(G)TD	(Global) Time Domain
(G)TM	(Global)Time Master
<Bus>TSyn	A bus specific Time Synchronization module
AVB	Audio Video Bridging
BMCA	Best Master Clock Algorithm
CID	Company ID (IEEE)
CRC	Cyclic Redundancy Checksum
Debounce Time	Minimum gap between two Tx messages with the same PDU.
DEM	Diagnostic Event Manager
DET	Default Error Tracer
ETH	Ethernet
EthTSyn	Time Synchronization Provider module for Ethernet
Follow_Up	Time transport message (Follow-Up)
GM(C)	Grand Master (Clock)
OFS	Offset synchronization
Pdelay	Propagation / path delay as given in IEEE 802.1AS
Pdelay_Req	Propagation / path delay request message
Pdelay_Resp	Propagation / path delay response message
Pdelay_Resp_Follow_Up	Propagation / path delay Follow-Up message
PDU	Protocol Data Unit
PTP	Precision Time Protocol
StbM	Synchronized Time-Base Manager
Timesync	Time Synchronization
Sync	Time synchronization message (Sync)
TG	Time Gateway
TLV	Type, Length, Value field (acc. to IEEE 802.1AS)
TS	Time Slave
TSD	Time Sub-domain
VLAN	Virtual Local Area Network

3 Related documentation

3.1 Input documents

[1] AUTOSAR Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

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

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

[4] Requirements on Ethernet Support in AUTOSAR
AUTOSAR_SRS_Ethernet.pdf

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

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

[7] Specification of the Ethernet Interface
AUTOSAR_SWS_EthernetInterface.pdf

[8] Specification of Default Error Tracer
AUTOSAR_SWS_DefaultErrorTracer.pdf

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

[10] AUTOSAR Specification of CRC Routines
AUTOSAR_SWS_CRCLibrary.pdf

[11] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

3.2 Related standards and norms

[12] IEEE Standard 802.1AS™- 30 of March 2011
http://standards.ieee.org/getieee802/download/802.1AS-2011.pdf

[13] IEEE 802.1Q-2011 - IEEE Standard for Local and metropolitan area networks - Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software (SWS BSW General [5]) which is also valid for EthTSyn.

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

4 Constraints and assumptions

4.1 Limitations

1. No support of BMCA protocol, like specified in [12].
2. No support of `Announce` and `Signaling` messages, like specified in [12].
3. The reception of a `Pdelay_Req` is not taken as a pre-condition to start with the transmission of `Sync`.
4. The Rate Correction will be performed by the StbM, which does not require the `Pdelay` mechanism. For some applications, e.g. for Audio/Video, it might be necessary to use `Pdelay` based Rate Correction performed by EthTSyn itself, which is optional and not considered by this specification.
5. Because of (4), EthTSyn will not maintain the Ethernet HW clock.
6. While IEEE 802.1AS states, that IEEE 802.1AS message shall not have a VLAN tag nor a priority tag, EthTSyn would allow Time Synchronization on VLANs under the condition, that the switch HW supports forwarding of reserved multicast address using the range of 01:80:C2:00:00:00 .. 0F.

Time Master and Time Slave shall work with a Time Base reference clock accuracy as defined in“[12], ANNEX B.1.2 Time measurement granularity”.

4.2 Applicability to car domains

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

5 Dependencies to other modules

The Global Time Synchronization over Ethernet (EthTSyn) has interfaces towards the Synchronized Time-Base Manager (StbM), the Ethernet Interface (EthIf), the Basic Software Mode Manager (BswM) and the Default Error Tracer (DET).

- StbM – Get and set the current time value
- EthIf – Receiving and transmitting messages
- BswM – Coordination of network access
- DET – Reporting of development errors

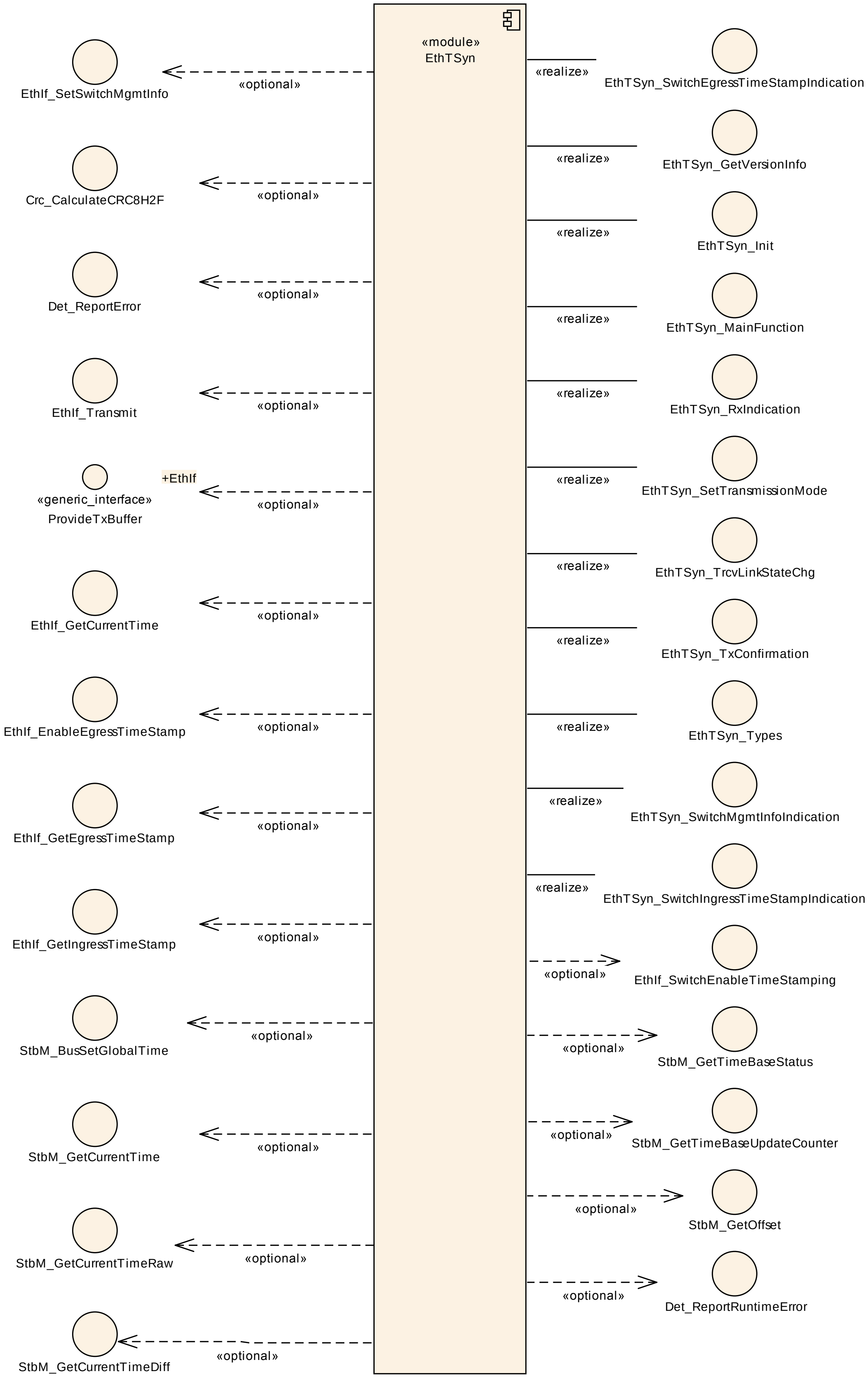

«module»
EthTSyn
EthIf_SetSwitchMgmtInfo
Crc_CalculateCRC8H2F
Det_ReportError
EthIf_Transmit
+EthIf
«generic_interface»
ProvideTxBuffer
EthIf_GetCurrentTime
EthIf_EnableEgressTimeStamp
EthIf_GetEgressTimeStamp
EthIf_GetIngressTimeStamp
StbM_BusSetGlobalTime
StbM_GetCurrentTime
StbM_GetCurrentTimeRaw
StbM_GetCurrentTimeDiff
«optional»
«realize»
EthTSyn_SwitchEgressTimeStampIndication
EthTSyn_GetVersionInfo
EthTSyn_Init
EthTSyn_MainFunction
EthTSyn_RxIndication
EthTSyn_SetTransmissionMode
EthTSyn_TrcvLinkStateChg
EthTSyn_TxConfirmation
EthTSyn_Types
EthTSyn_SwitchMgmtInfoIndication
EthTSyn_SwitchIngressTimeStampIndication
EthIf_SwitchEnableTimeStamping
StbM_GetTimeBaseStatus
StbM_GetTimeBaseUpdateCounter
StbM_GetOffset
Det_ReportRuntimeError

Figure 2: Module dependencies of the EthTSyn module

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 [5].

5.1.2 Header file structure

Besides the files defined in section 5.1.7 “Header file structure” of the SWS BSW General [5], the Global Time Synchronization over Ethernet needs to include the files defined below.

[SWS_EthTSyn_00001]⌈
The implementation header files shall include *ComStack_Types.h*.
⌋(SRS_BSW_00301, SRS_BSW_00456)

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

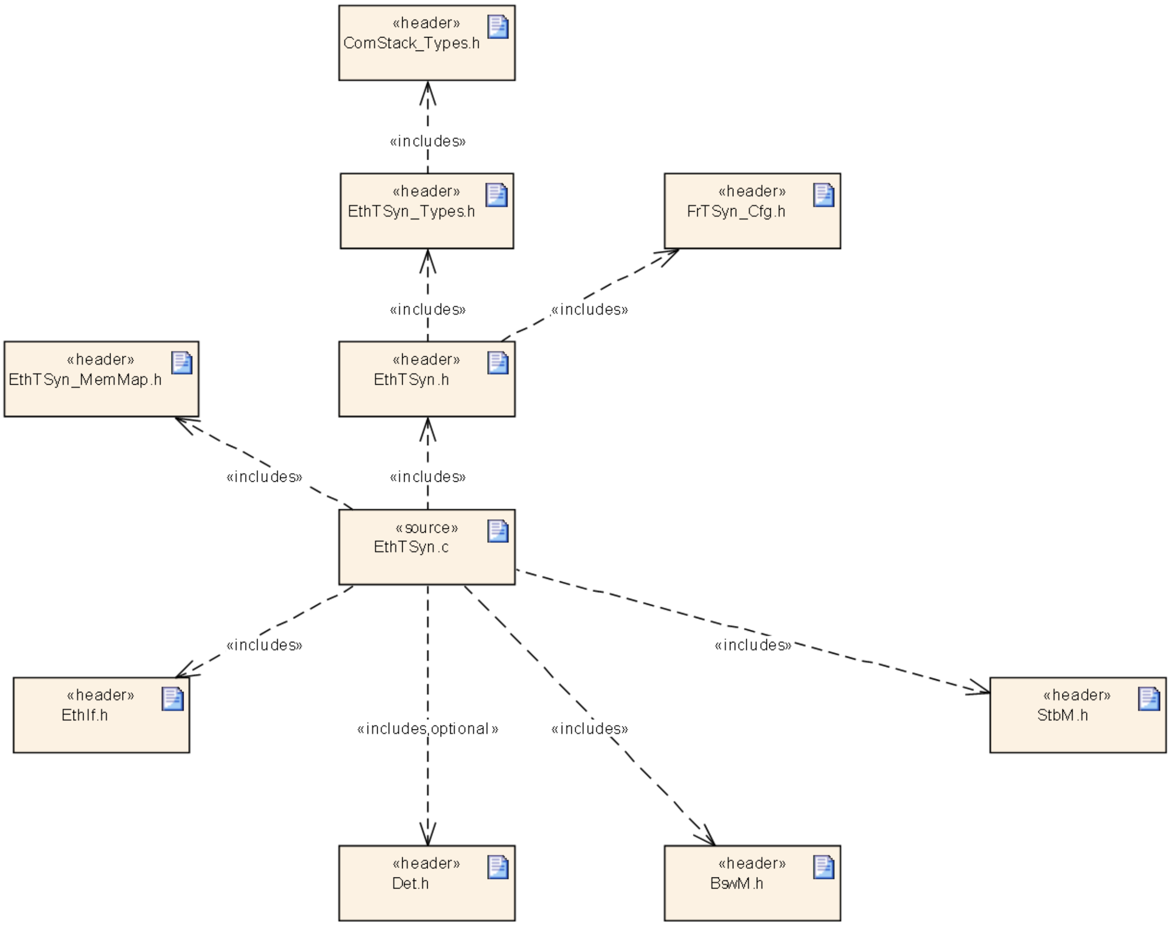


Figure 3: File structure of EthTSyn

6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_EthTSyn_00006, SWS_EthTSyn_00008
SRS_BSW_00301	All AUTOSAR Basic Software Modules shall only import the necessary information	SWS_EthTSyn_00001
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_EthTSyn_00029, SWS_EthTSyn_00030, SWS_EthTSyn_00041, SWS_EthTSyn_00172, SWS_EthTSyn_00174, SWS_EthTSyn_00175, SWS_EthTSyn_00177, SWS_EthTSyn_00178
SRS_BSW_00337	Classification of development errors	SWS_EthTSyn_00007, SWS_EthTSyn_00030, SWS_EthTSyn_00041, SWS_EthTSyn_00152, SWS_EthTSyn_00172, SWS_EthTSyn_00174, SWS_EthTSyn_00175, SWS_EthTSyn_00177, SWS_EthTSyn_00178
SRS_BSW_00385	List possible error notifications	SWS_EthTSyn_00030, SWS_EthTSyn_00144
SRS_BSW_00456	- A Header file shall be defined in order to harmonize BSW Modules	SWS_EthTSyn_00001
SRS_StbM_20047	The Ethernet Timesync module shall trigger Time Base Synchronization transmission	SWS_EthTSyn_00016, SWS_EthTSyn_00050, SWS_EthTSyn_00130, SWS_EthTSyn_00131, SWS_EthTSyn_00132, SWS_EthTSyn_00133, SWS_EthTSyn_00134, SWS_EthTSyn_00135, SWS_EthTSyn_00136, SWS_EthTSyn_00137, SWS_EthTSyn_00139, SWS_EthTSyn_00165
SRS_StbM_20048	The Ethernet Timesync module shall support IEEE 802.1AS as well as AUTOSAR extensions	SWS_EthTSyn_00002, SWS_EthTSyn_00003, SWS_EthTSyn_00004, SWS_EthTSyn_00005, SWS_EthTSyn_00010, SWS_EthTSyn_00011, SWS_EthTSyn_00012, SWS_EthTSyn_00013, SWS_EthTSyn_00014, SWS_EthTSyn_00016, SWS_EthTSyn_00017, SWS_EthTSyn_00018, SWS_EthTSyn_00019, SWS_EthTSyn_00020, SWS_EthTSyn_00021, SWS_EthTSyn_00022, SWS_EthTSyn_00023, SWS_EthTSyn_00024, SWS_EthTSyn_00025, SWS_EthTSyn_00028, SWS_EthTSyn_00031, SWS_EthTSyn_00032, SWS_EthTSyn_00033, SWS_EthTSyn_00034, SWS_EthTSyn_00035, SWS_EthTSyn_00036, SWS_EthTSyn_00039, SWS_EthTSyn_00040, SWS_EthTSyn_00042, SWS_EthTSyn_00043, SWS_EthTSyn_00044, SWS_EthTSyn_00045, SWS_EthTSyn_00047, SWS_EthTSyn_00049, SWS_EthTSyn_00050, SWS_EthTSyn_00052, SWS_EthTSyn_00053, SWS_EthTSyn_00054, SWS_EthTSyn_00055, SWS_EthTSyn_00056, SWS_EthTSyn_00057, SWS_EthTSyn_00058,

		SWS_EthTSyn_00059, SWS_EthTSyn_00060, SWS_EthTSyn_00061, SWS_EthTSyn_00062, SWS_EthTSyn_00063, SWS_EthTSyn_00064, SWS_EthTSyn_00065, SWS_EthTSyn_00066, SWS_EthTSyn_00067, SWS_EthTSyn_00068, SWS_EthTSyn_00069, SWS_EthTSyn_00070, SWS_EthTSyn_00071, SWS_EthTSyn_00072, SWS_EthTSyn_00075, SWS_EthTSyn_00077, SWS_EthTSyn_00079, SWS_EthTSyn_00086, SWS_EthTSyn_00122, SWS_EthTSyn_00123, SWS_EthTSyn_00124, SWS_EthTSyn_00126, SWS_EthTSyn_00127, SWS_EthTSyn_00128, SWS_EthTSyn_00141, SWS_EthTSyn_00142, SWS_EthTSyn_00148, SWS_EthTSyn_00149, SWS_EthTSyn_00154, SWS_EthTSyn_00159, SWS_EthTSyn_00160, SWS_EthTSyn_00161, SWS_EthTSyn_00162, SWS_EthTSyn_00163, SWS_EthTSyn_00164, SWS_EthTSyn_00166, SWS_EthTSyn_00167, SWS_EthTSyn_00168, SWS_EthTSyn_00169, SWS_EthTSyn_00170, SWS_EthTSyn_00171, SWS_EthTSyn_00179, SWS_EthTSyn_00180, SWS_EthTSyn_00181, SWS_EthTSyn_91000, SWS_EthTSyn_91001, SWS_EthTSyn_91002
SRS_StbM_20051	The Ethernet Timesync module shall detect and handle errors in synchronization protocol / communication	SWS_EthTSyn_00004, SWS_EthTSyn_00019, SWS_EthTSyn_00020, SWS_EthTSyn_00021, SWS_EthTSyn_00022, SWS_EthTSyn_00025, SWS_EthTSyn_00029, SWS_EthTSyn_00129, SWS_EthTSyn_00145, SWS_EthTSyn_00146, SWS_EthTSyn_00164
SRS_StbM_20052	The Ethernet Timesync configuration shall allow the EthTSyn to support different roles for a Time Base	SWS_EthTSyn_00051, SWS_EthTSyn_00064, SWS_EthTSyn_00094, SWS_EthTSyn_00156
SRS_StbM_20058	The Ethernet Timesync module shall provide the precision of Synchronized Time Bases	SWS_EthTSyn_00150
SRS_StbM_20059	The Ethernet Timesync module shall access all communication ports belonging to Time Synchronization	SWS_EthTSyn_00031, SWS_EthTSyn_00047, SWS_EthTSyn_00053, SWS_EthTSyn_00054, SWS_EthTSyn_00055, SWS_EthTSyn_00056, SWS_EthTSyn_00057, SWS_EthTSyn_00058, SWS_EthTSyn_00059, SWS_EthTSyn_00060, SWS_EthTSyn_00166, SWS_EthTSyn_00167, SWS_EthTSyn_00168, SWS_EthTSyn_00169, SWS_EthTSyn_00170, SWS_EthTSyn_00171, SWS_EthTSyn_91000, SWS_EthTSyn_91001, SWS_EthTSyn_91002
SRS_StbM_20061	The Ethernet Timesync module shall support means to protect the Time Synchronization protocol	SWS_EthTSyn_00062, SWS_EthTSyn_00063, SWS_EthTSyn_00065, SWS_EthTSyn_00066, SWS_EthTSyn_00067, SWS_EthTSyn_00068, SWS_EthTSyn_00069, SWS_EthTSyn_00070, SWS_EthTSyn_00071, SWS_EthTSyn_00072, SWS_EthTSyn_00074, SWS_EthTSyn_00075, SWS_EthTSyn_00076, SWS_EthTSyn_00077, SWS_EthTSyn_00078, SWS_EthTSyn_00079,

		SWS_EthTSyn_00080, SWS_EthTSyn_00081, SWS_EthTSyn_00082, SWS_EthTSyn_00084, SWS_EthTSyn_00085, SWS_EthTSyn_00086, SWS_EthTSyn_00087, SWS_EthTSyn_00088, SWS_EthTSyn_00089, SWS_EthTSyn_00091, SWS_EthTSyn_00092, SWS_EthTSyn_00093, SWS_EthTSyn_00096, SWS_EthTSyn_00097, SWS_EthTSyn_00098, SWS_EthTSyn_00099, SWS_EthTSyn_00100, SWS_EthTSyn_00101, SWS_EthTSyn_00102, SWS_EthTSyn_00103, SWS_EthTSyn_00104, SWS_EthTSyn_00105, SWS_EthTSyn_00106, SWS_EthTSyn_00107, SWS_EthTSyn_00108, SWS_EthTSyn_00109, SWS_EthTSyn_00111, SWS_EthTSyn_00112, SWS_EthTSyn_00113, SWS_EthTSyn_00114, SWS_EthTSyn_00115, SWS_EthTSyn_00116, SWS_EthTSyn_00117, SWS_EthTSyn_00118, SWS_EthTSyn_00119, SWS_EthTSyn_00120, SWS_EthTSyn_00153, SWS_EthTSyn_00157, SWS_EthTSyn_00181, SWS_EthTSyn_00182, SWS_EthTSyn_00183, SWS_EthTSyn_00184, SWS_EthTSyn_00185
SRS_StbM_20062	The Ethernet Timesync module shall support user specific data within the time measurement and synchronization protocol	SWS_EthTSyn_00062, SWS_EthTSyn_00063, SWS_EthTSyn_00065, SWS_EthTSyn_00066, SWS_EthTSyn_00067, SWS_EthTSyn_00068, SWS_EthTSyn_00069, SWS_EthTSyn_00070, SWS_EthTSyn_00071, SWS_EthTSyn_00072, SWS_EthTSyn_00074, SWS_EthTSyn_00075, SWS_EthTSyn_00076, SWS_EthTSyn_00077, SWS_EthTSyn_00078, SWS_EthTSyn_00079, SWS_EthTSyn_00080, SWS_EthTSyn_00081, SWS_EthTSyn_00082, SWS_EthTSyn_00084, SWS_EthTSyn_00085, SWS_EthTSyn_00086, SWS_EthTSyn_00087, SWS_EthTSyn_00088, SWS_EthTSyn_00089, SWS_EthTSyn_00092, SWS_EthTSyn_00103, SWS_EthTSyn_00104, SWS_EthTSyn_00106, SWS_EthTSyn_00118, SWS_EthTSyn_00119, SWS_EthTSyn_00120, SWS_EthTSyn_00153, SWS_EthTSyn_00181
SRS_StbM_20063	The Ethernet Timesync module shall use the Time Synchronization protocol for Synchronized Time Bases to transmit and receive Offset Time Bases	SWS_EthTSyn_00092, SWS_EthTSyn_00095, SWS_EthTSyn_00103, SWS_EthTSyn_00104, SWS_EthTSyn_00106, SWS_EthTSyn_00110, SWS_EthTSyn_00117, SWS_EthTSyn_00118, SWS_EthTSyn_00119, SWS_EthTSyn_00120
SRS_StbM_20066	The Ethernet Timesync module shall support a static (pre)configuration of IEEE 802.1AS Pdelay	SWS_EthTSyn_00003, SWS_EthTSyn_00011, SWS_EthTSyn_00012, SWS_EthTSyn_00140, SWS_EthTSyn_00141, SWS_EthTSyn_00142, SWS_EthTSyn_00143, SWS_EthTSyn_00149

7 Functional specification

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

7.1 Overview

The module EthTSyn is responsible to ensure the collection and distribution of synchronized time information across the Ethernet network. It interacts with the StbM and provides all Ethernet specific functions to the StbM.

[SWS_EthTSyn_00002]⌈
IEEE802.1AS [12] specifies default configuration values like the MAC destination address or Ethernet frame type. The EthTSyn shall use these default configuration values if not otherwise specified within this document.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00005]⌈
All messages belonging to the IEEE Rapid Spanning Tree Protocol (`PortAnnounceReceive`, `PortAnnounceInformation`, `PortRoleSelection`, `PortAnnounceTransmit`) shall be ignored on the receiver side and shall be prohibited on the sender side. Therefore, Time Master and Time Slave shall start their protocol state machines without `Announce` message recognition.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00148]⌈
If the parameter `EthTSynFramePrio` (**ECUC_EthTSyn_00034 :**) exists, the `EthTSynGlobalTimeEthIfRef` (**ECUC_EthTSyn_00065 :**) shall refer to a Virtual Ethernet Controller representing a VLAN.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00162]⌈
Time Slave and Time Master shall use the `EthTSynFramePrio` (**ECUC_EthTSyn_00034 :**) value as priority parameter when calling `EthIf_ProvideTxBuffer()`.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00163]⌈
If `EthTSynFramePrio` (**ECUC_EthTSyn_00034 :**) does not exist, a frame format without priority and VLAN tags shall be used.
⌋(SRS_StbM_20048)

7.2 Initialization

The Global Time Synchronization over Ethernet is initialized via `EthTSyn_Init()`. Except for `EthTSyn_GetVersionInfo()` and `EthTSyn_Init()`, the API functions of the EthTSyn module may only be called when the module has been properly initialized.

[SWS_EthTSyn_00006]⌈
A call to `EthTSyn_Init()` initializes all internal variables and sets the EthTSyn module to the initialized state.
⌋(SRS_BSW_00101)

[SWS_EthTSyn_00007]⌈
When DET reporting is enabled (refer `EthTSynDevErrorDetect` (**ECUC_EthTSyn_00002 :**)), the EthTSyn module shall call `Det_ReportError()` with the error code `ETHTSYN_E_UNINIT` when any API other than `EthTSyn_GetVersionInfo()` or `EthTSyn_Init()` is called in uninitialized state.⌋
(SRS_BSW_00337)

[SWS_EthTSyn_00008]⌈
When `EthTSyn_Init()` is called in initialized state, the EthTSyn module shall re-initialize its internal variables.
⌋(SRS_BSW_00101)

[SWS_EthTSyn_00010]⌈
When `EthTSyn_Init()` is called in initialized state, the EthTSyn module shall set each port-specific `Pdelay` value to 0.
⌋(SRS_StbM_20048)

7.3 Debounce Time

[SWS_EthTSyn_00130]⌈
If `EthTSynGlobalTimeDebounceTime` (**ECUC_EthTSyn_00048 :**) is set to 0, EthTSyn shall ignore any debouncing.
⌋(SRS_StbM_20047)

[SWS_EthTSyn_00131]⌈
If `EthTSynGlobalTimeDebounceTime` (**ECUC_EthTSyn_00048 :**) is greater than 0, EthTSyn shall always consider debouncing for all Timesync PDUs (`Sync`, `Follow_Up`, `Pdelay_Req`, `Pdelay_Resp` and `Pdelay_Resp_Follow_Up`) as described below.
⌋(SRS_StbM_20047)

[SWS_EthTSyn_00132]⌈
`EthTSynGlobalTimeDebounceTime` (**ECUC_EthTSyn_00048 :**) represents the reload value of a `debounceCounter` that will be reloaded at that point in time, where a Timesync PDU has been sent and that will be decremented on each `EthTSyn_MainFunction()` call if no Timesync PDU is transmitted.
⌋(SRS_StbM_20047)

[SWS_EthTSyn_00133]⌈

A new Timesync PDU shall only be sent, if the corresponding debounceCounter has reached 0.
⌋(SRS_StbM_20047)

7.4 Pdelay Protocol for Latency Calculation

[SWS_EthTSyn_00003]⌈

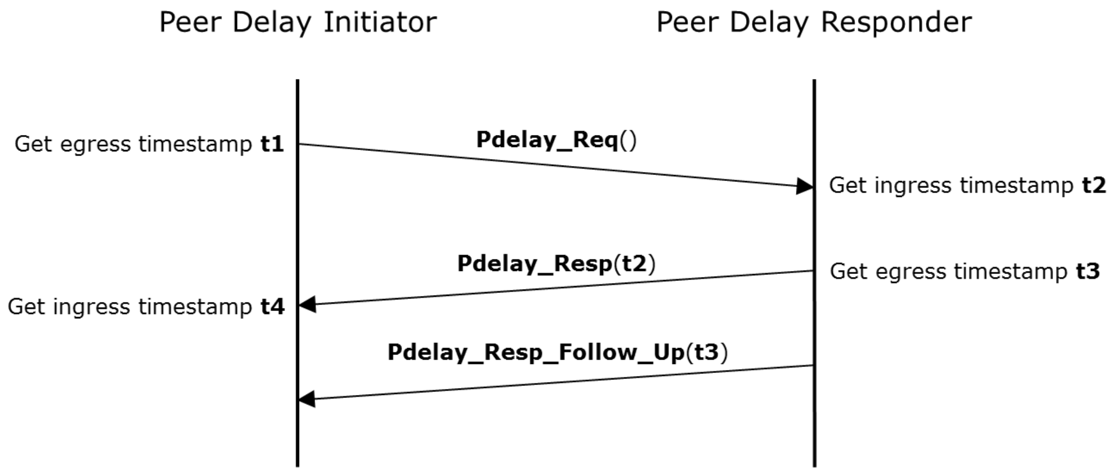

Peer Delay Initiator
Peer Delay Responder
Get egress timestamp **t1**
Pdelay_Req()
Get ingress timestamp **t2**
Pdelay_Resp(**t2**)
Get egress timestamp **t3**
Get ingress timestamp **t4**
Pdelay_Resp_Follow_Up(**t3**)

Figure 4: Propagation Delay Measurement (Pdelay)

The EthTSyn module shall use for latency calculation

- either static Pdelay values (EthTSynGlobalTimePropagationDelay (**ECUC_EthTSyn_00070 :**))
- or runtime-based values calculated by Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up according to Figure 4,

depending on configuration of EthTSynGlobalTimeTxPdelayReqPeriod (**ECUC_EthTSyn_00071 :**).
⌋(SRS_StbM_20048, SRS_StbM_20066)

[SWS_EthTSyn_00154]⌈
If EthTSynGlobalTimeTxPdelayReqPeriod is not equal to 0 and if the Pdelay latency calculation result exceeds EthTSynPdelayLatencyThreshold, the measured value shall be discarded and the previous value shall be kept.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00004]⌈
A Pdelay_Resp timeout or incomplete Pdelay protocol shall stop the latency calculation algorithm. In such cases, the device shall use the latest successful calculated latency value.
⌋(SRS_StbM_20048, SRS_StbM_20051)

Note: A timeout is detected, when sending the next subsequent Pdelay_Req before receiving the Pdelay_Resp resp. Pdelay_Resp_Follow_Up belonging to the Pdelay_Req before.

[SWS_EthTSyn_00164]⌈
Time Master and Time Slave shall observe the Pdelay timeout as given by EthTSynPdelayRespAndRespFollowUpTimeout (**ECUC_EthTSyn_00074 :**), if a Pdelay_Req has been transmitted (waiting for Pdelay_Resp) or if a Pdelay_Resp has been received (waiting for Pdelay_Resp_Follow_Up).
If the reception timeout occurs, any received Pdelay_Resp resp Pdelay_Resp_Follow_Up shall be ignored, until a new Pdelay_Req has been sent. A value of 0 deactivates this timeout observation.
⌋(SRS_StbM_20048, SRS_StbM_20051)

[SWS_EthTSyn_00140]⌈
If EthTSynGlobalTimeTxPdelayReqPeriod (**ECUC_EthTSyn_00071 :**) equals 0, Time Master and Time Slave shall not measure the propagation delay.
The Time Slave shall use a static value EthTSynGlobalTimePropagationDelay (**ECUC_EthTSyn_00070 :**) as propagation delay instead.
⌋(SRS_StbM_20066)

Note: Since EthTSynGlobalTimeTxPdelayReqPeriod is ECU specific, neither a Time Master nor all Time Slaves have to measure the propagation delay. Global Time Synchronization in AUTOSAR does yet not define dynamic reconfiguration or backup strategies that will reassign the role as Time Master, therefore propagation delay measurements make currently no sense for a Time Master (although a Time Master shall be able to handle Pdelay_Req initiated by a Time Slave).

[SWS_EthTSyn_00141]⌈
If EthTSynGlobalTimeTxPdelayReqPeriod (**ECUC_EthTSyn_00071 :**) is greater than 0, Time Master and Time Slave shall cyclically measure the propagation delay using Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up as defined in [12] chapter 11.1.2 “Propagation delay measurement”.
⌋(SRS_StbM_20048, SRS_StbM_20066)

[SWS_EthTSyn_00149]⌈
If EthTSynGlobalTimeTxPdelayReqPeriod (**ECUC_EthTSyn_00071 :**) is greater than 0, Time Master and Time Slave shall cyclically measure the propagation delay only on that Time Domain with the lowest Time Domain ID and shall use this value to adjust all Synchronized Time Bases.
⌋(SRS_StbM_20048, SRS_StbM_20066)

Note: There is no need to measure the propagation delay for all Time Domains, because the same value is expected. This requirement ensures also the usage of Time Domain 0 for Pdelay, to be compatible to [12].

[SWS_EthTSyn_00142]⌈
If EthTSynGlobalTimeTxPdelayReqPeriod (**ECUC_EthTSyn_00071 :**) is greater than 0, EthTSynGlobalTimePropagationDelay (**ECUC_EthTSyn_00070 :**) shall be used as default value for the propagation delay, until first valid propagation delay has been measured.
⌋(SRS_StbM_20048, SRS_StbM_20066)

[SWS_EthTSyn_00011]⌈
If `EthTSynGlobalTimeTxPdelayReqPeriod` (**ECUC_EthTSyn_00071 :**) is greater than 0, Time Master and Time Slave shall periodically transmit `Pdelay_Req` for latency calculation with the cycle `EthTSynGlobalTimeTxPdelayReqPeriod` (**ECUC_EthTSyn_00071 :**) as defined in [12] chapter 11.1.2 "Propagation delay measurement".

For that, the following sequence shall be applied:

1. Get a free transmission buffer via `EthIf_ProvideTxBuffer()`
2. Activate the time stamping via `EthIf_EnableEgressTimeStamp()` if `EthTSynHardwareTimestampSupport` (**ECUC_EthTSyn_00018 :**) is set to `TRUE`
3. Trigger transmit request via `EthIf_Transmit()`

⌋(SRS_StbM_20048, SRS_StbM_20066)

Note: `EthTSynGlobalTimePdelayRespEnable` allows disabling of `Pdelay_Resp` and `Pdelay_Resp_Follow_Up`, if no `Pdelay_Req` is expected to be received, i.e. for the Time Master, if all Time Slaves have set `EthTSynGlobalTimeTxPdelayReqPeriod` to 0 or for any Time Slave if the Time Master has set `EthTSynGlobalTimeTxPdelayReqPeriod` to 0.

[SWS_EthTSyn_00012]⌈
If `EthTSynGlobalTimePdelayRespEnable` (**ECUC_EthTSyn_00069 :**) is set to `TRUE`, Time Master and Time Slave shall react to `Pdelay_Req` by transmitting `Pdelay_Resp` for latency calculation as defined in [12] chapter 11.1.2 "Propagation delay measurement".

For that, the following sequence shall be applied:

1. Get a free transmission buffer via `EthIf_ProvideTxBuffer()`
2. Activate the time stamping via `EthIf_EnableEgressTimeStamp()` if `EthTSynHardwareTimestampSupport` (**ECUC_EthTSyn_00018 :**) is set to `TRUE`
3. Trigger transmit request via `EthIf_Transmit()`

⌋(SRS_StbM_20048, SRS_StbM_20066)

[SWS_EthTSyn_00013]⌈
On invocation of `EthTSyn_TxConfirmation()` the egress time stamp shall be retrieved for **t1** (`Pdelay_Req`) from the EthIf via `EthIf_GetEgressTimeStamp()` according to Figure 18, if `EthTSynHardwareTimestampSupport` **(ECUC_EthTSyn_00018 :)** is set to `TRUE`.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00123]⌈
On invocation of `EthTSyn_TxConfirmation()` the egress time stamp shall be retrieved for **t1** (`Pdelay_Req`) from the StbM via `StbM_GetCurrentTimeRaw()`

according to Figure 18, if EthTSynHardwareTimestampSupport **(ECUC_EthTSyn_00018 :)** is set to FALSE.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00159]⌈
On invocation of EthTSyn_TxConfirmation() the responseOriginTimestamp **t3** valid for Pdelay_Resp_Follow_Up shall be retrieved from the from the EthIf via EthIf_GetEgressTimeStamp() according to Figure 18, if EthTSynHardwareTimestampSupport **(ECUC_EthTSyn_00018 :)** is set to TRUE.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00122]⌈
On invocation of EthTSyn_TxConfirmation() the responseOriginTimestamp **t3** valid for Pdelay_Resp_Follow_Up shall be retrieved from the StbM via StbM_GetCurrentTimeRaw() on egress of Pdelay_Resp according to Figure 18 if EthTSynHardwareTimestampSupport **(ECUC_EthTSyn_00018 :)** is set to FALSE.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00014]⌈
If EthTSynGlobalTimePdelayRespEnable (**ECUC_EthTSyn_00069 :**) is set to TRUE, Time Master and Time Slave shall transmit Pdelay_Resp_Follow_Up with the transmission timestamp of that messages as defined in **[SWS_EthTSyn_00013]** as well as defined in [12] chapter 11.1.2 “Propagation delay measurement” considering debounceCounter which represents a time offset between Pdelay_Resp and Pdelay_Resp_Follow_Up.

For that, the following sequence shall be applied:

1. Get a free transmission buffer via EthIf_ProvideTxBuffer()
2. Trigger transmit request with the transmission timestamp of **[SWS_EthTSyn_00013]** via EthIf_Transmit()

⌋(SRS_StbM_20048)

[SWS_EthTSyn_00143]⌈
If EthTSynGlobalTimePdelayRespEnable (**ECUC_EthTSyn_00069 :**) is set to FALSE, Pdelay_Resp and Pdelay_Resp_Follow_Up shall be omitted.
⌋(SRS_StbM_20066)

[SWS_EthTSyn_00160]⌈
On invocation of EthTSyn_RxIndication() the requestReceiptTimestamp **t2** valid for Pdelay_Resp shall be retrieved from the EthIf via EthIf_GetIngressTimeStamp() on ingress of Pdelay_Req according to Figure 19, if EthTSynHardwareTimestampSupport **(ECUC_EthTSyn_00018 :)** is set to TRUE.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00124]⌈

On invocation of EthTSyn_RxIndication() the requestReceiptTimestamp **t2** valid for Pdelay_Resp shall be retrieved from the StbM via StbM_GetCurrentTimeRaw() on ingress of Pdelay_Req according to Figure 19, if EthTSynHardwareTimestampSupport **(ECUC_EthTSyn_00018 :)** is set to FALSE.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00049]⌈
On invocation of EthTSyn_RxIndication() the ingress time stamp shall be retrieved for **t4** (Pdelay_Resp) from the EthIf via EthIf_GetIngressTimeStamp() according to Figure 19, if EthTSynHardwareTimestampSupport (**ECUC_EthTSyn_00018 :**) is set to TRUE.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00161]⌈
On invocation of EthTSyn_RxIndication() the ingress time stamp shall be retrieved for **t4** (Pdelay_Resp) from the StbM via StbM_GetCurrentTimeRaw() according to Figure 19, if EthTSynHardwareTimestampSupport (**ECUC_EthTSyn_00018 :**) is set to FALSE.
⌋(SRS_StbM_20048)

7.5 Message Format

For harmonization purpose among all AUTOSAR <Bus>TSyn modules, some message extensions to the IEEE specification [12] are required. This is accomplished by a new AUTOSAR specific *TLV*, which is using a new IEEE CID (0x1A75FB) belonging to AUTOSAR only. An IEEE 802.1AS *TLV* is only available for the messageType Announce (not considered by this specification) and Follow_Up (extended by this specification). The organizationId of the new *TLV* identifies the AUTOSAR *TLV*, which is succeeding the IEEE 802.1AS *TLV*.

The AUTOSAR *TLV* contains *Sub-TLV's* which always consist of a Type, a Length and a data area.

The usage of the *CRC* is optional. To ensure a great variability between several time observing units, the configuration decides of how to handle the *CRC* of a secured *Sub-TLV*. If the receiver does not support the *CRC* calculation, it might be possible, that a receiver just uses the given values, without evaluating the *CRC* itself.

If the *CRC* option is used, one side effect must be considered. Due to the fact, that Pdelay messages do not contain any *TLV*, a *CRC* protection of the related timestamps is not possible. If applications using a *CRC* for Follow_Up together with a non-static Pdelay, unprotected Pdelay time values have to be mixed with protected Follow_Up time values, while calculating the value of the Synchronized Time Base.

[SWS_EthTSyn_00028]⌈
The message format, etc. shall be derived from [12] chapter 10. “Media-independent layer specification” and chapter 11. “Media-dependent layer specification for full-duplex, point-to-point links”, if not otherwise specified. The default values shall be used, if not specified different in this document.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00181]⌈
The byte order for multibyte values is ”Big Endian“, which is equal to the byte order defined by [12].
⌋(SRS_StbM_20048, SRS_StbM_20061, SRS_StbM_20062)

7.5.1 `Sync` and `Follow_Up` acc. to IEEE 802.1AS

[SWS_EthTSyn_00061]⌈
If `EthTSynMessageCompliance` (**ECUC_EthTSyn_00029 :**) is set to `TRUE`, `Sync and Follow_Up` format shall be supported acc. to [12].
⌋(SRS_StbM_20048)

Note: This implies that EthTSyn supports only one Time Domain (0).

The table below [Figure 5] gives an overview, how an IEEE conformant `Sync` looks like.

`Sync` Message Header [IEEE 802.1AS]				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
transportSpecific	**messageType**	1	0	`0x10`
reserved	**versionPTP**	1	1	`2`
messageLength		2	2	`44`
domainNumber		1	4	`(UInteger8)domainNumber = 0`
reserved		1	5	`0`
flags		2	6	`2`

correctionField		8	8	`0..281474976710655ns`
				`[1ns = 2^16 = 0x0000 0000 0001 0000]`

reserved		4	16	`0`

sourcePortIdentity		10	20	`(PortIdentity)portIdentity`
				from origin Time Aware End Station

sequenceId		2	30	`(UInteger16)SyncSequenceId =`
				`(UInteger16)(prevSyncSequenceId+1)`

control		1	32	`0`

logMessageInterval		1	33	(Integer8)currentLogSyncInterval
Sync Message Fields [IEEE 802.1AS]				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
PTP Message Header		34	0	[refer Sync Message Header]
reserved		10	34	0

Figure 5: Sync [IEEE 802.1AS]

The table below [Figure 6] gives an overview, how an IEEE conformant Follow_Up looks like.

Follow_Up Message Header [IEEE 802.1AS]				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
transportSpecific	**messageType**	1	0	0x18
reserved	**versionPTP**	1	1	0x02
messageLength		2	2	76
domainNumber		1	4	(UInteger8)domainNumber = 0
reserved		1	5	0
flags		2	6	2
correctionField		8	8	0..281474976710655ns [1ns = 2^16 = 0x0000 0000 0001 0000]
reserved		4	16	0
sourcePortIdentity		10	20	(PortIdentity)portIdentity from origin Time Aware End Station
sequenceId		2	30	(UInteger16)SyncSequenceId
control		1	32	2
logMessageInterval		1	33	(Integer8)currentLogSyncInterval
Follow_Up Message Fields [IEEE 802.1AS]				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
PTP Message Header		34	0	[refer Follow_Up Message Header]
preciseOriginTimestamp		10	34	(Timestamp)preciseOriginTimestamp

Follow_Up information TLV	32	44	[refer Follow_Up information TLV]

Follow_Up information TLV [IEEE 802.1AS]

High Nibble	Low Nibble	Octets	Offset	Value
tlvType		2	0	3
lengthField		2	2	28
organizationId		3	4	0x0080C2
organizationSubType		3	7	1
cumulativeScaledRateOffset		4	10	(Integer32)((RateRatio-1)*2^41)
gmTimeBaseIndicator		2	14	0
lastGmPhaseChange		12	16	0
scaledLastGmFreqChange		4	28	0

Figure 6: Follow_Up [IEEE 802.1AS]

7.5.2 Sync and Follow_Up acc. to AUTOSAR

[SWS_EthTSyn_00062]⌈
If EthTSynMessageCompliance (**ECUC_EthTSyn_00029 :**) is set to FALSE, the Sync and Follow_Up format shall be supported acc. to:
“Figure 7: Sync [AUTOSAR]” and
“Figure 8: Follow_Up [AUTOSAR]”
depending on configuration.
⌋(SRS_StbM_20048, SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00063]⌈
If EthTSynMessageCompliance (**ECUC_EthTSyn_00029 :**) is set to FALSE, the Follow_Up shall contain an AUTOSAR *TLV* acc. to:
“Figure 8: Follow_Up [AUTOSAR]”,
depending on configuration.
⌋(SRS_StbM_20048, SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00064]⌈

Sync Message Header [AUTOSAR]

High Nibble	Low Nibble	Octets	Offset	Value
transportSpecific	**messageType**	1	0	0x10

reserved	versionPTP	1	1	2
messageLength		2	2	44
domainNumber		1	4	(UInteger8)domainNumber = 0..15
reserved		1	5	0
flags		2	6	2
correctionField		8	8	0..281474976710655ns [1ns = 2^16 = 0x0000 0000 0001 0000]
reserved		4	16	0
sourcePortIdentity		10	20	(PortIdentity)portIdentity from origin Time Aware End Station
sequenceId		2	30	(UInteger16)SyncSequenceId = (UInteger16)(prevSyncSequenceId+1)
control		1	32	0
logMessageInterval		1	33	(Integer8)currentLogSyncInterval

Sync Message Fields [AUTOSAR]

High Nibble	Low Nibble	Octets	Offset	Value
PTP Message Header		34	0	[refer Sync Message Header]
reserved		10	34	0

Figure 7: Sync [AUTOSAR]

⌋(SRS_StbM_20048, SRS_StbM_20052)

[SWS_EthTSyn_00065]⌈

Follow_Up Message Header [AUTOSAR]

High Nibble	Low Nibble	Octets	Offset	Value
transportSpecific	messageType	1	0	0x18
reserved	versionPTP	1	1	0x02
messageLength		2	2	76+10+ΣSub-TLV‘s
domainNumber		1	4	(UInteger8)domainNumber=0..15
reserved		1	5	0
flags		2	6	2
correctionField		8	8	0..281474976710655ns [1ns = 2^16 = 0x0000 0000 0001 0000]

reserved		4	16	0
sourcePortIdentity		10	20	(PortIdentity)portIdentity from origin Time Aware End Station
sequenceId		2	30	(UInteger16)SyncSequenceId
control		1	32	2
logMessageInterval		1	33	(Integer8)currentLogSyncInterval
Follow_Up Message Fields [AUTOSAR]				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
PTP Message Header		34	0	[refer Follow_Up Message Header]
preciseOriginTimestamp		10	34	(Timestamp)preciseOriginTimestamp
Follow_Up information TLV		32+10+ΣSub-TLV‘s	44	[refer Follow_Up information TLV]
Follow_Up information TLV [IEEE 802.1AS]				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
tlvType		2	0	3
lengthField		2	2	28
organizationId		3	4	0x0080C2 [IEEE 802.1AS]
organizationSubType		3	7	1
cumulativeScaledRateOffset		4	10	(Integer32)((RateRatio-1)*2^41)
gmTimeBaseIndicator		2	14	0
lastGmPhaseChange		12	16	0
scaledLastGmFreqChange		4	28	0
Follow_Up information TLV [AUTOSAR]				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
AUTOSAR *TLV* Header				
tlvType		2	0	3
lengthField		2	2	6+ΣSub-TLV‘s
organizationId		3	4	0x1A75FB [AUTOSAR]

organizationSubType	3	7	`0x605676` [BCD coded GlobalTimeEthTSyn]
AUTOSAR TLV *Sub-TLV*: Time Secured			
Type	1	0	`0x28` [Time secured]
Length	1	1	`3`
CRC_Time_Flags	1	2	BitMask `0x01 [messageLength]` BitMask `0x02 [domainNumber]` BitMask `0x04 [correctionField]` BitMask `0x08 [sourcePortIdentity]` BitMask `0x10 [sequenceId]` BitMask `0x20 [preciseOriginTimestamp]` BitMask `0x40 [reserved]` BitMask `0x80 [reserved]`
CRC_Time_0	1	3	`0..255`
CRC_Time_1	1	4	`0..255`
AUTOSAR TLV *Sub-TLV*: Status Secured			
Type	1	0	`0x50` [Status secured]
Length	1	1	`2`
Status	1	2	BitMask `0x01 [SGW with SyncToGTM = 0 SyncToSubDomain = 1]` BitMask `0x02 [reserved]` BitMask `0x04 [reserved]` BitMask `0x08 [reserved]` BitMask `0x10 [reserved]` BitMask `0x20 [reserved]` BitMask `0x40 [reserved]` BitMask `0x80 [reserved]`
CRC_Status	1	3	`0..255`
AUTOSAR TLV *Sub-TLV*: Status Not Secured			
Type	1	0	`0x51` [Status not secured]
Length	1	1	`2`
Status	1	2	BitMask `0x01 [SGW with SyncToGTM = 0 SyncToSubDomain = 1]` BitMask `0x02 [reserved]` BitMask `0x04 [reserved]` BitMask `0x08 [reserved]` BitMask `0x10 [reserved]` BitMask `0x20 [reserved]` BitMask `0x40 [reserved]` BitMask `0x80 [reserved]`
reserved	1	3	`0`

AUTOSAR TLV *Sub-TLV*: UserData Secured			
Type	1	0	0x60 [UserData secured]
Length	1	1	5
UserDataLength	1	2	0..3 (default: 0)
UserByte_0	1	3	0..255 (default: 0)
UserByte_1	1	4	0..255 (default: 0)
UserByte_2	1	5	0..255 (default: 0)
CRC_UserData	1	6	0..255
AUTOSAR TLV *Sub-TLV*: UserData Not Secured			
Type	1	0	0x61 [UserData not secured]
Length	1	1	5
UserDataLength	1	2	0..3 (default: 0)
UserByte_0	1	3	0..255 (default: 0)
UserByte_1	1	4	0..255 (default: 0)
UserByte_2	1	5	0..255 (default: 0)
reserved	1	6	0
AUTOSAR TLV *Sub-TLV*: OFS Secured			
Type	1	0	0x44 [OFS secured]
Length	1	1	17
OfsTimeDomain	1	2	16..31
OfsTimeSec	6	3	0..281474976710655s
OfsTimeNSec	4	9	0..999999999ns
Status	1	13	BitMask 0x01 [SGW with SyncToGTM = 0 SyncToSubDomain = 1] BitMask 0x02 [reserved] BitMask 0x04 [reserved] BitMask 0x08 [reserved] BitMask 0x10 [reserved] BitMask 0x20 [reserved] BitMask 0x40 [reserved] BitMask 0x80 [reserved]

UserDataLength	1	14	0..3 (default: 0)
UserByte_0	1	15	0..255 (default: 0)
UserByte_1	1	16	0..255 (default: 0)
UserByte_2	1	17	0..255 (default: 0)
CRC_OFS	1	18	0..255
AUTOSAR TLV *Sub-TLV*: OFS Not Secured			
Type	1	0	0x34 [OFS not secured]
Length	1	1	17
OfsTimeDomain	1	2	16..31
OfsTimeSec	6	3	0..281474976710655s
OfsTimeNSec	4	9	0..999999999ns
Status	1	13	BitMask 0x01 [SGW with SyncToGTM = 0 SyncToSubDomain = 1] BitMask 0x02 [reserved] BitMask 0x04 [reserved] BitMask 0x08 [reserved] BitMask 0x10 [reserved] BitMask 0x20 [reserved] BitMask 0x40 [reserved] BitMask 0x80 [reserved]
UserDataLength	1	14	0..3 (default: 0)
UserByte_0	1	15	0..255 (default: 0)
UserByte_1	1	16	0..255 (default: 0)
UserByte_2	1	17	0..255 (default: 0)
reserved	1	18	0

Figure 8: `Follow_Up` [AUTOSAR]

⌋(SRS_StbM_20048, SRS_StbM_20061, SRS_StbM_20062)

7.5.2.1 `Follow_Up` Message Header [AUTOSAR]

[SWS_EthTSyn_00066]⌈
The `messageLength` of the `Follow_Up` Message Header has to be adapted according to the length of all existing *TLV's*.
⌋(SRS_StbM_20048, SRS_StbM_20061, SRS_StbM_20062)

7.5.2.2 AUTOSAR *TLV* Header

[SWS_EthTSyn_00067]⌈
The AUTOSAR *TLV* Header has a multiplicity of 1.
⌋(SRS_StbM_20048, SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00068]⌈
If an AUTOSAR *TLV* Header exists, at least one AUTOSAR *Sub-TLV* must exist as well.
⌋(SRS_StbM_20048, SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00069]⌈
If an AUTOSAR *TLV* Header exists, the `lengthField` shall be adapted according the number of existing AUTOSAR *Sub-TLV's*.
⌋(SRS_StbM_20048, SRS_StbM_20061, SRS_StbM_20062)

7.5.2.3 AUTOSAR TLV *Sub-TLV's*

[SWS_EthTSyn_00070]⌈
If an AUTOSAR *Sub-TLV* exists, it shall be placed after the AUTOSAR *TLV* Header.
⌋(SRS_StbM_20048, SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00071]⌈
If more than one AUTOSAR *Sub-TLV* exists, each *Sub-TLV* shall be placed after the preceding *Sub-TLV* without gaps.
⌋(SRS_StbM_20048, SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00072]⌈
If more than one AUTOSAR *Sub-TLV* exists, the positon of each *Sub-TLV* is arbitrary.
⌋(SRS_StbM_20048, SRS_StbM_20061, SRS_StbM_20062)

7.5.2.3.1 AUTOSAR TLV *Sub-TLV*: Time Secured

[SWS_EthTSyn_00074]⌈
The AUTOSAR *Sub-TLV*: Time Secured has a multiplicity of 1 and is only available, if *CRC* protection is required.
⌋(SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00075]⌈
If `EthTSynMessageCompliance` (**ECUC_EthTSyn_00029 :**) is set to `FALSE` and `EthTSynTLVFollowUpTimeSubTLV` (**ECUC_EthTSyn_00035 :**) is set to `TRUE`, the Time Master shall send a `Follow_Up,` which contains an AUTOSAR *Sub-TLV*: Time Secured.
⌋(SRS_StbM_20048, SRS_StbM_20061, SRS_StbM_20062)

7.5.2.3.2 AUTOSAR TLV *Sub-TLV*: Status Secured / Not Secured

[SWS_EthTSyn_00076]⌈

The AUTOSAR *Sub-TLV*: Status has a multiplicity of 1 and can either be *CRC* protected (Status Secured) or not (Status Not Secured).
⌋(SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00077]⌈
If `EthTSynMessageCompliance` (**ECUC_EthTSyn_00029 :**) is set to `FALSE` and `EthTSynTLVFollowUpStatusSubTLV` (**ECUC_EthTSyn_00036 :**) is set to `TRUE`, the Time Master shall send a `Follow_Up,` which contains an AUTOSAR *Sub-TLV*: Status.
⌋(SRS_StbM_20048, SRS_StbM_20061, SRS_StbM_20062)

7.5.2.3.3 AUTOSAR TLV *Sub-TLV*: UserData Secured / Not Secured

[SWS_EthTSyn_00078]⌈
The AUTOSAR *Sub-TLV*: UserData has a multiplicity of 1 and can either be *CRC* protected (UserData Secured) or not (UserData Not Secured).
⌋(SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00079]⌈
If `EthTSynMessageCompliance` (**ECUC_EthTSyn_00029 :**) is set to `FALSE` and `EthTSynTLVFollowUpUserDataSubTLV` (**ECUC_EthTSyn_00037 :**) is set to `TRUE`, the Time Master shall send a `Follow_Up,` which contains an AUTOSAR *Sub-TLV*: UserData.
⌋(SRS_StbM_20048, SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00080]⌈
The AUTOSAR *Sub-TLV*: UserData shall be mapped to the `StbM_UserDataType`, whereas the User Byte number given in the message and by the `StbM_UserDataType` shall match (`UserByte_0` mapped to `StbM_UserDataType.userByte0` etc.).
The `StbM_UserDataType.userDataLength` shall be set accordingly.
⌋(SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00153]⌈
If `StbM_UserDataType.userDataLength` is set to 0 the complete AUTOSAR *Sub-TLV*: UserData shall be excluded from the message.
⌋(SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00081]⌈
The AUTOSAR *Sub-TLV*: UserData shall be read from the current incoming message consistently.
⌋(SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00082]⌈
The AUTOSAR *Sub-TLV*: UserData shall be written to the next outgoing message consistently.
⌋(SRS_StbM_20061, SRS_StbM_20062)

7.5.2.3.4 AUTOSAR TLV *Sub-TLV*: OFS Secured / Not Secured

[SWS_EthTSyn_00084]⌈
The AUTOSAR *Sub-TLV*: OFS has a multiplicity of 16 and can either be *CRC* protected (OFS Secured) or not (OFS Not Secured).
⌋(SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00085]⌈
The element `OfsTimeDomain` of the AUTOSAR *Sub-TLV*: OFS shall contain the Offset Time Domain identifier, which is in a range between 16 and 31.
⌋(SRS_StbM_20061, SRS_StbM_20062)

Note: Compared to CAN and FlexRay, Ethernet does need any optimization on payload bytes on bit-level.

[SWS_EthTSyn_00086]⌈
If `EthTSynMessageCompliance` (**ECUC_EthTSyn_00029 :**) is set to `FALSE` and `EthTSynTLVFollowUpOFSSubTLV` (**ECUC_EthTSyn_00038 :**) is set to `TRUE`, the Time Master shall send a `Follow_Up,` which contains at least one AUTOSAR *Sub-TLV*: OFS.
⌋(SRS_StbM_20048, SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00087]⌈
The User Data of the AUTOSAR *Sub-TLV*: OFS shall be mapped to the `StbM_UserDataType`, whereas the byte number given in the message and by the `StbM_UserDataType` shall match (`UserByte_0` mapped to `StbM_UserDataType.userByte0` etc.).
The `StbM_UserDataType.userDataLength` shall be set accordingly.
⌋(SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00088]⌈
The User Data of the AUTOSAR *Sub-TLV*: OFS shall be read from an incoming message consistently.
⌋(SRS_StbM_20061, SRS_StbM_20062)

[SWS_EthTSyn_00089]⌈
The User Data of the AUTOSAR *Sub-TLV*: OFS shall be written to an outgoing message consistently.
⌋(SRS_StbM_20061, SRS_StbM_20062)

7.6 Acting as Time Master

A Time Master is an entity which is the master for a certain Time Base and which propagates this Time Base to a set of Time Slaves within a certain segment of a communication network, being a source for this Time Base.
If a Time Master is also the owner of the Time Base then he is the Global Time master. A time gateway typically consists of one Time Slave and one or more Time Masters. When mapping time entities to real ECUs, an ECU could be Time Master

(or even Global Time Master) for one Time Base and Time Slave for another Time Base.

7.6.1 Message processing

[SWS_EthTSyn_00050]⌈
The Time Master shall support the transmission of Sync and Follow_Up according [12] as well as the transmission and reception of Pdelay_Req, Pdelay_Resp and Pdelay_Resp_Follow_Up (**[SWS_EthTSyn_00003]**, **[SWS_EthTSyn_00004]**).
⌋(SRS_StbM_20047, SRS_StbM_20048)

[SWS_EthTSyn_00016]⌈
The Time Master shall periodically transmit Sync with the cycle EthTSynGlobalTimeTxPeriod (**ECUC_EthTSyn_00010 :**) as defined in [12] chapter 11.1.3 “Transport of time-synchronization information”, if the GLOBAL_TIME_BASE bit within the timeBaseStatus, which is read from StbM, is set and EthTSynGlobalTimeTxPeriod (**ECUC_EthTSyn_00010 :**) is not 0.

For that, the following sequence shall be applied:

1. Get a free transmission buffer via EthIf_ProvideTxBuffer()
2. Activate the time stamping via EthIf_EnableEgressTimeStamp() if EthTSynHardwareTimestampSupport (**ECUC_EthTSyn_00018 :**) is set to TRUE
3. Trigger transmit request via EthIf_Transmit()

⌋(SRS_StbM_20047, SRS_StbM_20048)

Note: The timeBaseStatus can be read from StbM by StbM_GetTimeBaseStatus() or StbM_GetCurrentTime().

[SWS_EthTSyn_00126]⌈
On invocation of EthTSyn_TxConfirmation() the Global Time shall be retrieved from the StbM via StbM_GetCurrentTime() on egress of Sync according to Figure 18.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00017]⌈
On invocation of EthTSyn_TxConfirmation() a reference time shall be retrieved for Sync via EthIf_GetCurrentTime()from the EthIf used for transmission delay compensation of the same message according to Figure 18, if EthTSynHardwareTimestampSupport (**ECUC_EthTSyn_00018 :**) is set to TRUE.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00127]⌈
On invocation of EthTSyn_TxConfirmation() the egress time stamp shall be retrieved for Sync via EthIf_GetEgressTimeStamp() from the EthIf used for transmission delay compensation of the same message according to Figure 18, if

EthTSynHardwareTimestampSupport (**ECUC_EthTSyn_00018 :**) is set to TRUE.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00018]⌈
The preciseOriginTimestamp and the result of **[SWS_EthTSyn_00017]** and **[SWS_EthTSyn_00127]** shall be used in the transmission of the Follow_Up as defined in [12] chapter 11.1.3 “Transport of time-synchronization information” considering debounceCounter which represents a time offset between Sync and Follow_Up. For that, the following sequence shall apply:

1. Get a free transmission buffer via EthIf_ProvideTxBuffer()
2. Trigger transmit request with the transmission timestamp of **[SWS_EthTSyn_00017]** via EthIf_Transmit()

⌋(SRS_StbM_20048)

7.6.1.1 Runtime Error detection

[SWS_EthTSyn_00145]⌈
If EthTSynMasterSlaveConflictDetection (**ECUC_EthTSyn_00075 :**) is set to TRUE and if the Time Master receives a Sync message from another Time Master, it shall report a runtime error by calling Det_ReportRuntimeError(ETHTSYN_E_TMCONFLICT) and discard the received Sync message.
⌋(SRS_StbM_20051)

7.6.1.2 Debounce Time

[SWS_EthTSyn_00165]⌈
A Follow_Up shall be sent immediately, right after the Sync transmission, if the corresponding debounceCounter has reached 0.
⌋(SRS_StbM_20047)

7.6.1.3 Immediate Time Synchronization

In addition to the standard cyclic message transmission, an immediate message transmission might be required. Depending on configuration, the EthTSyn module checks on each EthTSyn_MainFunction() call the necessity for a Timesync message transmission for each Time Base, where a Master Port belongs to.

[SWS_EthTSyn_00134]⌈
If EthTSynImmediateTimeSync (**ECUC_EthTSyn_00046 :**) is set to TRUE, EthTSyn shall check within each EthTSyn_MainFunction() call by calling StbM_GetTimeBaseUpdateCounter() if the returned timeBaseUpdateCounter has been changed.
⌋(SRS_StbM_20047)

[SWS_EthTSyn_00135]⌈
If

- EthTSynImmediateTimeSync (**ECUC_EthTSyn_00046 :**) is set to TRUE
- and the timeBaseUpdateCounter[timeBaseId] for the updated Time Base resp. timeBaseId has been changed
- and the GLOBAL_TIME_BASE bit within the timeBaseStatus, which is read from StbM, is set,

EthTSyn shall trigger an immediate transmission of Time Synchronization messages belonging to this Time Base.
⌋(SRS_StbM_20047)

Note: The timeBaseStatus can be read from StbM by StbM_GetTimeBaseStatus() or StbM_GetCurrentTime().
The debounceCounter as described in 7.3 has always to be considered.

[SWS_EthTSyn_00136]⌈
If EthTSynImmediateTimeSync (**ECUC_EthTSyn_00046 :**) is set to TRUE, EthTSynCyclicMsgResumeTime (**ECUC_EthTSyn_00047 :**) shall be considered.
⌋(SRS_StbM_20047)

[SWS_EthTSyn_00137]⌈
EthTSynCyclicMsgResumeTime (**ECUC_EthTSyn_00047 :**) represents the timeout value of a cyclicMsgResumeCounter that shall be started when a Sync has been sent immediately, asynchronous to the cyclic transmission. The cyclicMsgResumeCounter shall be decremented on each invocation of EthTSyn_MainFunction() if no Timesync PDU is transmitted asynchronously.
⌋(SRS_StbM_20047)

[SWS_EthTSyn_00139]⌈
If the cyclicMsgResumeCounter has reached a value equal or less than 0, EthTSyn shall resume cyclic Timesync message transmission by sending a Sync.
⌋(SRS_StbM_20047)

7.6.2 Link State and Transmission Mode

[SWS_EthTSyn_00019]⌈
A transceiver link state change (notification call of EthTSyn_TrcvLinkStateChg()) from ETHTRCV_LINK_STATE_ACTIVE to ETHTRCV_LINK_STATE_DOWN resets the state machines for transmission and reception of Time Synchronization messages.
⌋(SRS_StbM_20048, SRS_StbM_20051)

[SWS_EthTSyn_00020]⌈
A transceiver link state change (notification call of EthTSyn_TrcvLinkStateChg()) from ETHTRCV_LINK_STATE_DOWN to ETHTRCV_LINK_STATE_ACTIVE (re-)starts the transmission and reception of Time Synchronization messages.
⌋(SRS_StbM_20048, SRS_StbM_20051)

[SWS_EthTSyn_00021]⌈
If `EthTSyn_SetTransmissionMode()` is called and the parameter `Mode` equals `ETHTSYN_TX_OFF`, all transmit request from EthTSyn shall be omitted on this Ethernet controller.
⌋(SRS_StbM_20048, SRS_StbM_20051)

[SWS_EthTSyn_00022]⌈
If `EthTSyn_SetTransmissionMode()` is called and the parameter `Mode` equals `ETHTSYN_TX_ON`, all transmit request from EthTSyn on this Ethernet controller shall be able to be transmitted.
⌋(SRS_StbM_20048, SRS_StbM_20051)

7.6.3 Message Field Calculation and Assembling

[SWS_EthTSyn_00092]⌈
If `EthTSynMessageCompliance` (**ECUC_EthTSyn_00029 :**) is set to `FALSE`, a Time Master shall add an AUTOSAR *TLV* to the `Follow_Up` frame.
⌋(SRS_StbM_20061, SRS_StbM_20062, SRS_StbM_20063)

[SWS_EthTSyn_00091]⌈
If `EthTSynMessageCompliance` (**ECUC_EthTSyn_00029 :**) is set to `FALSE`, `EthTSynGlobalTimeTxCrcSecured` (**ECUC_EthTSyn_00039 :**) shall be considered.
⌋(SRS_StbM_20061)

[SWS_EthTSyn_00093]⌈
Depending on `EthTSynGlobalTimeTxCrcSecured` (**ECUC_EthTSyn_00039 :**) the `Follow_Up.TLV[AUTOSAR].Sub-TLV.Type` shall be:

	`Sub-TLV.Type`	
`EthTSynGlobalTimeTxCrcSecured`	`CRC_SUPPORTED`	`CRC_NOT_SUPPORTED`
	`0x28` Sub-TLV: Time Secured is *CRC* secured	`n.a.`
	`0x50` Sub-TLV: Status is *CRC* secured	**`0x51`** Sub-TLV: Status is not *CRC* secured
	`0x60` Sub-TLV: UserData is *CRC* secured	**`0x61`** Sub-TLV: UserData is not *CRC* secured
	`0x44` Sub-TLV: OFS is *CRC* secured	**`0x34`** Sub-TLV: OFS is not *CRC* secured

⌋(SRS_StbM_20061)

7.6.3.1 SGW Calculation

[SWS_EthTSyn_00094]⌈

The *SGW* value (Time Gateway synchronization status) shall be mapped to the `Status` element of the AUTOSAR *Sub-TLV*: Status resp. the AUTOSAR *Sub-TLV*: OFS.
If the `SYNC_TO_GATEWAY` bit within `timeBaseStatus` is set, the *SGW* value shall be *SyncToSubDomain*. Otherwise, it shall be *SyncToGTM*.
⌋(SRS_StbM_20052)

7.6.3.2 OFS Calculation

[SWS_EthTSyn_00095]⌈
The transmitter of an Offset Time Base (Time Master) shall perform the following steps to distribute the Offset Time Base:

1. Get second portion of the Offset Time Base via `StbM_GetOffset()` and write to `OfsTimeSec` element of the corresponding AUTOSAR *Sub-TLV*: OFS.
2. Use nanosecond portion of the Offset Time Base and write to `OfsTimeNSec` element of the corresponding AUTOSAR *Sub-TLV*: OFS.

⌋(SRS_StbM_20063)

7.6.3.3 CRC Calculation

[SWS_EthTSyn_00096]⌈
The function `Crc_CalculateCRC8H2F()` as defined in [10] shall be used to calculate the *CRC* if configured.
⌋(SRS_StbM_20061)

[SWS_EthTSyn_00097]⌈
The `DataID` shall be calculated as:
`DataID = DataIDList[Follow_Up.sequenceId` mod 16`]`, where `DataIDList` is given by configuration (**ECUC_EthTSyn_00030 :**) for the `Follow_Up`.
⌋(SRS_StbM_20061)

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

[SWS_EthTSyn_00182]⌈
If applying the *CRC* calculation on multibyte values, the byte order shall be such, that the byte containing the most significant bit of the value shall be used first.
⌋(SRS_StbM_20061)

[SWS_EthTSyn_00184]⌈
If applying the *CRC* calculation on multibyte message data, the byte order shall be in ascending order of the octets, i.e., the octet with the lowest offset shall be used first.
⌋(SRS_StbM_20061)

7.6.3.3.1 AUTOSAR TLV *Sub-TLV*: Time Secured

[SWS_EthTSyn_00098]⌈

If EthTSynGlobalTimeTxCrcSecured (**ECUC_EthTSyn_00039 :**) is set to CRC_SUPPORTED, the Time Master shall write the contents of EthTSynCrcTimeFlagsTxSecured (**ECUC_EthTSyn_00057 :**) to CRC_Time_Flags acc. to the following rule:

	EthTSynCrcTimeFlagsTxSecured contents:	
CRC_Time_Flags	Follow_Up Message Header	Follow_Up Message Field
BitMask 0x01	EthTSynCrcMessageLength (**ECUC_EthTSyn_00040 :**)	n.a.
BitMask 0x02	EthTSynCrcDomainNumber (**ECUC_EthTSyn_00041 :**)	n.a.
BitMask 0x04	EthTSynCrcCorrectionField (**ECUC_EthTSyn_00042 :**)	n.a.
BitMask 0x08	EthTSynCrcSourcePortIdentity (**ECUC_EthTSyn_00043 :**)	n.a.
BitMask 0x10	EthTSynCrcSequenceId (**ECUC_EthTSyn_00044 :**)	n.a.
BitMask 0x20	n.a.	EthTSynCrcPreciseOriginTimestamp (**ECUC_EthTSyn_00045 :**)
BitMask 0x40	n.a.	n.a.
BitMask 0x80	n.a.	n.a.

⌋(SRS_StbM_20061)

[SWS_EthTSyn_00099]⌈
If EthTSynGlobalTimeTxCrcSecured (**ECUC_EthTSyn_00039 :**) is set to CRC_SUPPORTED, the Time Master shall calculate the *CRC* for CRC_Time_0 by considering the contents of CRC_Time_Flags itself, the contents of the dependent fields as defined in EthTSynCrcTimeFlagsTxSecured (**ECUC_EthTSyn_00057 :**) acc. to the rule in the table below and the DataID.

	For CRC_Time_0 calculation considered contents:	
If CRC_Time_Flags is set to 1	Follow_Up Message Header	Follow_Up Message Field
BitMask 0x01	n.a.	n.a.
BitMask 0x02	domainNumber	n.a.
BitMask 0x04	n.a.	n.a.
BitMask 0x08	sourcePortIdentity	n.a.
BitMask 0x10	n.a.	n.a.
BitMask 0x20	n.a.	preciseOriginTimestamp
BitMask 0x40	n.a.	n.a.
BitMask 0x80	n.a.	n.a.

The data elements used for the calculation of the *CRC* shall apply the following order:

1. the value of CRC_Time_Flags
2. the domainNumber inside the Follow_Up Message Header, if CRC_Time_Flags contains BitMask 0x02

3. the sourcePortIdentity inside the Follow_Up Message Header, if CRC_Time_Flags contains BitMask 0x08
4. the preciseOriginTimestamp inside the Follow_Up Message Field, if CRC_Time_Flags contains BitMask 0x20
5. the DataID (refer to **[SWS_EthTSyn_00097]**)

⌋(SRS_StbM_20061)

Note: CRC_Time_Flags is having the same value like the configuration item EthTSynCrcTimeFlagsTxSecured, whereas the resulting *CRC* of the dependent items remains network wide unchanged.

[SWS_EthTSyn_00100]⌈
If EthTSynGlobalTimeTxCrcSecured (**ECUC_EthTSyn_00039 :**) is set to CRC_SUPPORTED, the Time Master shall calculate the *CRC* for CRC_Time_1 by considering the contents of CRC_Time_Flags itself, the contents of the dependent fields as defined in EthTSynCrcTimeFlagsTxSecured (**ECUC_EthTSyn_00057 :**) acc. to the rule in the table below and the DataID.

	For CRC_Time_1 calculation considered contents:	
If CRC_Time_Flags is set to 1	Follow_Up Message Header	Follow_Up Message Field
BitMask 0x01	messageLength	n.a.
BitMask 0x02	n.a.	n.a.
BitMask 0x04	correctionField	n.a.
BitMask 0x08	n.a.	n.a.
BitMask 0x10	sequenceId	n.a.
BitMask 0x20	n.a.	n.a.
BitMask 0x40	n.a.	n.a.
BitMask 0x80	n.a.	n.a.

The data elements used for the calculation of the *CRC* shall apply the following order:

1. the value of CRC_Time_Flags
2. the messageLength inside the Follow_Up Message Header, if CRC_Time_Flags contains BitMask 0x01
3. the correctionField inside the Follow_Up Message Header, if CRC_Time_Flags contains BitMask 0x04
4. the sequenceId inside the Follow_Up Message Header, if CRC_Time_Flags contains BitMask 0x10
5. the DataID (refer to **[SWS_EthTSyn_00097]**)

⌋(SRS_StbM_20061)

Note: CRC_Time_Flags has the same value as the configuration item EthTSynCrcTimeFlagsTxSecured.

7.6.3.3.2 AUTOSAR TLV *Sub-TLV*: Status secured

[SWS_EthTSyn_00101]⌈
If EthTSynGlobalTimeTxCrcSecured (**ECUC_EthTSyn_00039 :**) is set to CRC_SUPPORTED, the Time Master shall calculate the *CRC* for CRC_Status by considering the contents of Status and DataID (in this order).
⌋(SRS_StbM_20061)

7.6.3.3.3 AUTOSAR TLV *Sub-TLV*: UserData secured

[SWS_EthTSyn_00102]⌈
If EthTSynGlobalTimeTxCrcSecured (**ECUC_EthTSyn_00039 :**) is set to CRC_SUPPORTED, the Time Master shall calculate the *CRC* for CRC_UserData by considering the contents of UserDataLength, UserByte_0, UserByte_1, UserByte_2 and DataID (in this order).
⌋(SRS_StbM_20061)

7.6.3.3.4 AUTOSAR TLV *Sub-TLV*: OFS secured

[SWS_EthTSyn_00103]⌈
If EthTSynGlobalTimeTxCrcSecured (**ECUC_EthTSyn_00039 :**) is set to CRC_SUPPORTED, the Time Master shall calculate the *CRC* for CRC_OFS by considering the contents of OfsTimeDomain, OfsTimeSec, OfsTimeNSec, Status, UserDataLength, UserByte_0, UserByte_1, UserByte_2 and DataID (in this order).
⌋(SRS_StbM_20061, SRS_StbM_20062, SRS_StbM_20063)

7.6.3.4 Message Assembling

[SWS_EthTSyn_00104]⌈
For each transmission of a Time Synchronization message, the EthTSyn module shall assemble the message as follows:

1. If Sync: Calculate Message Header
2. If Follow_Up: Calculate Follow_Up.preciseOriginTimestamp and Message Header inclusive correctionField
3. If Follow_Up: Calculate IEEE *TLV*
4. If Follow_Up: Calculate AUTOSAR *TLV* (configuration dependent)
    a. Calculate *CRC* (configuration dependent)
5. Copy all data to the appropriate position within the related message

⌋(SRS_StbM_20061, SRS_StbM_20062, SRS_StbM_20063)

7.7 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.

7.7.1 Message processing

[SWS_EthTSyn_00023]⌈
The Time Slave shall support the reception of Sync and Follow_Up according [12] as well as the transmission and reception of Pdelay_Req, Pdelay_Resp and Pdelay_Resp_Follow_Up (**[SWS_EthTSyn_00003]**, **[SWS_EthTSyn_00004]**).
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00128]⌈
On invocation of EthTSyn_RxIndication the ingress time stamp shall be retrieved for Sync via EthIf_GetIngressTimeStamp() from the EthIf used for reception delay compensation of the time synchronization process according to Figure 19, if EthTSynHardwareTimestampSupport (**ECUC_EthTSyn_00018 :**) is set to TRUE.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00180]⌈
On invocation of EthTSyn_RxIndication a reference time shall be retrieved for Sync via StbM_GetCurrentTimeRaw()from the StbM used for reception delay compensation of the time synchronization process according to Figure 19, if EthTSynHardwareTimestampSupport (**ECUC_EthTSyn_00018 :**) is set to FALSE.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00024]⌈
On invocation of EthTSyn_RxIndication()a reference time shall be retrieved for Follow_Up via EthIf_GetCurrentTime()from the EthIf used for reception delay compensation of the time synchronization process according to Figure 19, if EthTSynHardwareTimestampSupport (**ECUC_EthTSyn_00018 :**) is set to TRUE.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00179]⌈
On invocation of EthTSyn_RxIndication()a reference time difference shall be retrieved for Follow_Up via StbM_GetCurrentTimeDiff()from the StbM used for reception delay compensation of the time synchronization process according to Figure 19, if EthTSynHardwareTimestampSupport (**ECUC_EthTSyn_00018 :**) is set to FALSE.
⌋(SRS_StbM_20048)

[SWS_EthTSyn_00025]⌈
For each configured Time Slave the EthTSyn module shall observe the reception timeout EthTSynGlobalTimeFollowUpTimeout (**ECUC_EthTSyn_00007 :**) between the Sync and its Follow_Up.
If the reception timeout occurs, the sequence shall be reset (i.e. waiting for a new Sync). A value of 0 deactivates this timeout observation.
⌋(SRS_StbM_20048, SRS_StbM_20051)

Note: A timeout is detected when receiving the next subsequent Sync before receiving the Follow_Up belonging to the Sync before. The general timeout monitoring for the Time Base update is located in the StbM and not in the provider modules.

[SWS_EthTSyn_00052]⌈
For a valid Follow_Up a new Global Time value shall be calculated and forwarded to the StbM module via StbM_BusSetGlobalTime(), according to Figure 19, where the value pointed by timeStampPtr given within StbM_BusSetGlobalTime() considers the sum of:
preciseOriginTimestamp,
correctionField,
Pdelay and
the Sync reception delay.
⌋(SRS_StbM_20048)

Note: The Pdelay value is not influenced by a RateRatio acc to [12] Note-2 of chapter 11.2.15.2.4 “computePropTime():”.

[SWS_EthTSyn_00150]⌈
On an invocation of StbM_BusSetGlobalTime() the current Pdelay value shall be passed by the parameter measureDataPtr->PathDelay.
⌋(SRS_StbM_20058)

[SWS_EthTSyn_00129]⌈
When providing a new time to the StbM by calling StbM_BusSetGlobalTime(), EthTSyn shall set the SYNC_TO_GATEWAY bit in timeBaseStatus (structure member, which is referenced by the parameter timeStampPtr), according to the *SGW* value (refer to **[SWS_EthTSyn_00156]**). The remaining status bits shall be set to 0.
⌋(SRS_StbM_20051)

7.7.1.1 Runtime Error detection

[SWS_EthTSyn_00146]⌈
If EthTSynMasterSlaveConflictDetection (**ECUC_EthTSyn_00075 :**) is set to TRUE and if the Time Slave receives a Sync frame with different sourcePortIdentity (i.e. different MAC addresses), it shall report a runtime error by calling Det_ReportRuntimeError(ETHTSYN_E_TSCONFLICT) and discard the received Sync frame.
⌋(SRS_StbM_20051)

7.7.2 Message Field Validation and Disassembling

[SWS_EthTSyn_00105]⌈
If EthTSynMessageCompliance (**ECUC_EthTSyn_00029 :**) is set to FALSE, EthTSynRxCrcValidated (**ECUC_EthTSyn_00049 :**) shall be considered.

⌋(SRS_StbM_20061)

[SWS_EthTSyn_00106]⌈
If EthTSynMessageCompliance (**ECUC_EthTSyn_00029 :**) is set to FALSE, a Time Slave shall check if an AUTOSAR *TLV* in the Follow_Up frame exists.
⌋(SRS_StbM_20061, SRS_StbM_20062, SRS_StbM_20063)

[SWS_EthTSyn_00107]⌈
The *CRC* of the Follow_Up *TLV* shall be validated, depending on EthTSynRxCrcValidated (**ECUC_EthTSyn_00049 :**) and the Follow_Up.TLV[AUTOSAR].Sub-TLV.Type acc. to:

	Sub-TLV.Type	
EthTSynRxCrcValidated	CRC_VALIDATED	CRC_NOT_VALIDATED
	0x28 Sub-TLV: Time Secured is *CRC* secured	n.a.
	0x50 Sub-TLV: Status is *CRC* secured	**0x51** Sub-TLV: Status is not *CRC* secured
	0x60 Sub-TLV: UserData is *CRC* secured	**0x61** Sub-TLV: UserData is not *CRC* secured
	0x44 Sub-TLV: OFS is *CRC* secured	**0x34** Sub-TLV: OFS is not *CRC* secured

⌋(SRS_StbM_20061)

[SWS_EthTSyn_00108]⌈
The *CRC* of the Follow_Up *TLV* shall be ignored, if EthTSynRxCrcValidated (**ECUC_EthTSyn_00049 :**) is set to CRC_IGNORED and the Follow_Up.TLV[AUTOSAR].Sub-TLV.Type contains any of the following defined values:

	Sub-TLV.Type	
EthTSynRxCrcValidated	CRC_IGNORED	
	0x28 Sub-TLV: Time Secured is *CRC* secured	n.a.
	0x50 Sub-TLV: Status is *CRC* secured	**0x51** Sub-TLV: Status is not *CRC* secured
	0x60 Sub-TLV: UserData is *CRC* secured	**0x61** Sub-TLV: UserData is not *CRC* secured
	0x44 Sub-TLV: OFS is *CRC* secured	**0x34** Sub-TLV: OFS is not *CRC* secured

⌋(SRS_StbM_20061)

[SWS_EthTSyn_00109]⌈
The *CRC* of the Follow_Up *TLV* shall be either validated or not validated, if EthTSynRxCrcValidated (**ECUC_EthTSyn_00049 :**) is set to CRC_OPTIONAL

and the `Follow_Up.TLV[AUTOSAR].Sub-TLV.Type` contains any of the following defined values:

	`Sub-TLV.Type`	
`EthTSynRxCrcValidated`	`CRC_OPTIONAL`	
	CRC shall be validated	*CRC* shall not be validated
	`0x28` Sub-TLV: Time Secured is *CRC* secured	`n.a.`
	`0x50` Sub-TLV: Status is *CRC* secured	**`0x51`** Sub-TLV: Status is not *CRC* secured
	`0x60` Sub-TLV: UserData is *CRC* secured	**`0x61`** Sub-TLV: UserData is not *CRC* secured
	`0x44` Sub-TLV: OFS is *CRC* secured	**`0x34`** Sub-TLV: OFS is not *CRC* secured

⌋(SRS_StbM_20061)

7.7.2.1 SGW Calculation

[SWS_EthTSyn_00156]⌈
The *SGW* value (Time Gateway synchronization status) shall be retrieved from the `Status` element of the AUTOSAR *Sub-TLV*: Status resp. the AUTOSAR *Sub-TLV*: OFS.
If the *SGW* value is set to *SyncToSubDomain*, the `SYNC_TO_GATEWAY` bit within `timeBaseStatus` shall be set. Otherwise, it shall be zero.
⌋(SRS_StbM_20052)

7.7.2.2 OFS Calculation

[SWS_EthTSyn_00110]⌈
The receiver of an Offset Time Base (Time Slave) shall perform the following steps to calculate the Offset Time Base:

1. Retrieve second portion of the Offset Time Base from `OfsTimeSec` element of the corresponding AUTOSAR *Sub-TLV*: OFS.
2. Retrieve nanosecond portion of the Offset Time Base from `OfsTimeNSec` element of the corresponding AUTOSAR *Sub-TLV*: OFS.
3. Forward the new Offset Time to the StbM via `StbM_BusSetGlobalTime()`, if successfully validated.

⌋(SRS_StbM_20063)

7.7.2.3 CRC Validation

[SWS_EthTSyn_00111]⌈
The function `Crc_CalculateCRC8H2F()` as defined in [10] shall be used to calculate the *CRC* if configured.
⌋(SRS_StbM_20061)

[SWS_EthTSyn_00112]⌈
The DataID shall be calculated as:
DataID = DataIDList[Follow_Up.sequenceId mod 16], where DataIDList is given by configuration (**ECUC_EthTSyn_00030 :**) for the Follow_Up.
⌋(SRS_StbM_20061)

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

[SWS_EthTSyn_00183]⌈
If applying the *CRC* calculation on multibyte values, the byte order shall be such that the byte containing the most significant bit of the value shall be used first.
⌋(SRS_StbM_20061)

[SWS_EthTSyn_00185]⌈
If applying the *CRC* calculation on multibyte message data, the byte order shall be in ascending order of the octets, i.e., the octet with the lowest offset shall be used first.
⌋(SRS_StbM_20061)

7.7.2.3.1 AUTOSAR TLV *Sub-TLV*: Time Secured

[SWS_EthTSyn_00157]⌈
If EthTSynRxCrcValidated (**ECUC_EthTSyn_00049 :**) is set to CRC_VALIDATED or CRC_OPTIONAL, the Time Slave shall validate the *CRC* as defined in EthTSynCrcFlagsRxValidated (**ECUC_EthTSyn_00050 :**) acc. to the following rule:

	Validate if EthTSynCrcFlagsRxValidated element is set to TRUE:	
Element	Follow_Up Message Header	Follow_Up Message Field
EthTSynCrcMessageLength (**ECUC_EthTSyn_00051 :**)	messageLength	n.a.
EthTSynCrcDomainNumber (**ECUC_EthTSyn_00052 :**)	domainNumber	n.a.
EthTSynCrcCorrectionField (**ECUC_EthTSyn_00053 :**)	correctionField	n.a.
EthTSynCrcSourcePortIdentity (**ECUC_EthTSyn_00054 :**)	sourcePortIdentity	n.a.
EthTSynCrcSequenceId (**ECUC_EthTSyn_00055 :**)	sequenceId	n.a.
EthTSynCrcPreciseOriginTimestamp (**ECUC_EthTSyn_00056 :**)	n.a.	preciseOriginTimestamp

⌋(SRS_StbM_20061)

[SWS_EthTSyn_00113]⌈
If EthTSynRxCrcValidated (**ECUC_EthTSyn_00049 :**) is set to CRC_VALIDATED or CRC_OPTIONAL, the Time Slave shall validate the *CRC* for CRC_Time_0 by considering the contents of CRC_Time_Flags itself, the contents of

the dependent fields as defined in EthTSynCrcFlagsRxValidated (**ECUC_EthTSyn_00050 :**) acc. to the rule in the table below and the DataID.

	For CRC_Time_0 verification required contents:	
If EthTSynCrcFlagsRxValidated element is set to TRUE:	Follow_Up Message Header	Follow_Up Message Field
EthTSynCrcMessageLength (**ECUC_EthTSyn_00051 :**)	n.a.	n.a.
EthTSynCrcDomainNumber (**ECUC_EthTSyn_00052 :**)	domainNumber	n.a.
EthTSynCrcCorrectionField (**ECUC_EthTSyn_00053 :**)	n.a.	n.a.
EthTSynCrcSourcePortIdentity (**ECUC_EthTSyn_00054 :**)	sourcePortIdentity	n.a.
EthTSynCrcSequenceId (**ECUC_EthTSyn_00055 :**)	n.a.	n.a.
EthTSynCrcPreciseOriginTimestamp (**ECUC_EthTSyn_00056 :**)	n.a.	preciseOriginTimestamp

The data elements used for the calculation and thus validation of the *CRC* shall apply the following order:

1. the value of CRC_Time_Flags
2. the domainNumber inside the Follow_Up Message Header, if EthTSynCrcDomainNumber (**ECUC_EthTSyn_00052 :**) is set to TRUE
3. the sourcePortIdentity inside the Follow_Up Message Header, if EthTSynCrcSourcePortIdentity (**ECUC_EthTSyn_00054 :**) is set to TRUE
4. the preciseOriginTimestamp inside the Follow_Up Message Field, if EthTSynCrcPreciseOriginTimestamp (**ECUC_EthTSyn_00056 :**) is set to TRUE
5. the DataID (refer to **[SWS_EthTSyn_00112]**)

⌋(SRS_StbM_20061)

[SWS_EthTSyn_00114]⌈
If EthTSynRxCrcValidated (**ECUC_EthTSyn_00049 :**) is set to CRC_VALIDATED or CRC_OPTIONAL, the Time Slave shall validate the *CRC* for CRC_Time_1 by considering the contents of CRC_Time_Flags itself, the contents of the dependent fields as defined in EthTSynCrcFlagsRxValidated (**ECUC_EthTSyn_00050 :**) acc. to the rule in the table below and the DataID.

	For CRC_Time_1 verification required contents:	
If EthTSynCrcFlagsRxValidated element is set to TRUE:	Follow_Up Message Header	Follow_Up Message Field
EthTSynCrcMessageLength (**ECUC_EthTSyn_00051 :**)	messageLength	n.a.
EthTSynCrcDomainNumber (**ECUC_EthTSyn_00052 :**)	n.a.	n.a.
EthTSynCrcCorrectionField	correctionField	n.a.

(**ECUC_EthTSyn_00053 :**)		
EthTSynCrcSourcePortIdentity (**ECUC_EthTSyn_00054 :**)	n.a.	n.a.
EthTSynCrcSequenceId (**ECUC_EthTSyn_00055 :**)	sequenceId	n.a.
EthTSynCrcPreciseOriginTimestamp (**ECUC_EthTSyn_00056 :**)	n.a.	n.a.

The data elements used for the calculation and thus validation of the *CRC* shall apply the following order:

1. the value of CRC_Time_Flags
2. the messageLength inside the Follow_Up Message Header, if EthTSynCrcMessageLength (**ECUC_EthTSyn_00051 :**) is set to TRUE
3. the correctionField inside the Follow_Up Message Header, if EthTSynCrcCorrectionField (**ECUC_EthTSyn_00053 :**) is set to TRUE
4. the sequenceId inside the Follow_Up Message Header, if EthTSynCrcSequenceId (**ECUC_EthTSyn_00055 :**) is set to TRUE
5. the DataID (refer to **[SWS_EthTSyn_00112]**)

⌋(SRS_StbM_20061)

7.7.2.3.2 AUTOSAR TLV *Sub-TLV*: Status secured

[SWS_EthTSyn_00115]⌈
If EthTSynRxCrcValidated (**ECUC_EthTSyn_00049 :**) is set to CRC_VALIDATED or CRC_OPTIONAL, the Time Slave shall validate the *CRC* for CRC_Status by considering the contents of Status and DataID (in this order).
⌋(SRS_StbM_20061)

7.7.2.3.3 AUTOSAR TLV *Sub-TLV*: UserData secured

[SWS_EthTSyn_00116]⌈
If EthTSynRxCrcValidated (**ECUC_EthTSyn_00049 :**) is set to CRC_VALIDATED or CRC_OPTIONAL, the Time Slave shall validate the *CRC* for CRC_UserData by considering the contents of UserDataLength, UserByte_0, UserByte_1, UserByte_2 and DataID (in this order).
⌋(SRS_StbM_20061)

7.7.2.3.4 AUTOSAR TLV *Sub-TLV*: OFS secured

[SWS_EthTSyn_00117]⌈
If EthTSynRxCrcValidated (**ECUC_EthTSyn_00049 :**) is set to CRC_VALIDATED or CRC_OPTIONAL, the Time Slave shall validate the *CRC* for CRC_OFS by considering the contents of OfsTimeDomain, OfsTimeSec, OfsTimeNSec, Status, UserDataLength, UserByte_0, UserByte_1, UserByte_2 and DataID (in this order).
⌋(SRS_StbM_20061, SRS_StbM_20063)

7.7.2.4 Message Disassembling

[SWS_EthTSyn_00118]⌈
If the Type of a *Sub-TLV* cannot be recognized at the receiver side, it shall be ignored and the next subsequent *Sub-TLV* shall be evaluated.
⌋(SRS_StbM_20061, SRS_StbM_20062, SRS_StbM_20063)

Note: The Length field of each *Sub-TLV* is always at the same position within each *Sub-TLV*. It will be used to jump over the unknown *Sub-TLV* to the next Type field.

[SWS_EthTSyn_00119]⌈
For each received Time Synchronization message, the EthTSyn module shall validate the message as follows (all conditions must match):

1. If Follow_Up: The sequenceId of the Follow_Up matches the sequenceId of the corresponding Sync.
2. If Follow_Up: Follow_Up.TLV[AUTOSAR].Sub-TLV.Type matches depending on configuration of EthTSynRxCrcValidated (**ECUC_EthTSyn_00049 :**)
3. The Time Domain matches to the defined Time Domain range for each domainNumber resp. to the element OfsTimeDomain of the AUTOSAR *Sub-TLV*: OFS (configuration dependent).
4. The Time Domain matches to one of the configured Time Domains.
5. If Follow_Up: The range of the element OfsTimeNSec of the AUTOSAR *Sub-TLV*: OFS matches the defined range of StbM_TimeStampType.nanoseconds.
6. If Follow_Up: All *CRC's* (including DataID) matching depending on the configuration of EthTSynRxCrcValidated (**ECUC_EthTSyn_00049 :**) and EthTSynCrcFlagsRxValidated (**ECUC_EthTSyn_00050 :**).

⌋(SRS_StbM_20061, SRS_StbM_20062, SRS_StbM_20063)

[SWS_EthTSyn_00120]⌈
For each received Time Synchronization message, the EthTSyn module shall disassemble the message after successful validation **[SWS_EthTSyn_00119]**.
⌋(SRS_StbM_20061, SRS_StbM_20062, SRS_StbM_20063)

7.8 Time measurement with Switches

In a time aware Ethernet network, two basic HW types of control units exists:

1. Endpoints directly working on a local Ethernet-Controller
2. Time Gateways, resp. Time Aware Bridges, where the local Ethernet-Controller connects to an external Switch device.

The extension “Time measurement with Switches” focusses on 2.
A Switch device leads to additional delays, which have to be considered for the calculation of the Synchronized Time Base. Additionally, the support of time stamping in HW is a Switch-Port specific feature, which leads to an extension of the used

function APIs. These APIs enabling a Switch port specific detection of ingress and egress messages together with a given timestamp, if enabled.

If the Switch Management and Global Time support is implemented as a part of the program running on the Switch HW, this will not be considered by 2. For this case, the behavior can be seen as described in 1.

[SWS_EthTSyn_00053]⌈
Time measurement with Switches supports the use case “Time Aware Bridge with GTM as Management CPU” like shown in Figure 9.

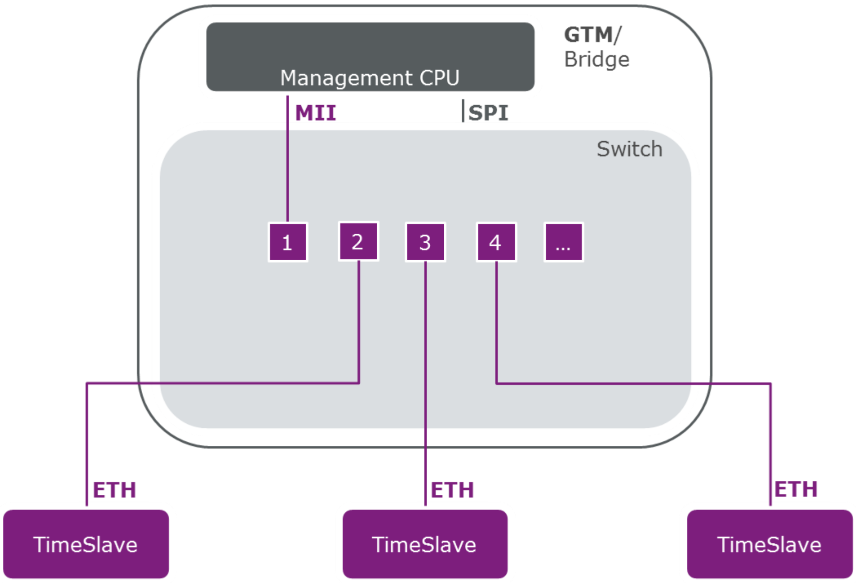


Figure 9: Time Aware Bridge with GTM as Management CPU

⌋(SRS_StbM_20048, SRS_StbM_20059)

[SWS_EthTSyn_00054]⌈
Time measurement with Switches supports the use case “Time Aware Bridge with GTM not as Management CPU” like shown in Figure 10.

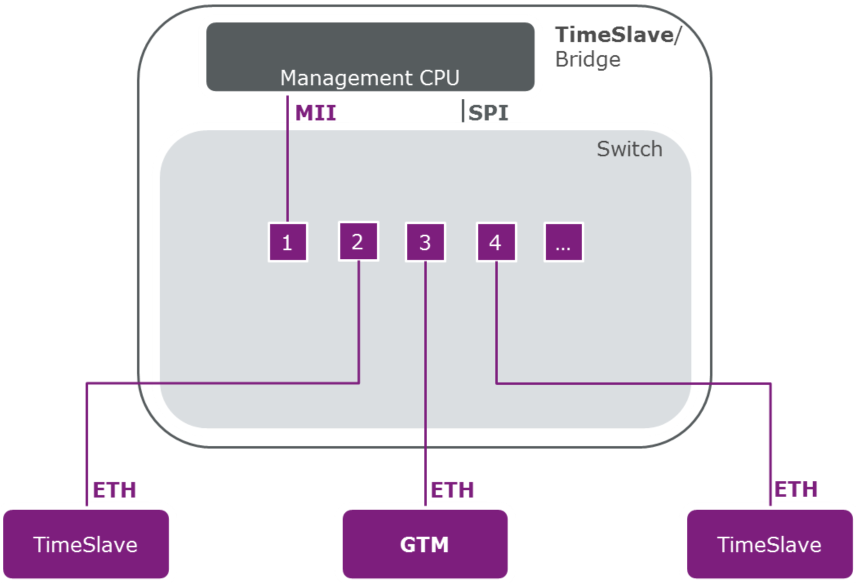


Figure 10: Time Aware Bridge with GTM not as Management CPU

⌋(SRS_StbM_20048, SRS_StbM_20059)

7.8.1 `Pdelay` and Time Synchronization measurement point

[SWS_EthTSyn_00055]⌈
The path delay measurement will be done always as Port-to-Port measurement like specified in in [12] chapter 11.1.2 "Propagation delay measurement" for the device external Ethernet path.
⌋(SRS_StbM_20048, SRS_StbM_20059)

[SWS_EthTSyn_00056]⌈
The inner delay of the Ethernet path (Residence Time) is determined at the time where `Sync` is received and transmitted, by using the message specific ingress and egress timestamps.
⌋(SRS_StbM_20048, SRS_StbM_20059)

Note: This belongs to the fact, that the Residence Time might be discontinuous, depending on the current busload, while `Sync` messages are transmitted / received, the Switch HW architecture and the message forwarding method. A static delay measurement method for this part of the communication path might lead to an unprecise time measurement. Nevertheless, static Residence Time parameters are considered by this specification, to increase the performance while calculating the Global Time resp. the `correctionField` and the flexibility to support different Switch devices, such as Switches, which do not support time stamping on each ingress or egress port.

7.8.2 Use case "Time Aware Bridge with GTM as Management CPU"

[SWS_EthTSyn_00057]⌈

Time measurement with Switches supporting the use case “Time Aware Bridge with GTM as Management CPU” following the given timestamping points like shown in Figure 11 and Figure 12.

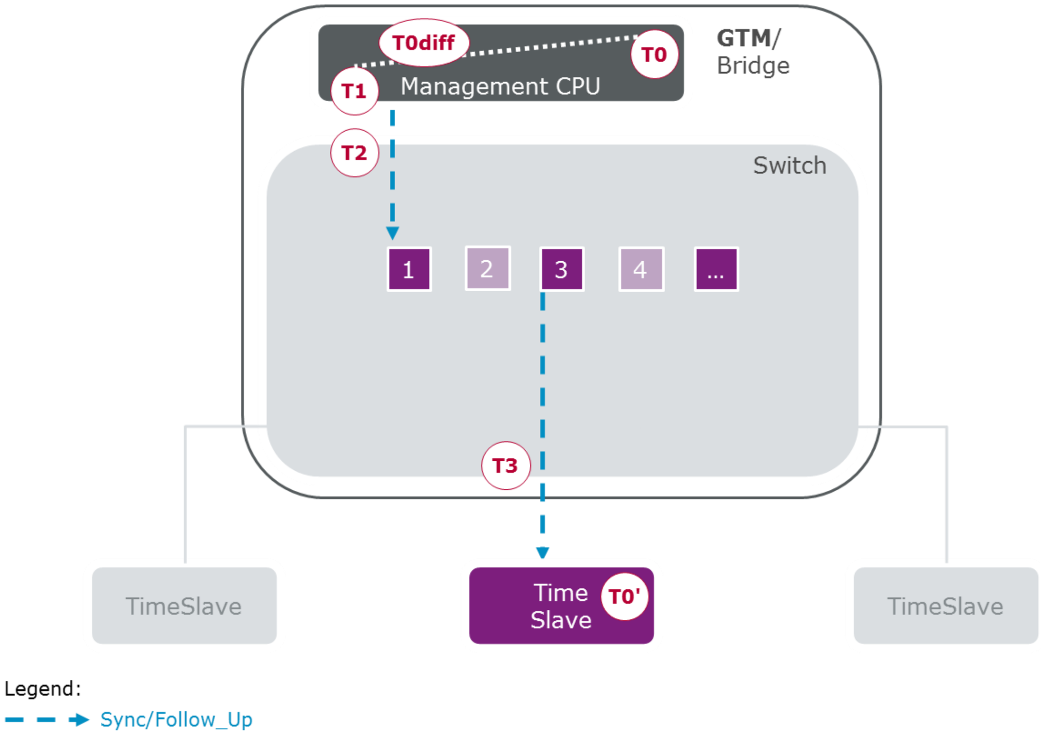


Figure 11: `Sync/Follow_Up` message flow with Timestamping points for `Sync` for Time Aware Bridge with GTM as Management CPU

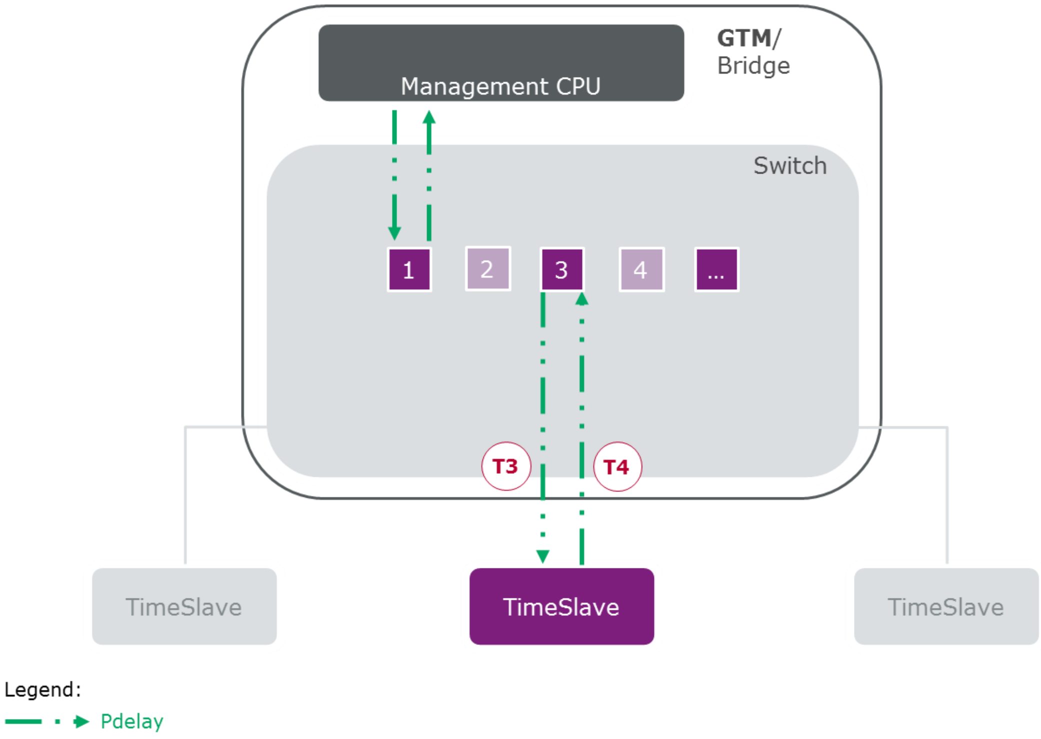


Figure 12: `Pdelay` message flow with Timestamping points for Time Aware Bridge with GTM as Management CPU

⌋(SRS_StbM_20048, SRS_StbM_20059)

Note: The picture in Figure 11 and Figure 12 shows an example Port selection as simplification.

[SWS_EthTSyn_00058]⌈
Time measurement with Switches supporting the use case “Time Aware Bridge with GTM as Management CPU” considers the inner Switch delay by a modification of the `correctionField` as well as `Pdelay` timestamping for `requestReceiptTimestamp` and `responseOriginTimestamp` like shown in Figure 13.

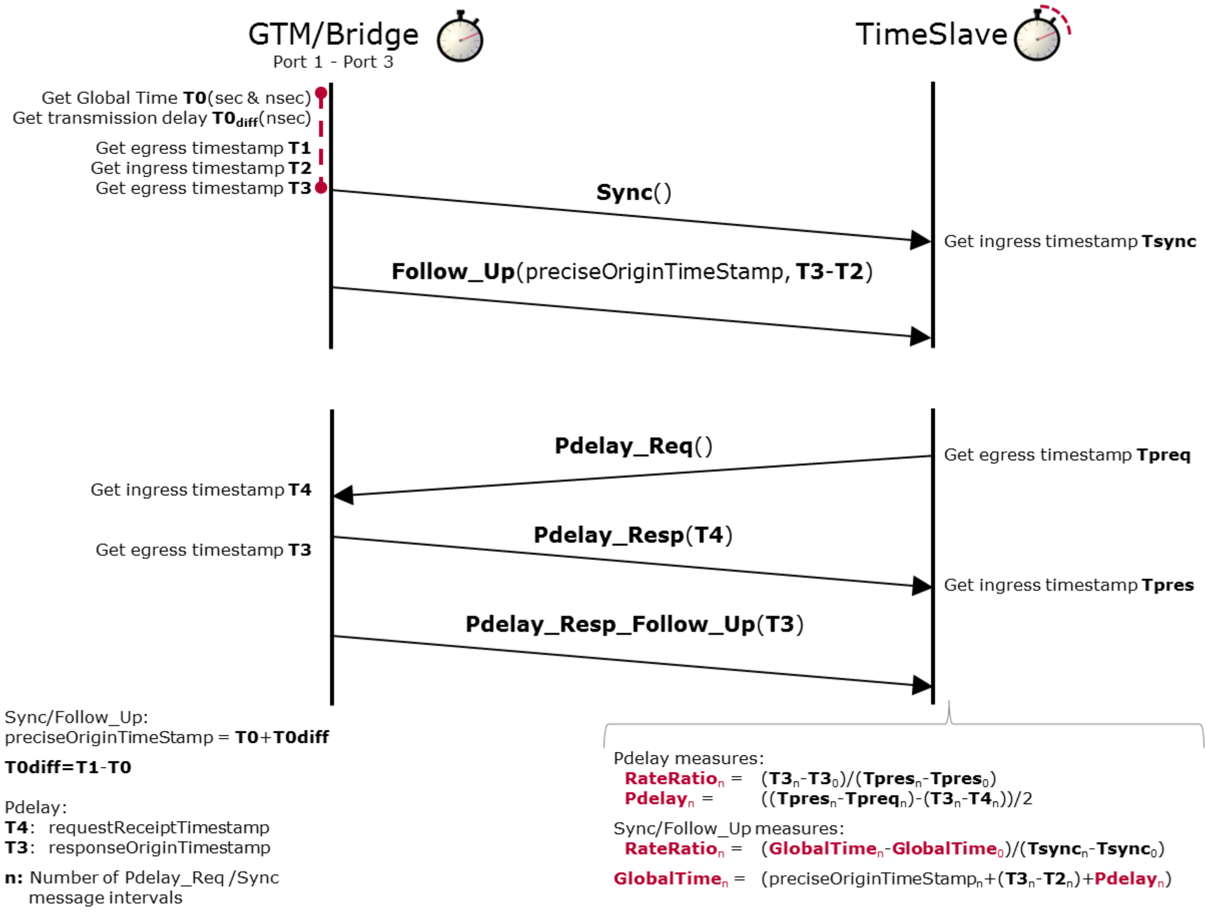

GTM/Bridge
Port 1 - Port 3
TimeSlave
Get Global Time T0(sec & nsec)
Get transmission delay T0diff(nsec)
Get egress timestamp T1
Get ingress timestamp T2
Get egress timestamp T3
Sync()
Get ingress timestamp Tsync
Follow_Up(preciseOriginTimeStamp, T3-T2)
Pdelay_Req()
Get egress timestamp Tpreq
Get ingress timestamp T4
Pdelay_Resp(T4)
Get egress timestamp T3
Get ingress timestamp Tpres
Pdelay_Resp_Follow_Up(T3)
Sync/Follow_Up:
preciseOriginTimeStamp = T0+T0diff
T0diff=T1-T0
Pdelay:
T4: requestReceiptTimestamp
T3: responseOriginTimestamp
n: Number of Pdelay_Req /Sync message intervals
Pdelay measures:
RateRatio_n = (T3_n-T3_0)/(Tpres_n-Tpres_0)
Pdelay_n = ((Tpres_n-Tpreq_n)-(T3_n-T4_n))/2
Sync/Follow_Up measures:
RateRatio_n = (GlobalTime_n-GlobalTime_0)/(Tsync_n-Tsync_0)
GlobalTime_n = (preciseOriginTimeStamp_n+(T3_n-T2_n)+Pdelay_n)

Figure 13: Timestamping sequence for Time Aware Bridge with GTM as Management CPU

⌋(SRS_StbM_20048, SRS_StbM_20059)

Note: The calculation in Figure 13 shows an example Port selection as simplification.

[SWS_EthTSyn_00166]⌈
If `EthTSynGlobalTimeUplinkToTxSwitchResidenceTime` (**ECUC_EthTSyn_00061 :**) is set to 0, EthTSyn shall ignore this parameter and measure the inner delay of the Switch egress Ethernet path (Uplink to Tx Residence Time (**T3** – **T2**)) by using always the ingress (**T2**) and egress (**T3**) timestamp as given in Figure 13.
⌋(SRS_StbM_20048, SRS_StbM_20059)

[SWS_EthTSyn_00167]⌈
If EthTSynGlobalTimeUplinkToTxSwitchResidenceTime (**ECUC_EthTSyn_00061 :**) is greater than 0, EthTSyn shall use this parameter as value for the inner delay of the Switch egress Ethernet path (Uplink to Tx Residence Time (**T3** – **T2**)) instead of using the measurement method described in **[SWS_EthTSyn_00166]**.
⌋(SRS_StbM_20048, SRS_StbM_20059)

7.8.3 Use case "Time Aware Bridge with GTM not as Management CPU"

[SWS_EthTSyn_00059]⌈
Time measurement with Switches supporting the use case "Time Aware Bridge with GTM not as Management CPU" following the given timestamping points like shown in Figure 14 and Figure 15.

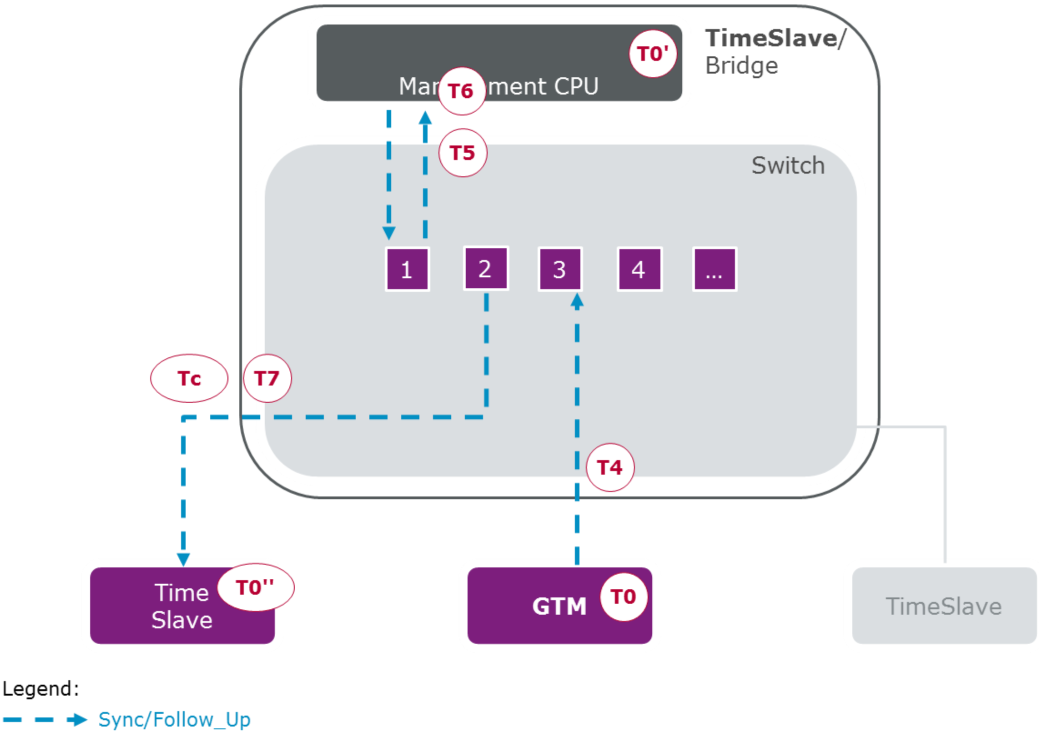


Figure 14: `Sync/Follow_Up` message flow with Timestamping points for `Sync` for Time Aware Bridge with GTM not as Management CPU

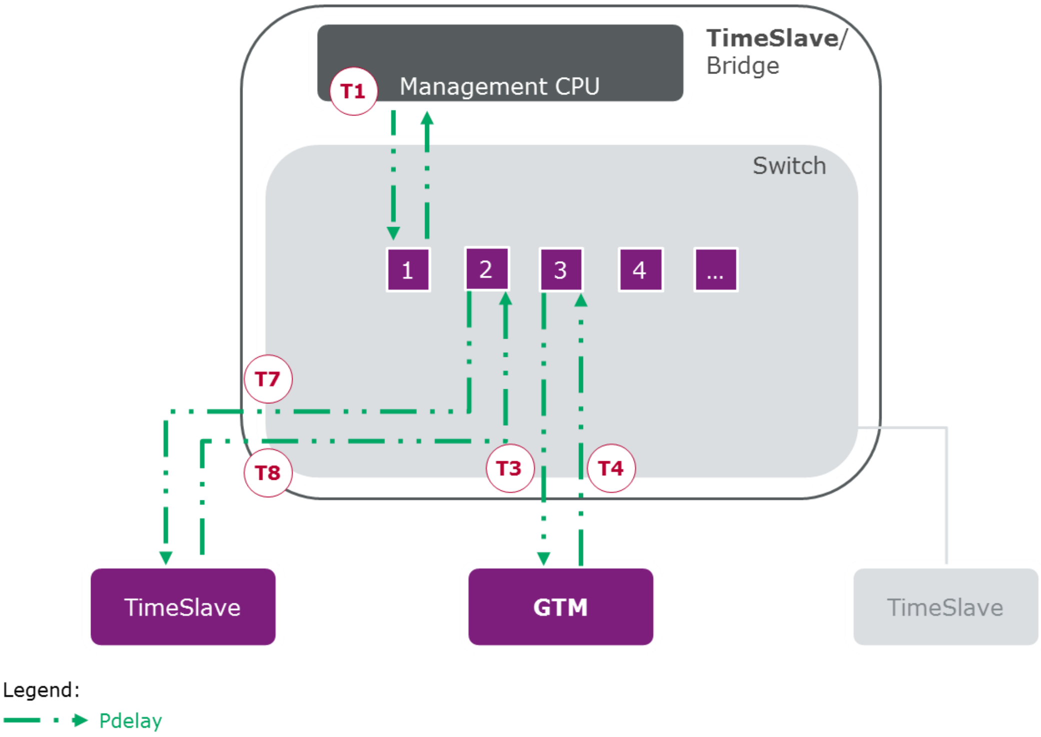


Figure 15: `Pdelay` message flow with Timestamping points for Time Aware Bridge with GTM not as Management CPU

⌋(SRS_StbM_20048, SRS_StbM_20059)

Note: The pictures in Figure 14 and Figure 15 show an example Port selection as simplification.

[SWS_EthTSyn_00060]⌈
Time measurement with Switches supporting the use case “Time Aware Bridge with GTM not as Management CPU” considers the inner Switch delay by a modification of the `correctionField` as well as `Pdelay` timestamping for `requestReceiptTimestamp` and `responseOriginTimestamp` like shown in Figure 16.
If the `Follow_Up` message contains an AUTOSAR *TLV*, which contains a *Sub-TLV*: Time Secured it shall be checked, if the element `CRC_Time_Flags` contains BitMask `0x04` (i.e., the content of `correctionField` is *CRC* protected).
If this bit is set then the validation of the `CRC_Time_1` element shall be done as follows:

- The *CRC* Validation shall be done as specified in section 7.7.2.3.
- The data elements used for the calculation and thus validation of the *CRC* shall be applied with the following order:
    1. the value of `CRC_Time_Flags`
    2. the messageLength inside the `Follow_Up` Message Header, if the element `CRC_Time_Flags` contains BitMask `0x01`
    3. the `correctionField` inside the `Follow_Up` Message Header
    4. the `sequenceId` inside the `Follow_Up` Message Header, if the element `CRC_Time_Flags` contains BitMask `0x10`
    5. the `DataID` (refer to **[SWS_EthTSyn_00112]**)

If the validation fails, the `Follow_Up` message shall be dropped instead of being forwarded.

If the validation is successful, the `correctionField` shall be modified and the element `CRC_Time_1` inside the *Sub-TLV*: Time Secured shall be calculated due to the content of the `CRC_Time_Flags` element acc. to the section below the table in **[SWS_EthTSyn_00100]**.

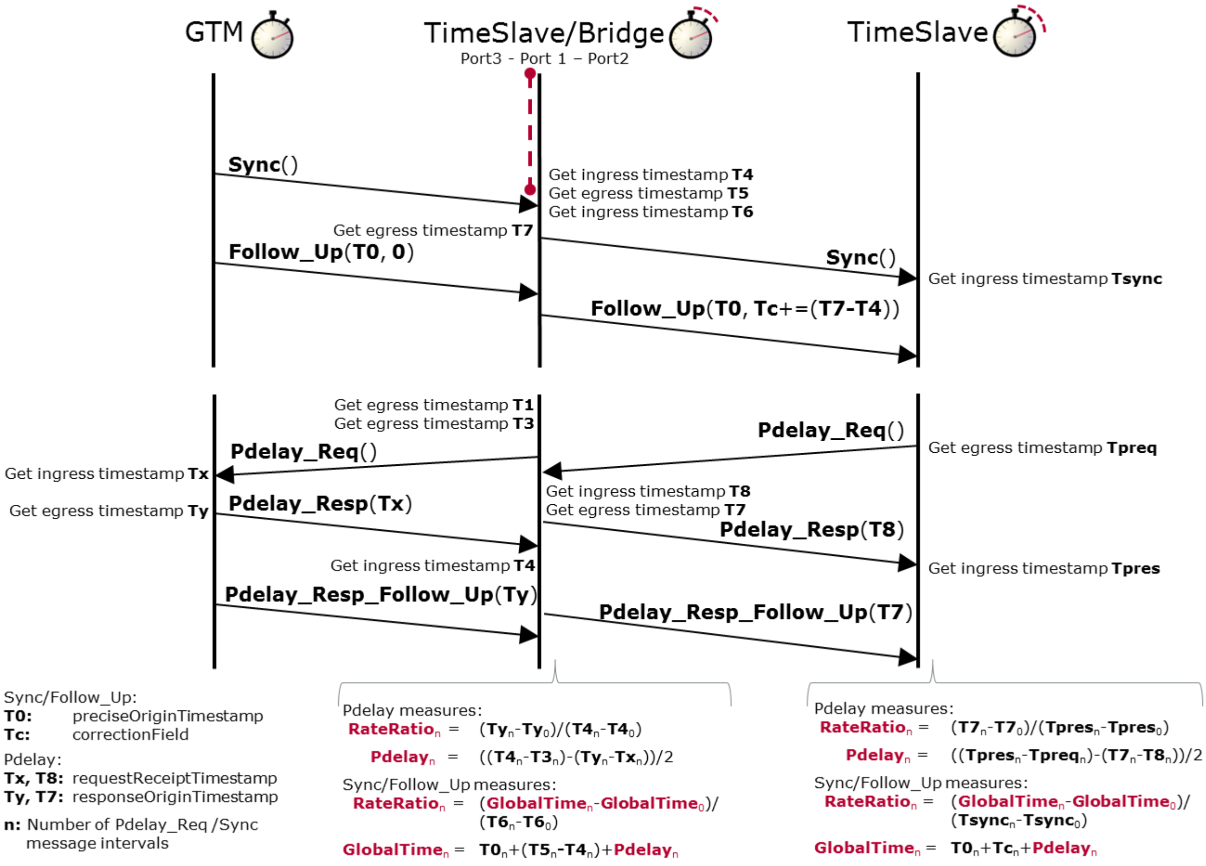

GTM
TimeSlave/Bridge
Port3 - Port 1 – Port2
TimeSlave
Sync()
Get ingress timestamp T4
Get egress timestamp T5
Get ingress timestamp T6
Get egress timestamp T7
Follow_Up(T0, 0)
Sync()
Get ingress timestamp Tsync
Follow_Up(T0, Tc+=(T7-T4))
Get egress timestamp T1
Get egress timestamp T3
Pdelay_Req()
Get egress timestamp Tpreq
Pdelay_Req()
Get ingress timestamp Tx
Get egress timestamp Ty
Pdelay_Resp(Tx)
Get ingress timestamp T8
Get egress timestamp T7
Pdelay_Resp(T8)
Get ingress timestamp Tpres
Get ingress timestamp T4
Pdelay_Resp_Follow_Up(Ty)
Pdelay_Resp_Follow_Up(T7)
Sync/Follow_Up:
T0: preciseOriginTimestamp
Tc: correctionField
Pdelay:
Tx, T8: requestReceiptTimestamp
Ty, T7: responseOriginTimestamp
n: Number of Pdelay_Req /Sync message intervals
Pdelay measures:
RateRatio_n = (Ty_n-Ty_0)/(T4_n-T4_0)
Pdelay_n = ((T4_n-T3_n)-(Ty_n-Tx_n))/2
Sync/Follow_Up measures:
RateRatio_n = (GlobalTime_n-GlobalTime_0)/(T6_n-T6_0)
GlobalTime_n = T0_n+(T5_n-T4_n)+Pdelay_n
Pdelay measures:
RateRatio_n = (T7_n-T7_0)/(Tpres_n-Tpres_0)
Pdelay_n = ((Tpres_n-Tpreq_n)-(T7_n-T8_n))/2
Sync/Follow_Up measures:
RateRatio_n = (GlobalTime_n-GlobalTime_0)/(Tsync_n-Tsync_0)
GlobalTime_n = T0_n+Tc_n+Pdelay_n

Figure 16: Timestamping sequence for Time Aware Bridge with GTM not as Management CPU

⌋(SRS_StbM_20048, SRS_StbM_20059)

Note: The calculation in Figure 16 shows an example Port selection as simplification.

[SWS_EthTSyn_00168]⌈
If `EthTSynGlobalTimeRxToUplinkSwitchResidenceTime` (**ECUC_EthTSyn_00060 :**) is set to 0, EthTSyn shall ignore this parameter and measure the inner delay of the Switch ingress Ethernet path (Rx to Uplink Residence Time (**T5** – **T4**)) by using always the ingress (**T4**) and egress (**T5**) timestamp as given in Figure 16.
⌋(SRS_StbM_20048, SRS_StbM_20059)

[SWS_EthTSyn_00171]⌈
If EthTSynGlobalTimeRxToUplinkSwitchResidenceTime (**ECUC_EthTSyn_00060 :**) is greater than 0, EthTSyn shall use this parameter as value for the inner delay of the Switch ingress Ethernet path (Rx to Uplink Residence Time (**T5** – **T4**)) instead of using the measurement method described in **[SWS_EthTSyn_00168]**.
⌋(SRS_StbM_20048, SRS_StbM_20059)

[SWS_EthTSyn_00169]⌈
If EthTSynGlobalTimeRxToUplinkSwitchResidenceTime (**ECUC_EthTSyn_00060 :**) and EthTSynGlobalTimeUplinkToTxSwitchResidenceTime (**ECUC_EthTSyn_00061 :**) are set to 0, EthTSyn shall ignore both parameter and measure the inner delay of the Switch ingress and egress Ethernet path (Rx to Uplink and Uplink to Tx Residence Time (**T7** – **T4**)) by using always the ingress (**T4**) and egress (**T7**) timestamp as given in Figure 16.
⌋(SRS_StbM_20048, SRS_StbM_20059)

[SWS_EthTSyn_00170]⌈
If EthTSynGlobalTimeRxToUplinkSwitchResidenceTime (**ECUC_EthTSyn_00060 :**) and EthTSynGlobalTimeUplinkToTxSwitchResidenceTime (**ECUC_EthTSyn_00061 :**) are greater than 0, EthTSyn shall use the sum of both parameter for the value of the inner delay of the Switch ingress and egress Ethernet path (Rx to Uplink and Uplink to Tx Residence Time (**T7** – **T4**)) instead of using the measurement method described in **[SWS_EthTSyn_00169]**.
⌋(SRS_StbM_20048, SRS_StbM_20059)

Note: A separate Uplink to Tx Residence Time (**T7** – $T_{UplinkMmCpu}$) replacement by using EthTSynGlobalTimeUplinkToTxSwitchResidenceTime might be also possible, but is not considered by the scenario given in Figure 16.

7.9 Error Classification

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

[SWS_EthTSyn_00029]⌈
On errors and exceptions, the EthTSyn module shall not modify its current module state but shall simply report the error event.
⌋(SRS_StbM_20051, SRS_BSW_00323)

7.9.1 Development Errors

The detection of development errors is configurable (refer (**ECUC_EthTSyn_00002 :**)).

[SWS_EthTSyn_00030]⌈
EthTSyn shall use following development errors:

Type or error	***Related error code***	***Value [hex]***
API service used in un-initialized state	ETHTSYN_E_UNINIT	0x20
EthTSyn initialization failed	ETHTSYN_E_INIT_FAILED	0x21
API called with invalid controller index	ETHTSYN_E_CTRL_IDX	0x22
API called with invalid pointer	ETHTSYN_E_PARAM_POINTER	0x23
API called with invalid parameter	ETHTSYN_E_PARAM	0x24

⌋(SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00323)

7.9.2 Runtime Errors

[SWS_EthTSyn_00144]⌈
EthTSyn shall use following runtime errors:

Type or error	***Related error code***	***Value [hex]***
Time Master conflict	ETHTSYN_E_TMCONFLICT	0x01
Time Slave conflict	ETHTSYN_E_TSCONFLICT	0x02

⌋(SRS_BSW_00385)

7.9.3 Transient Faults

No Transient Faults defined.

7.9.4 Production Errors

No Production Errors defined.

7.9.5 Extended Production Errors

No Extended Production Errors defined.

8 API specification

8.1 API

8.1.1 Imported types

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

[SWS_EthTSyn_00031] ⌈

Module	*Imported Type*
ComStack_Types	BufReq_ReturnType
EthSwt	EthSwt_MgmtInfoType
Eth_GeneralTypes	EthTrcv_LinkStateType
	Eth_BufIdxType
	Eth_DataType
	Eth_FrameType
	Eth_TimeStampQualType
	Eth_TimeStampType
StbM	StbM_MeasurementType
	StbM_SynchronizedTimeBaseType
	StbM_TimeBaseStatusType
	StbM_TimeStampRawType
	StbM_TimeStampType
	StbM_UserDataType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋ (SRS_StbM_20048, SRS_StbM_20059)

8.1.2 Type definitions

8.1.2.1 EthTSyn_ConfigType

[SWS_EthTSyn_00032] ⌈

Name:	EthTSyn_ConfigType		
Type:	Structure		
Element:	void	implementation specific	--
Description:	This is the base type for the configuration of the Global Time Synchronization over Ethernet. A pointer to an instance of this structure will be used in the initialization of the Global Time Synchronization over Ethernet. The content of this structure is defined in chapter 10 Configuration specification.		

⌋ (SRS_StbM_20048)

8.1.2.2 EthTSyn_TransmissionModeType

[SWS_EthTSyn_00033] ⌈

Name:	EthTSyn_TransmissionModeType		
Type:	Enumeration		
Range:	ETHTSYN_TX_OFF	0x00	Transmission Disabled
	ETHTSYN_TX_ON	0x01	Transmission Enabled
Description:	Handles the enabling and disabling of the transmission mode		

⌋ (SRS_StbM_20048)

8.1.2.3 EthTSyn_SyncStateType

[SWS_EthTSyn_00034] ⌈

Name:	EthTSyn_SyncStateType		
Type:	Enumeration		
Range:	ETHTSYN_SYNC	0x00	Ethernet time synchronous
	ETHTSYN_UNSYNC	0x01	Ethernet not time synchronous
	ETHTSYN_UNCERTAIN	0x02	Ethernet Sync state uncertain
	ETHTSYN_NEVERSYNC	0x03	No Sync message received between EthTSyn_Init() and current requested state.
Description:	Depending on the HW, quality information regarding the evaluated Sync state might be supported. If not supported, the value shall be always ETHTSYN_SYNC. For ETHTSYN_UNSYNC and ETHTSYN_UNCERTAIN values, the upper layer shall discard the time synchronous information. Within this enumeration, ETHTSYN_NEVERSYNC is having a higher priority than ETHTSYN_UNSYNC.		

⌋ (SRS_StbM_20048)

8.1.3 Function definitions

8.1.3.1 EthTSyn_Init

[SWS_EthTSyn_00035] ⌈

Service name:	EthTSyn_Init	

Syntax:	`void EthTSyn_Init(`
	`    const EthTSyn_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 Ethernet.	

⌋ (SRS_StbM_20048)

See section 7.2 for details.

8.1.3.2 EthTSyn_GetVersionInfo

[SWS_EthTSyn_00036] ⌈

Service name:	EthTSyn_GetVersionInfo	

Syntax:	`void EthTSyn_GetVersionInfo(`
	`    Std_VersionInfoType* versioninfo`
	`)`	

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

⌋ (SRS_StbM_20048)

8.1.3.3 EthTSyn_SetTransmissionMode

[SWS_EthTSyn_00039] ⌈

Service name:	EthTSyn_SetTransmissionMode	

Syntax:	`void EthTSyn_SetTransmissionMode(`
	`    uint8 CtrlIdx,`
	`    EthTSyn_TransmissionModeType Mode`
	`)`	

Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller
	Mode	ETHTSYN_TX_OFF ETHTSYN_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 EthTSyn.	

⌋ (SRS_StbM_20048)

[SWS_EthTSyn_00172]⌈
The function `EthTSyn_SetTransmissionMode()` shall inform the DET, if development error detection is enabled (`EthTSynDevErrorDetect` (**ECUC_EthTSyn_00002 :**) is set to `TRUE`) and if function call has failed because of the following reasons:

- `CtrlIdx` is invalid (`ETHTSYN_E_CTRL_IDX`)
- `Mode` is invalid (`ETHTSYN_E_PARAM`)

⌋(SRS_BSW_00323, SRS_BSW_00337)

8.1.4 Call-back notifications

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

8.1.4.1 EthTSyn_RxIndication

[SWS_EthTSyn_00040] ⌈

Service name:	EthTSyn_RxIndication	

Syntax:	`void EthTSyn_RxIndication(`
	`    uint8 CtrlIdx,`
	`    Eth_FrameType FrameType,`
	`    boolean IsBroadcast,`
	`    const uint8* PhysAddrPtr,`
	`    const uint8* DataPtr,`
	`    uint16 LenByte`
	`)`	

Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller
	FrameType	frame type of received Ethernet frame
	IsBroadcast	parameter to indicate a broadcast frame
	PhysAddrPtr	pointer to Physical source address (MAC address in network byte order) of received Ethernet frame
	DataPtr	Pointer to payload of the received Ethernet frame (i.e. Ethernet header is not provided).
	LenByte	Length of received data.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	By this API service the EthTSyn gets an indication and the data of a received frame.	

⌋ (SRS_StbM_20048)

[SWS_EthTSyn_00041]⌈
The callback function `EthTSyn_RxIndication()` shall inform the DET, if development error detection is enabled (`EthTSynDevErrorDetect` (**ECUC_EthTSyn_00002 :**) is set to `TRUE`) and if the function call has failed because of the following reasons:

- `CtrlIdx` is invalid (`ETHTSYN_E_CTRL_IDX`)
- `DataPtr` or `PhysAddrPtr` is invalid (`ETHTSYN_E_PARAM_POINTER`)

⌋(SRS_BSW_00337, SRS_BSW_00323)

8.1.4.2 EthTSyn_TxConfirmation

[SWS_EthTSyn_00042] ⌈

Service name:	EthTSyn_TxConfirmation	

Syntax:	`void EthTSyn_TxConfirmation(`
	`    uint8 CtrlIdx,`
	`    Eth_BufIdxType BufIdx`
	`)`	

Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Dont care	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	BufIdx	Index of the buffer resource

Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Confirms the transmission of an Ethernet frame

⌋ (SRS_StbM_20048)

[SWS_EthTSyn_00175]⌈
The function `EthTSyn_TxConfirmation()` shall inform the DET, if development error detection is enabled (`EthTSynDevErrorDetect` (**ECUC_EthTSyn_00002 :**) is set to `TRUE`) and if function call has failed because of the following reasons:

- `CtrlIdx` is invalid (`ETHTSYN_E_CTRL_IDX`)

⌋(SRS_BSW_00323, SRS_BSW_00337)

8.1.4.3 EthTSyn_TrcvLinkStateChg

[SWS_EthTSyn_00043] ⌈

Service name:	EthTSyn_TrcvLinkStateChg	

Syntax:	`Std_ReturnType EthTSyn_TrcvLinkStateChg(`
	`    uint8 CtrlIdx,`
	`    EthTrcv_LinkStateType TrcvLinkState`
	`)`	

Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller
	TrcvLinkState	ETHTRCV_LINK_STATE_DOWN ETHTRCV_LINK_STATE_ACTIVE
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description:	Allows resetting state machine in case of unexpected Link loss to avoid inconsistent Sync and Follow_Up sequences	

⌋ (SRS_StbM_20048)

[SWS_EthTSyn_00174]⌈
The function `EthTSyn_TrcvLinkStateChg()` shall inform the DET, if development error detection is enabled (`EthTSynDevErrorDetect` (**ECUC_EthTSyn_00002 :**) is set to `TRUE`) and if function call has failed because of the following reasons:

- `CtrlIdx` is invalid (`ETHTSYN_E_CTRL_IDX`)

⌋(SRS_BSW_00323, SRS_BSW_00337)

8.1.4.4 EthTSyn_SwitchMgmtInfoIndication

[SWS_EthTSyn_91000] ⌈

Service name:	EthTSyn_SwitchMgmtInfoIndication

Syntax:	`void EthTSyn_SwitchMgmtInfoIndication(`
	`    uint8 CtrlIdx,`
	`    uint8* DataPtr,`

	`    EthSwt_MgmtInfoType* MgmtInfoPtr`
	`)`	

Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	DataPtr	Data pointer where the management information belongs
	MgmtInfoPtr	Management information if not NULL
Parameters (inout):	None	
Parameters (out):	None	
Return value:	void	--
Description:	Ingress Switch management info indication redirected call to upper layers who registered for the call.	

⌋ (SRS_StbM_20048, SRS_StbM_20059)

[SWS_EthTSyn_00152]⌈
The callback function `EthTSyn_SwitchMgmtInfoIndication()` shall inform the DET, if development error detection is enabled (`EthTSynDevErrorDetect` (**ECUC_EthTSyn_00002 :**) is set to `TRUE`) and if the function call has failed because of the following reasons:

- `CtrlIdx` is invalid (`ETHTSYN_E_CTRL_IDX`)
- `DataPtr` is invalid (`ETHTSYN_E_PARAM_POINTER`)

⌋(SRS_BSW_00337)

8.1.4.5 EthTSyn_SwitchEgressTimeStampIndication

[SWS_EthTSyn_91001] ⌈

Service name:	EthTSyn_SwitchEgressTimeStampIndication	

Syntax:	`void EthTSyn_SwitchEgressTimeStampIndication(`
	`    uint8 CtrlIdx,`
	`    uint8* DataPtr,`
	`    EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`    Eth_TimeStampType* TimeStampPtr`
	`)`	

Service ID[hex]:	0x0c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	DataPtr	Data pointer
	MgmtInfoPtr	Management information
	TimeStampPtr	Current timestamp
Parameters (inout):	None	
Parameters (out):	None	
Return value:	void	--
Description:	Delivers to upper layers an egress timestamp value from the Switch where MgmtInfo refers. If the HW resolution is lower than the Eth_TimeStampType resolution resp. range, than the remaining bits will be filled with 0.	

⌋ (SRS_StbM_20048, SRS_StbM_20059)

[SWS_EthTSyn_00177]⌈
The callback function `EthTSyn_SwitchEgressTimeStampIndication()` shall inform the DET, if development error detection is enabled (`EthTSynDevErrorDetect` (**ECUC_EthTSyn_00002 :**) is set to `TRUE`) and if the function call has failed because of the following reasons:

- `CtrlIdx` is invalid (`ETHTSYN_E_CTRL_IDX`)
- `DataPtr`, `MgmtInfoPtr` or `TimeStampPtr` is invalid (`ETHTSYN_E_PARAM_POINTER`)

⌋(SRS_BSW_00337, SRS_BSW_00323)

8.1.4.6 EthTSyn_SwitchIngressTimeStampIndication

[SWS_EthTSyn_91002] ⌈

Service name:	EthTSyn_SwitchIngressTimeStampIndication	

Syntax:	`void EthTSyn_SwitchIngressTimeStampIndication(`
	`    uint8 CtrlIdx,`
	`    uint8* DataPtr,`
	`    EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`    Eth_TimeStampType* TimeStampPtr`
	`)`	

Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	DataPtr	Data pointer
	MgmtInfoPtr	Management information
	TimeStampPtr	Current timestamp
Parameters (inout):	None	
Parameters (out):	None	
Return value:	void	--
Description:	Delivers to upper layers an ingress timestamp value from the Switch where MgmtInfo refers. If the HW resolution is lower than the Eth_TimeStampType resolution resp. range, than the remaining bits will be filled with 0.	

⌋ (SRS_StbM_20048, SRS_StbM_20059)

[SWS_EthTSyn_00178]⌈
The callback function `EthTSyn_SwitchIngressTimeStampIndication()` shall inform the DET, if development error detection is enabled (`EthTSynDevErrorDetect` (**ECUC_EthTSyn_00002 :**) is set to `TRUE`) and if the function call has failed because of the following reasons:

- `CtrlIdx` is invalid (`ETHTSYN_E_CTRL_IDX`)
- `DataPtr`, `MgmtInfoPtr` or `TimeStampPtr` is invalid (`ETHTSYN_E_PARAM_POINTER`)

⌋(SRS_BSW_00337, SRS_BSW_00323)

8.1.5 Scheduled functions

The Basic Software Scheduler directly calls these functions. The following functions shall have no return value and no parameters. All functions shall be non-reentrant.

8.1.5.1 EthTSyn_MainFunction

[SWS_EthTSyn_00044] ⌈

Service name:	EthTSyn_MainFunction

Syntax:	`void EthTSyn_MainFunction(`
	`    void`
	`)`

Service ID[hex]:	0x09
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Main function for cyclic call / resp. Sync, Follow_Up and Pdelay_Req transmissions

⌋ (SRS_StbM_20048)

[SWS_EthTSyn_00045]⌈
The frequency of invocations of `EthTSyn_MainFunction()` is determined by the configuration parameter `EthTSynMainFunctionPeriod` (**ECUC_EthTSyn_00012 :**).
⌋(SRS_StbM_20048)

8.1.6 Expected Interfaces

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

8.1.6.1 Mandatory Interfaces

There are no mandatory interfaces defined.

8.1.6.2 Optional Interfaces

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

[SWS_EthTSyn_00047] ⌈

API function	***Description***
Crc_CalculateCRC8H2F	This service makes a CRC8 calculation with the Polynomial 0x2F on Crc_Length
Det_ReportError	Service to report development errors.
Det_ReportRuntimeError	Service to report runtime errors. If a callout has been configured then this callout shall be called.
EthIf_EnableEgressTimeStamp	Activates egress time stamping on a dedicated message object. Some HW does store once the egress time stamp marker and some HW needs it always before transmission. There will be no

	"disable" functionality, due to the fact, that the message type is always "time stamped" by network design.
EthIf_GetCurrentTime	Returns a time value out of the HW registers according to the capability of the HW. Is the HW resolution is lower than the Eth_TimeStampType resolution resp. range, the remaining bits will be filled with 0.
EthIf_GetEgressTimeStamp	Reads back the egress time stamp on a dedicated message object. It must be called within the TxConfirmation() function.
EthIf_GetIngressTimeStamp	Reads back the ingress time stamp on a dedicated message object. It must be called within the RxIndication() function.
EthIf_ProvideTxBuffer	Provides access to a transmit buffer of the specified Ethernet controller.
EthIf_SetSwitchMgmtInfo	Provides additional management information along to an Ethernet frame that requires special treatment within the Switch. It has to be called between EthIf_ProvideTxBuffer() and EthIf_Transmit() of the related frame.
EthIf_SwitchEnableTimeStamping	Activates egress time stamping on a dedicated message object, addressed by CtrlIdx and BufIdx.
EthIf_Transmit	Triggers transmission of a previously filled transmit buffer
StbM_BusSetGlobalTime	Allows the Time Base Provider Modules to forward a new Global Time value to the StbM, which has been received from a bus.
StbM_GetCurrentTime	Returns a time value (Local Time Base derived from Global Time Base) in standard format.
StbM_GetCurrentTimeDiff	Returns time difference of the nanoseconds part of the Virtual Local Time of the referenced Time Base minus the time given by the parameter givenTimeStamp.
StbM_GetCurrentTimeRaw	Returns nanosecond part of the Virtual Local Time of the referenced Time Base.
StbM_GetOffset	Allows the Timesync Modules to get the current Offset Time and User Data.
StbM_GetTimeBaseStatus	Returns the detailed status of the Time Base. For Offset Time Bases the status of the Offset Time Base itself and the status of the underlying Synchronized Time Base is returned.
StbM_GetTimeBaseUpdateCounter	Allows the Timesync Modules to detect, whether a Time Base should be transmitted immediately in the subsequent <Bus>TSyn_MainFunction() cycle.

⌋ (SRS_StbM_20048, SRS_StbM_20059)

9 Sequence diagrams

Note: Please consider, that all sequence diagrams use case specific (Ethernet controller w/o Switch).

9.1 EthIf_EnableEgressTimeStamp

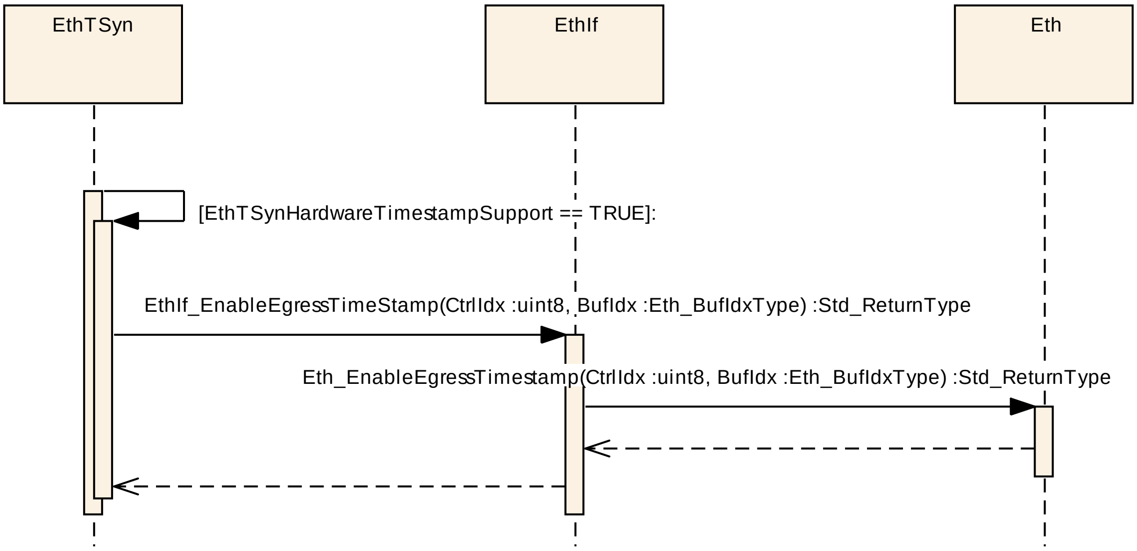
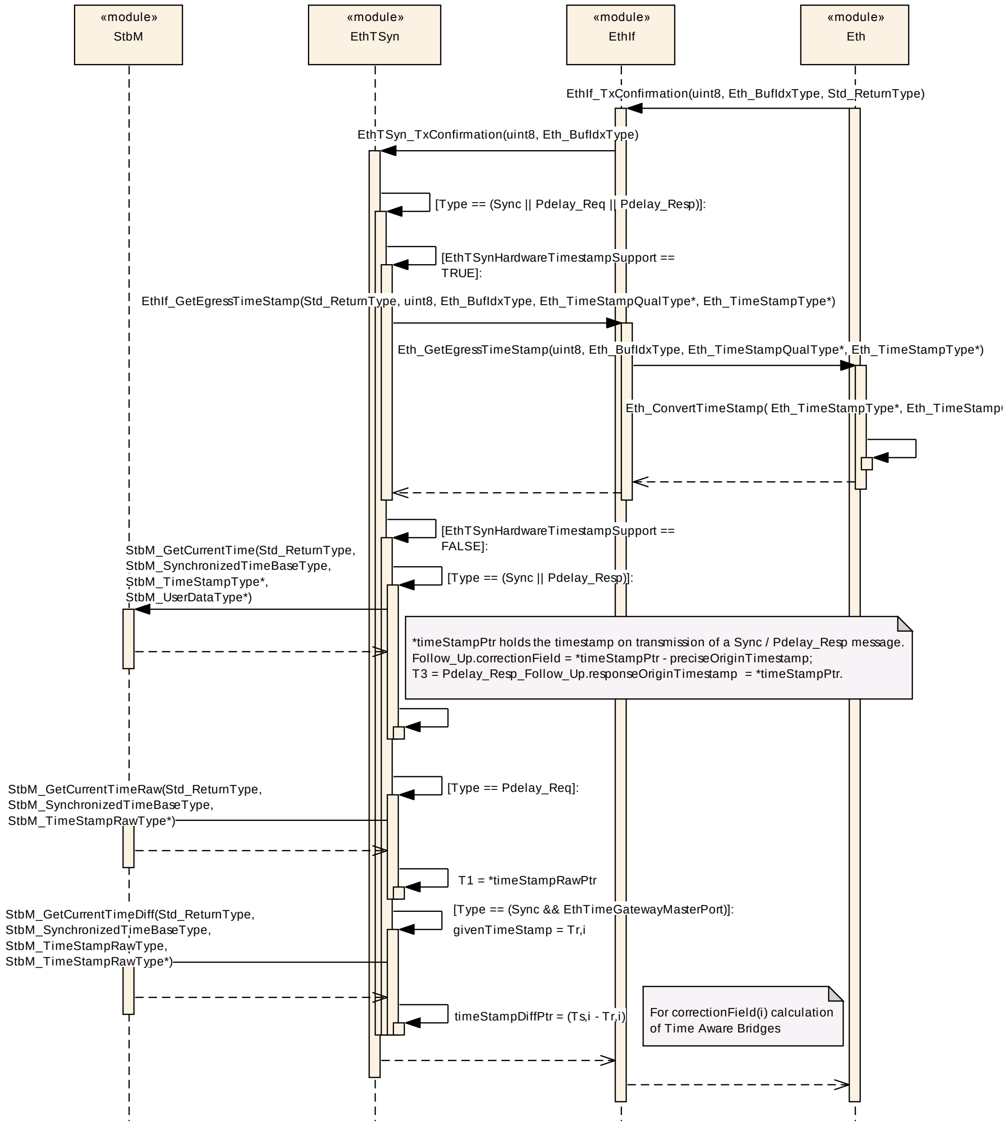

EthTSyn
EthIf
Eth
[EthTSynHardwareTimestampSupport == TRUE]:
EthIf_EnableEgressTimeStamp(CtrlIdx :uint8, BufIdx :Eth_BufIdxType) :Std_ReturnType
Eth_EnableEgressTimestamp(CtrlIdx :uint8, BufIdx :Eth_BufIdxType) :Std_ReturnType

Figure 17: EthIf_EnableEgressTimeStamp

9.2 Time Master `Sync/Follow_Up` and `Pdelay` – Tx


«module»
StbM
«module»
EthTSyn
«module»
EthIf
«module»
Eth
EthIf_TxConfirmation(uint8, Eth_BufIdxType, Std_ReturnType)
EthTSyn_TxConfirmation(uint8, Eth_BufIdxType)
[Type == (Sync || Pdelay_Req || Pdelay_Resp)]:
[EthTSynHardwareTimestampSupport == TRUE]:
EthIf_GetEgressTimeStamp(Std_ReturnType, uint8, Eth_BufIdxType, Eth_TimeStampQualType*, Eth_TimeStampType*)
Eth_GetEgressTimeStamp(uint8, Eth_BufIdxType, Eth_TimeStampQualType*, Eth_TimeStampType*)
Eth_ConvertTimeStamp(Eth_TimeStampType*, Eth_TimeStamp
[EthTSynHardwareTimestampSupport == FALSE]:
StbM_GetCurrentTime(Std_ReturnType, StbM_SynchronizedTimeBaseType, StbM_TimeStampType*, StbM_UserDataType*)
[Type == (Sync || Pdelay_Resp)]:
[Type == Pdelay_Req]:
StbM_GetCurrentTimeRaw(Std_ReturnType, StbM_SynchronizedTimeBaseType, StbM_TimeStampRawType*)
T1 = *timeStampRawPtr
StbM_GetCurrentTimeDiff(Std_ReturnType, StbM_SynchronizedTimeBaseType, StbM_TimeStampRawType, StbM_TimeStampRawType*)
[Type == (Sync && EthTimeGatewayMasterPort)]:
givenTimeStamp = Tr,i
timeStampDiffPtr = (Ts,i - Tr,i)

*timeStampPtr holds the timestamp on transmission of a Sync / Pdelay_Resp message.
Follow_Up.correctionField = *timeStampPtr - preciseOriginTimestamp;
T3 = Pdelay_Resp_Follow_Up.responseOriginTimestamp = *timeStampPtr.

For correctionField(i) calculation of Time Aware Bridges

Figure 18: Time Master `Sync/Follow_Up` and Pdelay – Tx

9.3 Time Slave `Sync`/`Follow_Up` and `Pdelay` – Rx

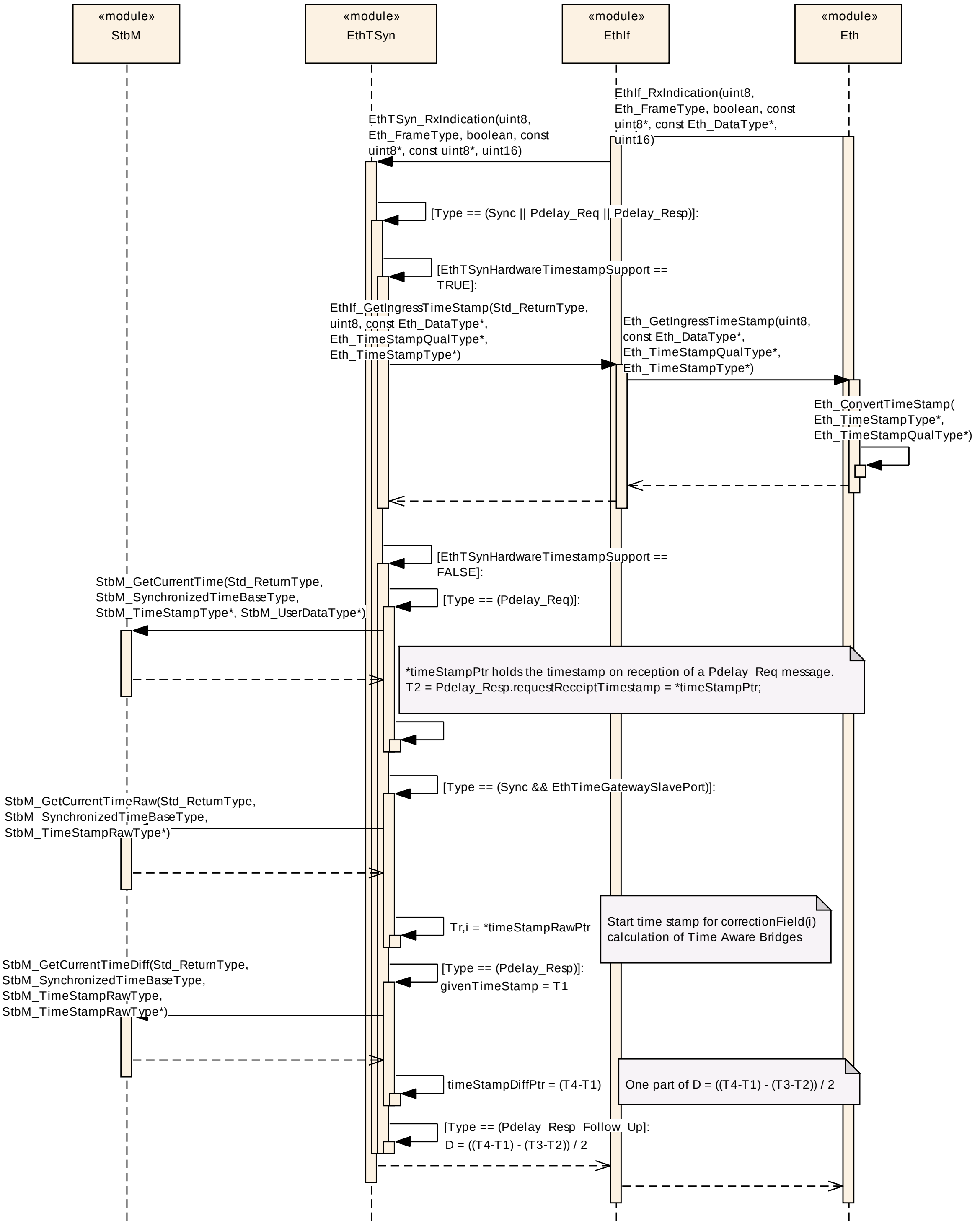

«module»
StbM
«module»
EthTSyn
«module»
EthIf
«module»
Eth
EthIf_RxIndication(uint8, Eth_FrameType, boolean, const uint8*, const Eth_DataType*, uint16)
EthTSyn_RxIndication(uint8, Eth_FrameType, boolean, const uint8*, const uint8*, uint16)
[Type == (Sync || Pdelay_Req || Pdelay_Resp)]:
[EthTSynHardwareTimestampSupport == TRUE]:
EthIf_GetIngressTimeStamp(Std_ReturnType, uint8, const Eth_DataType*, Eth_TimeStampQualType*, Eth_TimeStampType*)
Eth_GetIngressTimeStamp(uint8, const Eth_DataType*, Eth_TimeStampQualType*, Eth_TimeStampType*)
Eth_ConvertTimeStamp(Eth_TimeStampType*, Eth_TimeStampQualType*)
[EthTSynHardwareTimestampSupport == FALSE]:
[Type == (Pdelay_Req)]:
StbM_GetCurrentTime(Std_ReturnType, StbM_SynchronizedTimeBaseType, StbM_TimeStampType*, StbM_UserDataType*)
*timeStampPtr holds the timestamp on reception of a Pdelay_Req message.
T2 = Pdelay_Resp.requestReceiptTimestamp = *timeStampPtr;
[Type == (Sync && EthTimeGatewaySlavePort)]:
StbM_GetCurrentTimeRaw(Std_ReturnType, StbM_SynchronizedTimeBaseType, StbM_TimeStampRawType*)
Tr,i = *timeStampRawPtr
Start time stamp for correctionField(i) calculation of Time Aware Bridges
[Type == (Pdelay_Resp)]:
givenTimeStamp = T1
StbM_GetCurrentTimeDiff(Std_ReturnType, StbM_SynchronizedTimeBaseType, StbM_TimeStampRawType, StbM_TimeStampRawType*)
timeStampDiffPtr = (T4-T1)
One part of D = ((T4-T1) - (T3-T2)) / 2
[Type == (Pdelay_Resp_Follow_Up]:
D = ((T4-T1) - (T3-T2)) / 2

Figure 19: Time Slave `Sync`/`Follow_Up` and `Pdelay` – Rx

9.4 Time measurement with Switches

9.4.1 Time Aware Bridge with GTM as Management CPU – Tx

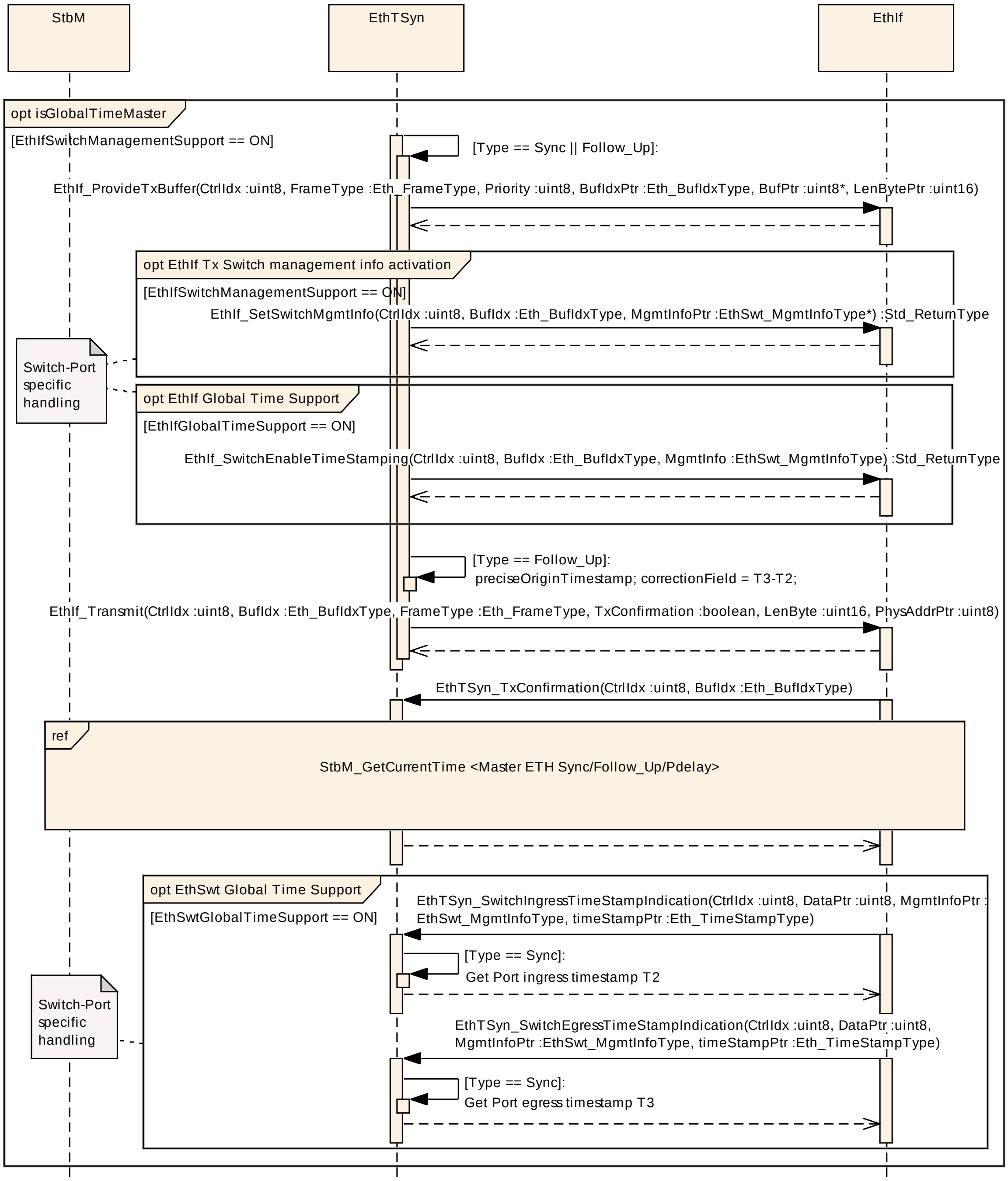

StbM
EthTSyn
EthIf
opt isGlobalTimeMaster
[EthIfSwitchManagementSupport == ON]
[Type == Sync || Follow_Up]:
EthIf_ProvideTxBuffer(CtrlIdx :uint8, FrameType :Eth_FrameType, Priority :uint8, BufIdxPtr :Eth_BufIdxType, BufPtr :uint8*, LenBytePtr :uint16)
opt EthIf Tx Switch management info activation
[EthIfSwitchManagementSupport == ON]
EthIf_SetSwitchMgmtInfo(CtrlIdx :uint8, BufIdx :Eth_BufIdxType, MgmtInfoPtr :EthSwt_MgmtInfoType*) :Std_ReturnType
Switch-Port specific handling
opt EthIf Global Time Support
[EthIfGlobalTimeSupport == ON]
EthIf_SwitchEnableTimeStamping(CtrlIdx :uint8, BufIdx :Eth_BufIdxType, MgmtInfo :EthSwt_MgmtInfoType) :Std_ReturnType
[Type == Follow_Up]:
preciseOriginTimestamp; correctionField = T3-T2;
EthIf_Transmit(CtrlIdx :uint8, BufIdx :Eth_BufIdxType, FrameType :Eth_FrameType, TxConfirmation :boolean, LenByte :uint16, PhysAddrPtr :uint8)
EthTSyn_TxConfirmation(CtrlIdx :uint8, BufIdx :Eth_BufIdxType)
ref
StbM_GetCurrentTime <Master ETH Sync/Follow_Up/Pdelay>
opt EthSwt Global Time Support
[EthSwtGlobalTimeSupport == ON]
EthTSyn_SwitchIngressTimeStampIndication(CtrlIdx :uint8, DataPtr :uint8, MgmtInfoPtr :
EthSwt_MgmtInfoType, timeStampPtr :Eth_TimeStampType)
[Type == Sync]:
Get Port ingress timestamp T2
Switch-Port specific handling
EthTSyn_SwitchEgressTimeStampIndication(CtrlIdx :uint8, DataPtr :uint8,
MgmtInfoPtr :EthSwt_MgmtInfoType, timeStampPtr :Eth_TimeStampType)
[Type == Sync]:
Get Port egress timestamp T3

Figure 20: Time Aware Bridge with GTM as Management CPU [Sync/Follow_Up Tx]

9.4.2 Time Aware Bridge without GTM as Management CPU – Tx

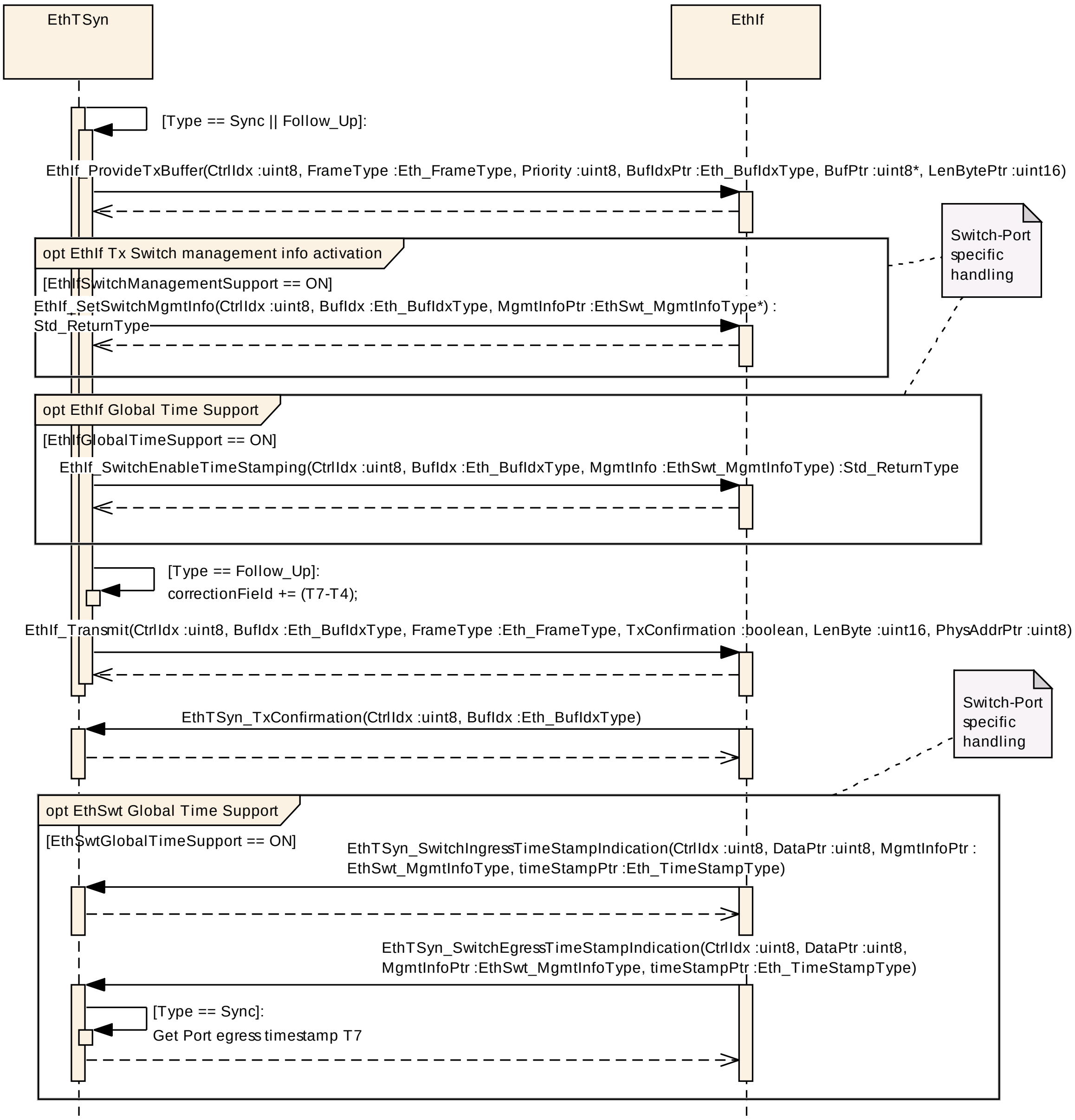

EthTSyn
EthIf
[Type == Sync || Follow_Up]:
EthIf_ProvideTxBuffer(CtrlIdx :uint8, FrameType :Eth_FrameType, Priority :uint8, BufIdxPtr :Eth_BufIdxType, BufPtr :uint8*, LenBytePtr :uint16)
Switch-Port specific handling
opt EthIf Tx Switch management info activation
[EthIfSwitchManagementSupport == ON]
EthIf_SetSwitchMgmtInfo(CtrlIdx :uint8, BufIdx :Eth_BufIdxType, MgmtInfoPtr :EthSwt_MgmtInfoType*) : Std_ReturnType
opt EthIf Global Time Support
[EthIfGlobalTimeSupport == ON]
EthIf_SwitchEnableTimeStamping(CtrlIdx :uint8, BufIdx :Eth_BufIdxType, MgmtInfo :EthSwt_MgmtInfoType) :Std_ReturnType
[Type == Follow_Up]:
correctionField += (T7-T4);
EthIf_Transmit(CtrlIdx :uint8, BufIdx :Eth_BufIdxType, FrameType :Eth_FrameType, TxConfirmation :boolean, LenByte :uint16, PhysAddrPtr :uint8)
EthTSyn_TxConfirmation(CtrlIdx :uint8, BufIdx :Eth_BufIdxType)
Switch-Port specific handling
opt EthSwt Global Time Support
[EthSwtGlobalTimeSupport == ON]
EthTSyn_SwitchIngressTimeStampIndication(CtrlIdx :uint8, DataPtr :uint8, MgmtInfoPtr : EthSwt_MgmtInfoType, timeStampPtr :Eth_TimeStampType)
EthTSyn_SwitchEgressTimeStampIndication(CtrlIdx :uint8, DataPtr :uint8, MgmtInfoPtr :EthSwt_MgmtInfoType, timeStampPtr :Eth_TimeStampType)
[Type == Sync]:
Get Port egress timestamp T7

Figure 21: Time Aware Bridge without GTM as Management CPU [`Sync`/`Follow_Up` Tx]

9.4.3 Time Aware Bridge without GTM as Management CPU – Rx

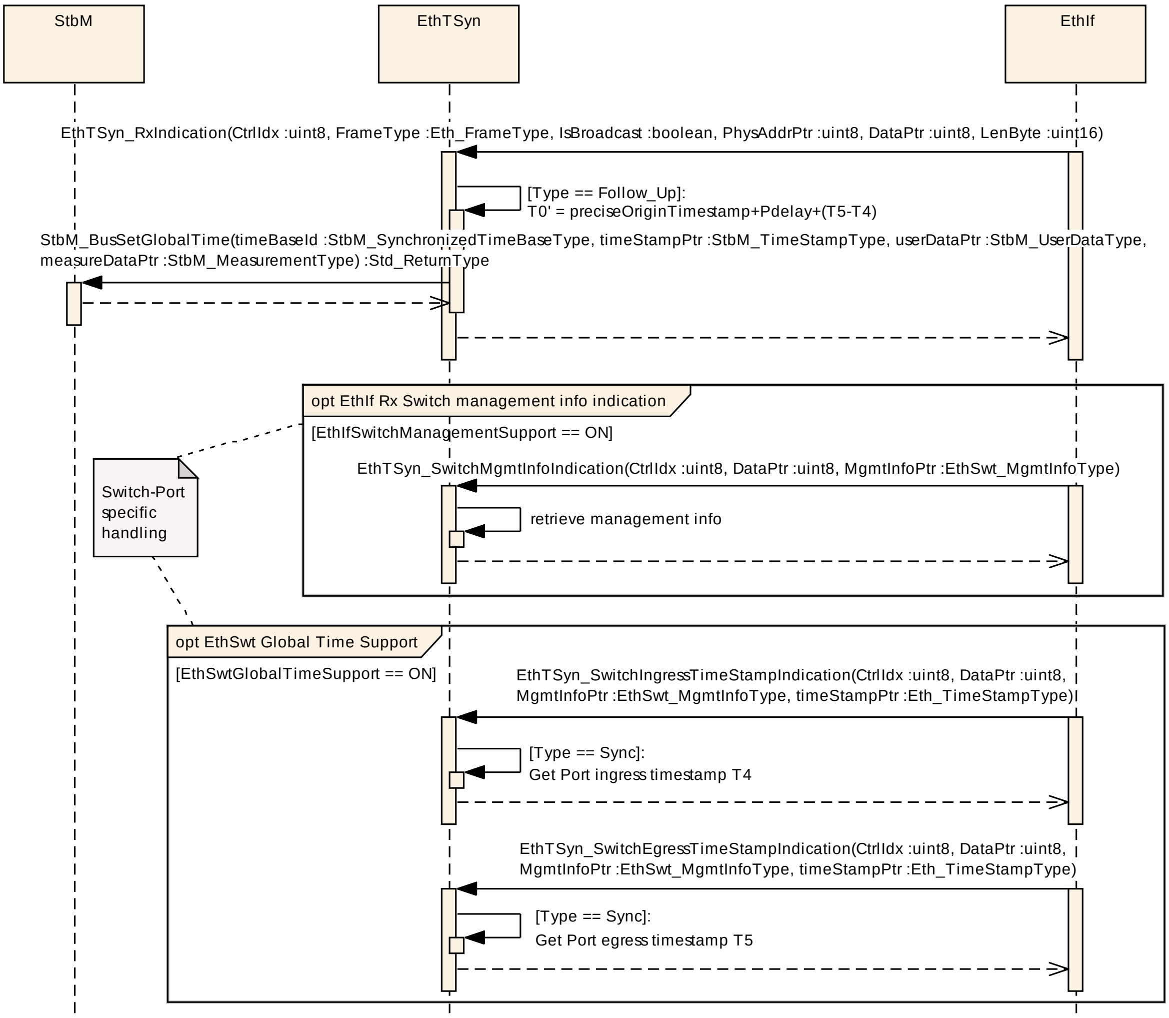

StbM
EthTSyn
EthIf
EthTSyn_RxIndication(CtrlIdx :uint8, FrameType :Eth_FrameType, IsBroadcast :boolean, PhysAddrPtr :uint8, DataPtr :uint8, LenByte :uint16)
[Type == Follow_Up]:
T0' = preciseOriginTimestamp+Pdelay+(T5-T4)
StbM_BusSetGlobalTime(timeBaseId :StbM_SynchronizedTimeBaseType, timeStampPtr :StbM_TimeStampType, userDataPtr :StbM_UserDataType,
measureDataPtr :StbM_MeasurementType) :Std_ReturnType
opt EthIf Rx Switch management info indication
[EthIfSwitchManagementSupport == ON]
EthTSyn_SwitchMgmtInfoIndication(CtrlIdx :uint8, DataPtr :uint8, MgmtInfoPtr :EthSwt_MgmtInfoType)
retrieve management info
Switch-Port
specific
handling
opt EthSwt Global Time Support
[EthSwtGlobalTimeSupport == ON]
EthTSyn_SwitchIngressTimeStampIndication(CtrlIdx :uint8, DataPtr :uint8,
MgmtInfoPtr :EthSwt_MgmtInfoType, timeStampPtr :Eth_TimeStampType)
[Type == Sync]:
Get Port ingress timestamp T4
EthTSyn_SwitchEgressTimeStampIndication(CtrlIdx :uint8, DataPtr :uint8,
MgmtInfoPtr :EthSwt_MgmtInfoType, timeStampPtr :Eth_TimeStampType)
[Type == Sync]:
Get Port egress timestamp T5

Figure 22: Time Aware Bridge without GTM as Management CPU [`Sync`/`Follow_Up` Rx]

10 Configuration specification

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

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

Section 10.3 specifies published information of the Global Time Synchronization over Ethernet.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in [5].

[SWS_EthTSyn_00051]⌈
The EthTSyn module shall support the configuration for Time Master, Time Slave and Time Gateway.
⌋(SRS_StbM_20052)

10.2 Containers and configuration parameters

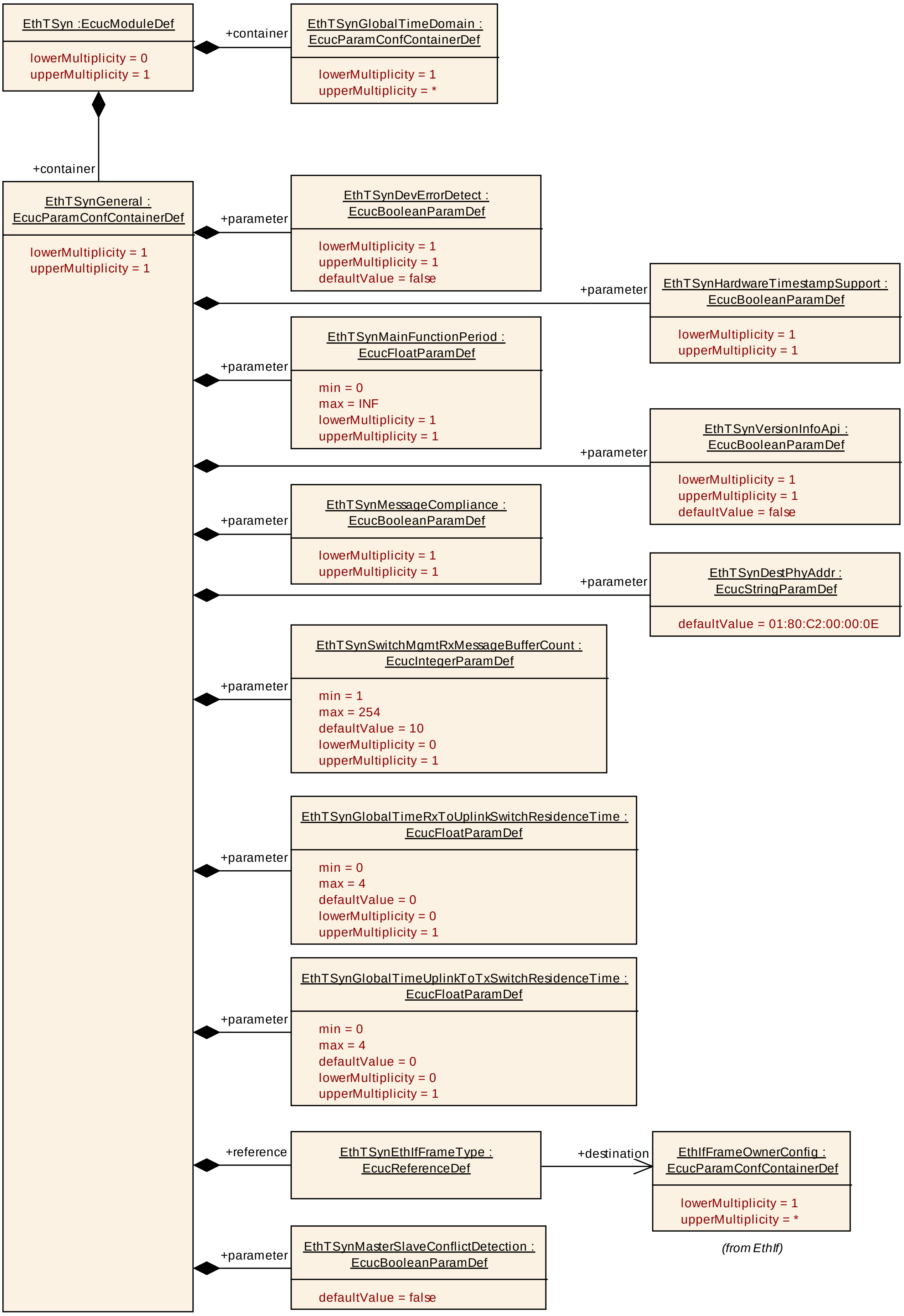
The following sections summarize all configuration parameters of the Global Time Synchronization over Ethernet. The detailed meaning of the parameters is described in chapters 7 and 8.
The module supports different post-build variants (previously known as post-build selectable configuration sets), but not post-build loadable configuration.

10.2.1 EthTSyn

SWS Item	**ECUC_EthTSyn_00001 :**
Module Name	*EthTSyn*
Module Description	Configuration of the Synchronized Time-base Manager (StbM) module with respect to global time handling on Ethernet.
Post-Build Variant Support	false
Supported Config Variants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthTSynGeneral	1	This container holds the general parameters of the Ethernet-specific Synchronized Time-base Manager
EthTSynGlobalTimeDomain	1..*	This represents the existence of a global time domain on Ethernet. The EthTSyn module can administrate several global time domains at the same time that in itself form a hierarchy of domains and sub-domains. If the EthTSyn exists it is assumed that at least one global time

		domain exists.


EthTSyn :EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
EthTSynGlobalTimeDomain : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+container
EthTSynGeneral : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
EthTSynDevErrorDetect : EcucBooleanParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
defaultValue = false
+parameter
EthTSynHardwareTimestampSupport : EcucBooleanParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
EthTSynMainFunctionPeriod : EcucFloatParamDef
min = 0
max = INF
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
EthTSynVersionInfoApi : EcucBooleanParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
defaultValue = false
+parameter
EthTSynMessageCompliance : EcucBooleanParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
EthTSynDestPhyAddr : EcucStringParamDef
defaultValue = 01:80:C2:00:00:0E
+parameter
EthTSynSwitchMgmtRxMessageBufferCount : EcucIntegerParamDef
min = 1
max = 254
defaultValue = 10
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthTSynGlobalTimeRxToUplinkSwitchResidenceTime : EcucFloatParamDef
min = 0
max = 4
defaultValue = 0
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthTSynGlobalTimeUplinkToTxSwitchResidenceTime : EcucFloatParamDef
min = 0
max = 4
defaultValue = 0
lowerMultiplicity = 0
upperMultiplicity = 1
+reference
EthTSynEthIfFrameType : EcucReferenceDef
+destination
EthIfFrameOwnerConfig : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
(from EthIf)
+parameter
EthTSynMasterSlaveConflictDetection : EcucBooleanParamDef
defaultValue = false

10.2.2 EthTSynGeneral

SWS Item	**ECUC_EthTSyn_00003 :**
Container Name	EthTSynGeneral
Description	This container holds the general parameters of the Ethernet-specific Synchronized Time-base Manager
Configuration Parameters	

SWS Item	**ECUC_EthTSyn_00058 :**		
Name	EthTSynDestPhyAddr		
Parent Container	EthTSynGeneral		
Description	Destination Physical Address (MAC-Address). Destination Physical Hardware Address (MAC-Address) of EthTSyn-gPTP Frames. Input format has to match xx:xx:xx:xx:xx:xx, where x stands for a hex value between 0 and F.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	01:80:C2:00:00:0E		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00002 :**		
Name	EthTSynDevErrorDetect		
Parent Container	EthTSynGeneral		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00060 :**	
Name	EthTSynGlobalTimeRxToUplinkSwitchResidenceTime	
Parent Container	EthTSynGeneral	
Description	This parameter is specifying the default value used for the residence time of the Ethernet Switch [Ingress to Uplink]. This value is used by the EthTSyn if the calculation of the residence time failed. Unit: seconds	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. 4[	

Default value	0		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00061 :**		
Name	EthTSynGlobalTimeUplinkToTxSwitchResidenceTime		
Parent Container	EthTSynGeneral		
Description	This parameter is specifying the default value used for the residence time of the Ethernet Switch [Uplink to Egress]. This value is used by the EthTSyn if the calculation of the residence time failed. Unit: seconds		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. 4[		
Default value	0		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00018 :**		
Name	EthTSynHardwareTimestampSupport		
Parent Container	EthTSynGeneral		
Description	Activate/Deactivate the hardware time stamping functionality of the Ethernet hardware. True: Timestamp is retrieved from the Ethernet hardware False: Timestamp is retrieved from the StbM		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00012 :**
Name	EthTSynMainFunctionPeriod
Parent Container	EthTSynGeneral
Description	Schedule period of the main function EthTSyn_MainFunction.

	Unit: seconds.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00075 :**		
Name	EthTSynMasterSlaveConflictDetection		
Parent Container	EthTSynGeneral		
Description	Enables master / slave conflict detection and notification. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00029 :**		
Name	EthTSynMessageCompliance		
Parent Container	EthTSynGeneral		
Description	• true: IEEE 802.1AS compliant message format will be used. • false: IEEE 802.1AS message format with AUTOSAR extension will be used.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00059 :**	
Name	EthTSynSwitchMgmtRxMessageBufferCount	
Parent Container	EthTSynGeneral	
Description	This parameter is used to determine the amount of Rx message buffers available in the EthTSyn when EthTSyn is used in a Bridge configuration.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 254	
Default value	10	
Post-Build Variant Multiplicity	false	

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

SWS Item	**ECUC_EthTSyn_00015 :**		
Name	EthTSynVersionInfoApi		
Parent Container	EthTSynGeneral		
Description	Activate/Deactivate the version information API (EthTSyn_GetVersionInfo). True: version information API activated False: version information API deactivated.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00062 :**		
Name	EthTSynEthIfFrameType		
Parent Container	EthTSynGeneral		
Description	The chosen frame owner determines which frames (in respect to ethertype) are received.		
Multiplicity	1		
Type	Reference to [EthIfFrameOwnerConfig]		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.3 EthTSynGlobalTimeDomain

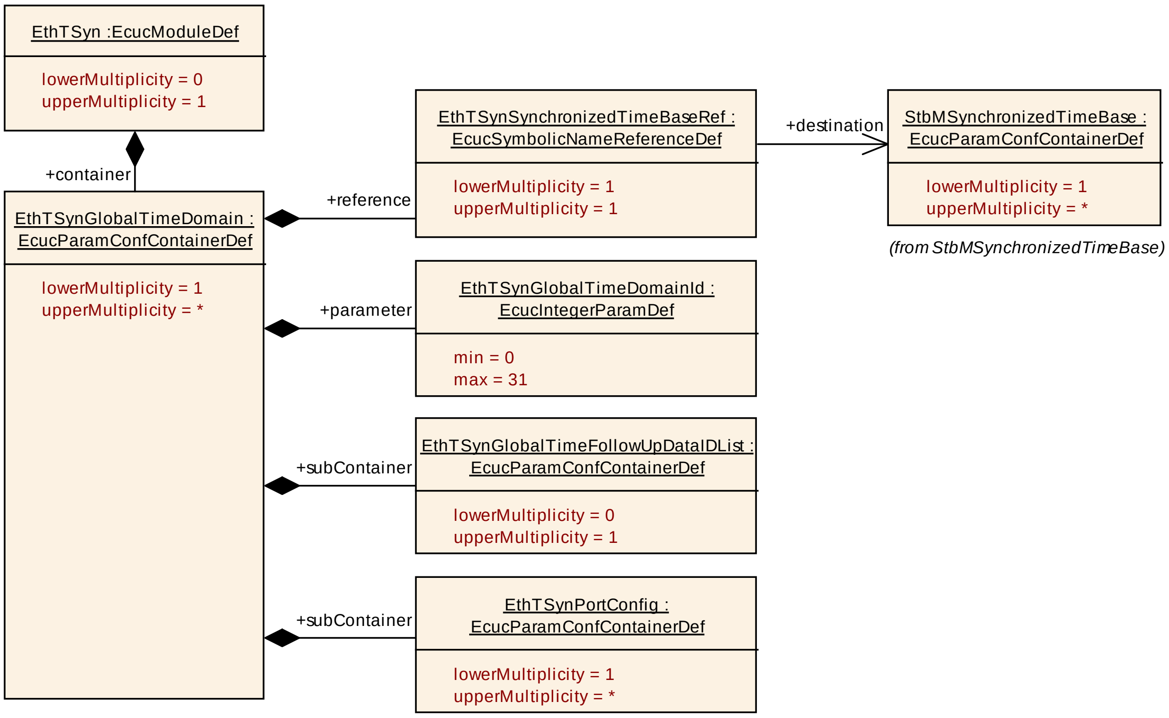
SWS Item	**ECUC_EthTSyn_00004 :**
Container Name	EthTSynGlobalTimeDomain
Description	This represents the existence of a global time domain on Ethernet. The EthTSyn module can administrate several global time domains at the same time that in itself form a hierarchy of domains and sub-domains. If the EthTSyn exists it is assumed that at least one global time domain exists.
Configuration Parameters	

SWS Item	**ECUC_EthTSyn_00005 :**

Name	EthTSynGlobalTimeDomainId		
Parent Container	EthTSynGlobalTimeDomain		
Description	The global time domain ID.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 31		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00013 :**		
Name	EthTSynSynchronizedTimeBaseRef		
Parent Container	EthTSynGlobalTimeDomain		
Description	Mandatory reference to the required synchronized time-base.		
Multiplicity	1		
Type	Symbolic name reference to [StbMSynchronizedTimeBase]		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

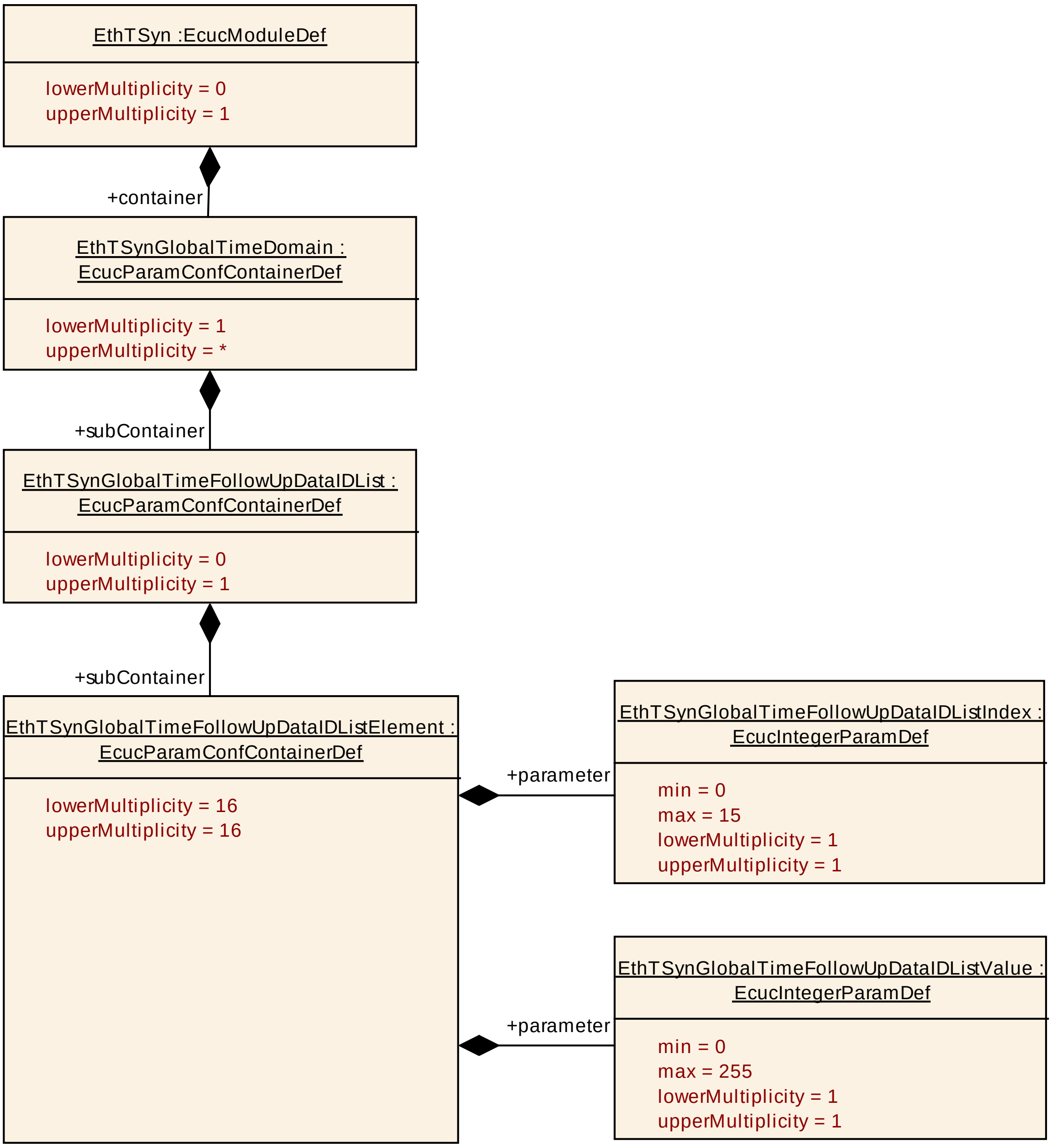
Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthTSynGlobalTimeFollowUpDataIDList	0..1	The DataIDList for Follow_Up message ensures the identification of data elements due to CRC calculation process.
EthTSynPortConfig	1..*	Configuration of the EthTSyn-Ports within the TimeDomain.



10.2.4 EthTSynGlobalTimeFollowUpDataIDList

SWS Item	**ECUC_EthTSyn_00030 :**		
Container Name	EthTSynGlobalTimeFollowUpDataIDList		
Description	The DataIDList for Follow_Up message ensures the identification of data elements due to CRC calculation process.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Configuration Parameters			

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



10.2.5 EthTSynGlobalTimeFollowUpDataIDListElement

SWS Item	**ECUC_EthTSyn_00031 :**
Container Name	EthTSynGlobalTimeFollowUpDataIDListElement
Description	Element of the DataIDList for Follow_Up message ensures the identification of data elements due to CRC calculation process.
Configuration Parameters	

SWS Item	**ECUC_EthTSyn_00032 :**
Name	EthTSynGlobalTimeFollowUpDataIDListIndex
Parent Container	EthTSynGlobalTimeFollowUpDataIDListElement
Description	Index of the DataIDList for Follow_Up message ensures the identification

	of data elements due to CRC calculation process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 15		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00033 :**		
Name	EthTSynGlobalTimeFollowUpDataIDListValue		
Parent Container	EthTSynGlobalTimeFollowUpDataIDListElement		
Description	Value of the DataIDList for Follow_Up message ensures the identification of data elements due to CRC calculation process.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.6 EthTSynPortConfig

SWS Item	**ECUC_EthTSyn_00063 :**		
Container Name	EthTSynPortConfig		
Description	Configuration of the EthTSyn-Ports within the TimeDomain.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Configuration Parameters			

SWS Item	**ECUC_EthTSyn_00034 :**		
Name	EthTSynFramePrio		
Parent Container	EthTSynPortConfig		
Description	This optional parameter, if present, indicates the priority of outgoing EthTSyn messages, if sent via VLAN (used for the 3-bit PCP field of the VLAN tag). If this optional parameter is not present, frames are sent without a priority and VLAN field.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default value	--		
Post-Build Variant	true		

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

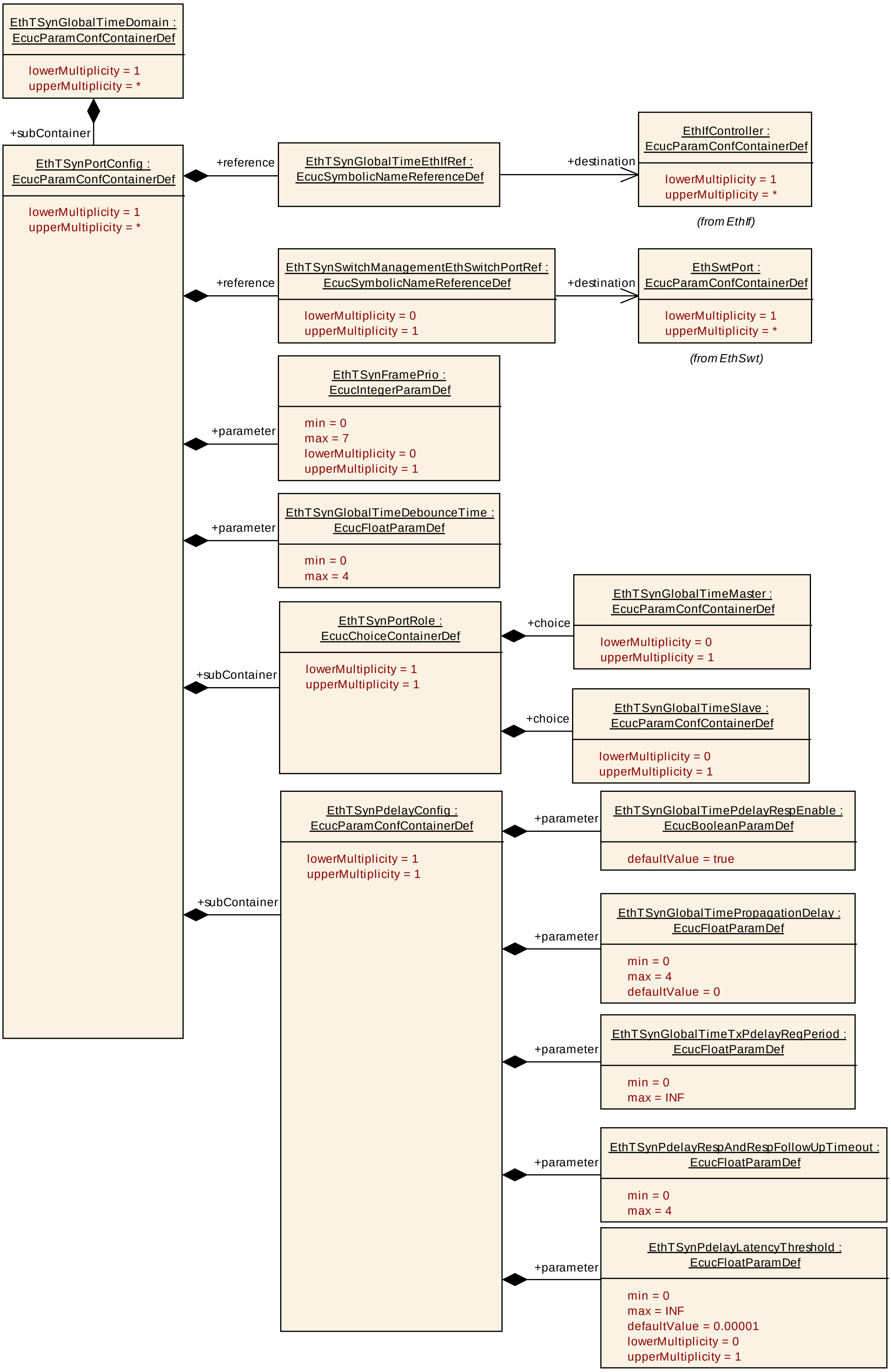
SWS Item	**ECUC_EthTSyn_00048 :**		
Name	EthTSynGlobalTimeDebounceTime		
Parent Container	EthTSynPortConfig		
Description	This represents the configuration of a TX debounce time for Sync and Follow_Up messages compared to a message before with the same PDU. Unit: seconds		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 4]		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00065 :**		
Name	EthTSynGlobalTimeEthIfRef		
Parent Container	EthTSynPortConfig		
Description	This represents the reference to the Ethernet interface taken to fetch the global time information.		
Multiplicity	1		
Type	Symbolic name reference to [EthIfController]		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00066 :**		
Name	EthTSynSwitchManagementEthSwitchPortRef		
Parent Container	EthTSynPortConfig		
Description	In an AVB-Bridge config, this reference is used to assign the EthTSyn-Port to an Ethernet Switch-Port.		
Multiplicity	0..1		
Type	Symbolic name reference to [EthSwtPort]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

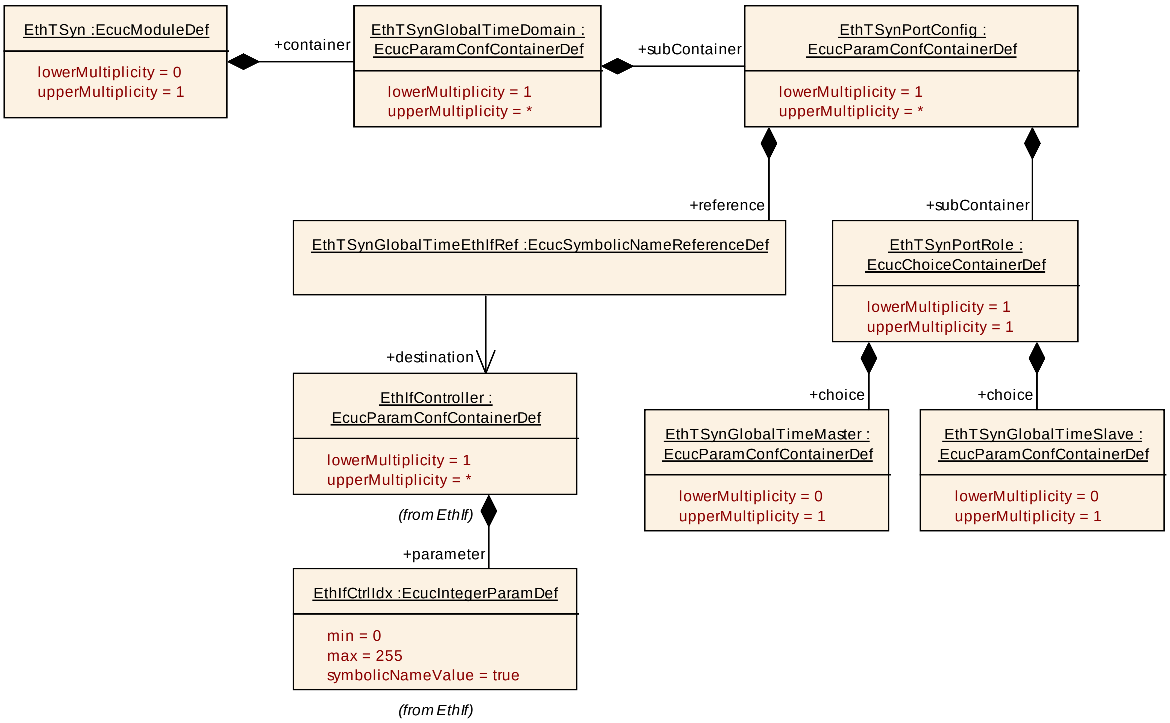
Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthTSynPdelayConfig	1	Configuration of cyclic propagation delay measurement.
EthTSynPortRole	1	Specifying the Role of the EthTSyn-Port (Master or Slave).


EthTSynGlobalTimeDomain : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+subContainer
EthTSynPortConfig : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+reference
EthTSynGlobalTimeEthIfRef : EcucSymbolicNameReferenceDef
+destination
EthIfController : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
(from EthIf)
+reference
EthTSynSwitchManagementEthSwitchPortRef : EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
EthSwtPort : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
(from EthSwt)
+parameter
EthTSynFramePrio : EcucIntegerParamDef
min = 0
max = 7
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthTSynGlobalTimeDebounceTime : EcucFloatParamDef
min = 0
max = 4
+subContainer
EthTSynPortRole : EcucChoiceContainerDef
lowerMultiplicity = 1
upperMultiplicity = 1
+choice
EthTSynGlobalTimeMaster : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+choice
EthTSynGlobalTimeSlave : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
EthTSynPdelayConfig : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
EthTSynGlobalTimePdelayRespEnable : EcucBooleanParamDef
defaultValue = true
+parameter
EthTSynGlobalTimePropagationDelay : EcucFloatParamDef
min = 0
max = 4
defaultValue = 0
+parameter
EthTSynGlobalTimeTxPdelayReqPeriod : EcucFloatParamDef
min = 0
max = INF
+parameter
EthTSynPdelayRespAndRespFollowUpTimeout : EcucFloatParamDef
min = 0
max = 4
+parameter
EthTSynPdelayLatencyThreshold : EcucFloatParamDef
min = 0
max = INF
defaultValue = 0.00001
lowerMultiplicity = 0
upperMultiplicity = 1

10.2.7 EthTSynPortRole

SWS Item	**ECUC_EthTSyn_00067 :**		
Choice container Name	EthTSynPortRole		
Description	Specifying the Role of the EthTSyn-Port (Master or Slave).		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Container Choices		
Container Name	***Multiplicity***	***Scope / Dependency***
EthTSynGlobalTimeMaster	0..1	Configuration of the global time master. Each global time domain is required to have exactly one global time master. This master may or may not exist on the configured ECU.
EthTSynGlobalTimeSlave	0..1	Configuration of a time slave. Each global time domain is required to have at least one time slave. The configured ECU may or may not represent a time slave.



10.2.8 EthTSynPdelayConfig

SWS Item	**ECUC_EthTSyn_00068 :**
Container Name	EthTSynPdelayConfig
Description	Configuration of cyclic propagation delay measurement.
Post-Build Variant	true

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

SWS Item	**ECUC_EthTSyn_00069 :**		
Name	EthTSynGlobalTimePdelayRespEnable		
Parent Container	EthTSynPdelayConfig		
Description	This parameter allows disabling Pdelay_Resp / Pdelay_Resp_Follow_Up transmission, if no Pdelay_Req messages are expected. FALSE: No Pdelay requests expected. Pdelay_Resp / Pdelay_Resp_Follow_Up transmission is disabled. TRUE: Pdelay requests expected. Pdelay_Resp / Pdelay_Resp_Follow_Up transmission is enabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00070 :**		
Name	EthTSynGlobalTimePropagationDelay		
Parent Container	EthTSynPdelayConfig		
Description	If cyclic propagation delay measurement is enabled, this parameter represents the default value of the propagation delay until the first actually measured propagation delay is available. If cyclic propagation delay measurement is disabled, this parameter replaces a measured propagation delay by a fixed value. Unit: seconds		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 4]		
Default value	0		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00071 :**	
Name	EthTSynGlobalTimeTxPdelayReqPeriod	
Parent Container	EthTSynPdelayConfig	
Description	This represents configuration of the TX period for Pdelay_Req messages. A value of 0 disables the cyclic Pdelay measurement. Unit: seconds	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF[	
Default value	--	

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

SWS Item	**ECUC_EthTSyn_00076 :**		
Name	EthTSynPdelayLatencyThreshold		
Parent Container	EthTSynPdelayConfig		
Description	Threshold for calculated Pdelay. If a measured Pdelay exceeds EthTSynPdelayLatencyThreshold, this value is discarded. Unit: seconds		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	1E-5		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00074 :**		
Name	EthTSynPdelayRespAndRespFollowUpTimeout		
Parent Container	EthTSynPdelayConfig		
Description	Timeout value for Pdelay_Resp and Pdelay_Resp_Follow_Up after a Pdelay_Req has been transmitted resp. a Pdelay_Resp has been received. A value of 0 deactivates this timeout observation. Unit: seconds		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 4]		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.9 EthTSynGlobalTimeMaster

SWS Item	**ECUC_EthTSyn_00008 :**
Container Name	EthTSynGlobalTimeMaster
Description	Configuration of the global time master. Each global time domain is required to have exactly one global time master. This master may or may not exist on the configured ECU.
Post-Build Variant Multiplicity	true

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

SWS Item	**ECUC_EthTSyn_00047 :**		
Name	EthTSynCyclicMsgResumeTime		
Parent Container	EthTSynGlobalTimeMaster		
Description	Defines the time where the 1st regular cycle time based message transmission takes place, after an immediate transmission before. Unit: seconds		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF[		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00039 :**		
Name	EthTSynGlobalTimeTxCrcSecured		
Parent Container	EthTSynGlobalTimeMaster		
Description	This represents the configuration of whether or not CRC is supported.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRC_NOT_SUPPORTED	This represents a configuration where CRC is not supported.	
	CRC_SUPPORTED	This represents a configuration where CRC is supported.	
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00010 :**		
Name	EthTSynGlobalTimeTxPeriod		
Parent Container	EthTSynGlobalTimeMaster		
Description	This represents configuration of the TX period. Unit: seconds		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF[		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00046 :**
Name	EthTSynImmediateTimeSync

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

SWS Item	**ECUC_EthTSyn_00072 : (Obsolete)**		
Name	EthTSynIsSystemWideGlobalTimeMaster		
Parent Container	EthTSynGlobalTimeMaster		
Description	This represents the configuration whether or not the global time master represents the root of a tree of global time domains. It is possible that several global time masters exist that have set this parameter set to true because the global time masters exist once per global time domain and one ECU may start several global time domains on different busses it is connected to. **Tags:** atp.Status=obsolete atp.StatusRevisionBegin=4.3.1		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00038 :**		
Name	EthTSynTLVFollowUpOFSSubTLV		
Parent Container	EthTSynGlobalTimeMaster		
Description	This represents the configuration of whether an AUTOSAR Follow_Up TLV OFS Sub-TLV is used or not. • true: This represents a configuration where an AUTOSAR Follow_Up TLV OFS Sub-TLV is used. • false: This represents a configuration where an AUTOSAR Follow_Up TLV OFS Sub-TLV is not used.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

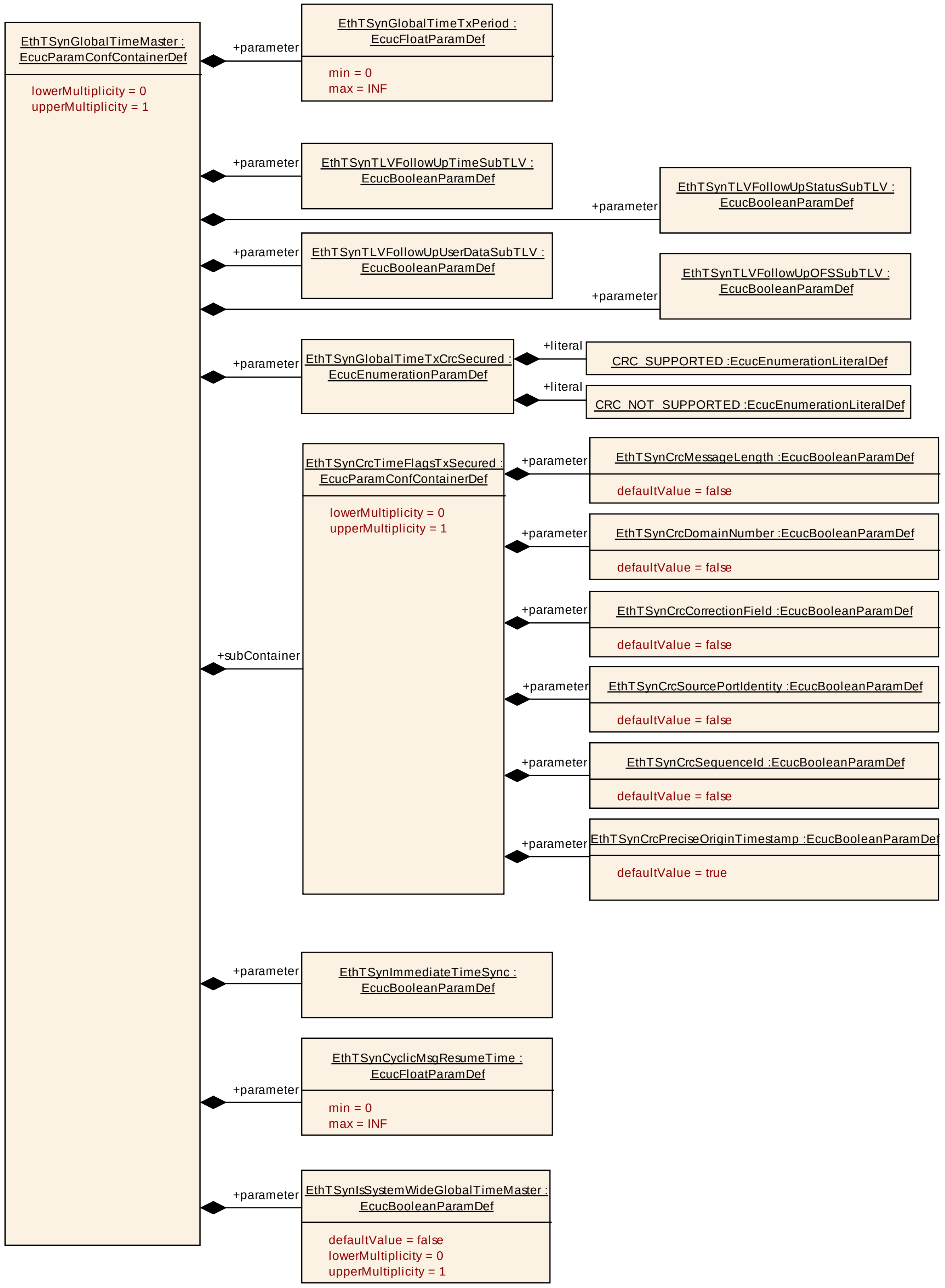
SWS Item	**ECUC_EthTSyn_00036 :**
Name	EthTSynTLVFollowUpStatusSubTLV
Parent Container	EthTSynGlobalTimeMaster

Description	This represents the configuration of whether an AUTOSAR Follow_Up TLV Status Sub-TLV is used or not. • true: This represents a configuration where an AUTOSAR Follow_Up TLV Status Sub-TLV is used. • false: This represents a configuration where an AUTOSAR Follow_Up TLV Status Sub-TLV is not used.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00035 :**		
Name	EthTSynTLVFollowUpTimeSubTLV		
Parent Container	EthTSynGlobalTimeMaster		
Description	This represents the configuration of whether an AUTOSAR Follow_Up TLV Time Sub-TLV is used or not. • true: This represents a configuration where an AUTOSAR Follow_Up TLV Time Sub-TLV is used. • false: This represents a configuration where an AUTOSAR Follow_Up TLV Time Sub-TLV is not used.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00037 :**		
Name	EthTSynTLVFollowUpUserDataSubTLV		
Parent Container	EthTSynGlobalTimeMaster		
Description	This represents the configuration of whether an AUTOSAR Follow_Up TLV UserData Sub-TLV is used or not. • true: This represents a configuration where an AUTOSAR Follow_Up TLV UserData Sub-TLV is used. • false: This represents a configuration where an AUTOSAR Follow_Up TLV UserData Sub-TLV is not used.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthTSynCrcTimeFlagsTxSecured	0..1	This container collects definitions which parts of the Follow_Up message elements shall be used for CRC calculation.


EthTSynGlobalTimeMaster :
EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthTSynGlobalTimeTxPeriod :
EcucFloatParamDef
min = 0
max = INF
+parameter
EthTSynTLVFollowUpTimeSubTLV :
EcucBooleanParamDef
+parameter
EthTSynTLVFollowUpStatusSubTLV :
EcucBooleanParamDef
+parameter
EthTSynTLVFollowUpUserDataSubTLV :
EcucBooleanParamDef
+parameter
EthTSynTLVFollowUpOFSSubTLV :
EcucBooleanParamDef
+parameter
EthTSynGlobalTimeTxCrcSecured :
EcucEnumerationParamDef
+literal
CRC_SUPPORTED :EcucEnumerationLiteralDef
+literal
CRC_NOT_SUPPORTED :EcucEnumerationLiteralDef
+subContainer
EthTSynCrcTimeFlagsTxSecured :
EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthTSynCrcMessageLength :EcucBooleanParamDef
defaultValue = false
+parameter
EthTSynCrcDomainNumber :EcucBooleanParamDef
defaultValue = false
+parameter
EthTSynCrcCorrectionField :EcucBooleanParamDef
defaultValue = false
+parameter
EthTSynCrcSourcePortIdentity :EcucBooleanParamDef
defaultValue = false
+parameter
EthTSynCrcSequenceId :EcucBooleanParamDef
defaultValue = false
+parameter
EthTSynCrcPreciseOriginTimestamp :EcucBooleanParamDef
defaultValue = true
+parameter
EthTSynImmediateTimeSync :
EcucBooleanParamDef
+parameter
EthTSynCyclicMsgResumeTime :
EcucFloatParamDef
min = 0
max = INF
+parameter
EthTSynIsSystemWideGlobalTimeMaster :
EcucBooleanParamDef
defaultValue = false
lowerMultiplicity = 0
upperMultiplicity = 1

10.2.10 EthTSynCrcTimeFlagsTxSecured

SWS Item	**ECUC_EthTSyn_00057 :**

Container Name	EthTSynCrcTimeFlagsTxSecured		
Description	This container collects definitions which parts of the Follow_Up message elements shall be used for CRC calculation.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Configuration Parameters			

SWS Item	**ECUC_EthTSyn_00042 :**		
Name	EthTSynCrcCorrectionField		
Parent Container	EthTSynCrcTimeFlagsTxSecured		
Description	correctionField from the Follow_Up Message Header shall be included in CRC calculation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00041 :**		
Name	EthTSynCrcDomainNumber		
Parent Container	EthTSynCrcTimeFlagsTxSecured		
Description	domainNumber from the Follow_Up Message Header shall be included in CRC calculation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00040 :**		
Name	EthTSynCrcMessageLength		
Parent Container	EthTSynCrcTimeFlagsTxSecured		
Description	messageLength from the Follow_Up Message Header shall be included in CRC calculation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00045 :**
Name	EthTSynCrcPreciseOriginTimestamp
Parent Container	EthTSynCrcTimeFlagsTxSecured

Description	preciseOriginTimestamp from the Follow_Up Message Field shall be included in CRC calculation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00044 :**		
Name	EthTSyncCrcSequenceId		
Parent Container	EthTSyncCrcTimeFlagsTxSecured		
Description	sequenceId from the Follow_Up Message Header shall be included in CRC calculation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00043 :**		
Name	EthTSyncCrcSourcePortIdentity		
Parent Container	EthTSyncCrcTimeFlagsTxSecured		
Description	sourcePortIdentity from the Follow_Up Message Header shall be included in CRC calculation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.11 EthTSynGlobalTimeSlave

SWS Item	**ECUC_EthTSyn_00009 :**		
Container Name	EthTSynGlobalTimeSlave		
Description	Configuration of a time slave. Each global time domain is required to have at least one time slave. The configured ECU may or may not represent a time slave.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	

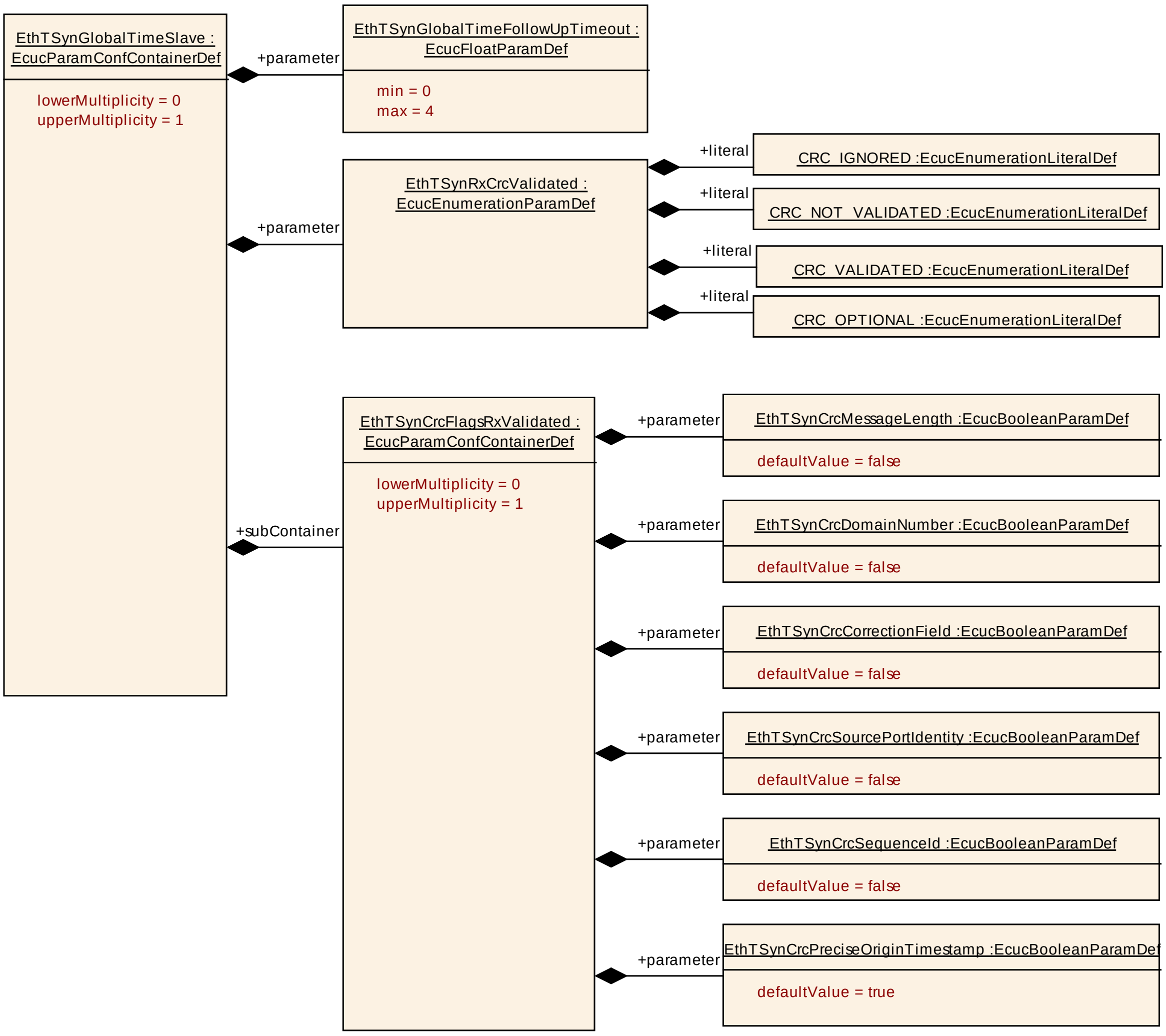
	Post-build time	--	
Configuration Parameters			

SWS Item	**ECUC_EthTSyn_00007 :**		
Name	EthTSynGlobalTimeFollowUpTimeout		
Parent Container	EthTSynGlobalTimeSlave		
Description	Timeout value of the Follow_Up message (of the subsequent Sync message). A value of 0 deactivates this timeout observation. Unit: seconds		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 4]		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00049 :**		
Name	EthTSynRxCrcValidated		
Parent Container	EthTSynGlobalTimeSlave		
Description	Definition of whether or not validation of the CRC takes place.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRC_IGNORED	EthTSyn ignores any CRC inside the Sub-TLVs.	
	CRC_NOT_VALIDATED	If EthTSynMessageCompliance is set to FALSE: EthTSyn discards Follow_Up messages with Sub-TLVs of Type 0x28, 0x44, 0x50 or 0x60.	
	CRC_OPTIONAL	If EthTSynMessageCompliance is set to FALSE: EthTSyn discards Follow_Up messages with Sub-TLVs of Type 0x28, 0x44, 0x50 or 0x60, that contain an incorrect CRC value.	
	CRC_VALIDATED	If EthTSynMessageCompliance is set to FALSE: EthTSyn discards Follow_Up messages with Sub-TLVs of Type 0x28, 0x44, 0x50 or 0x60, that contain an incorrect CRC value. EthTSyn rejects Follow_Up messages with Sub-TLVs of Type 0x34, 0x51 or 0x61.	
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
EthTSynCrcFlagsRxValidated	0..1	This container collects definitions which parts of the Follow_Up message elements shall be included in CRC

		validation.


EthTSynGlobalTimeSlave : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthTSynGlobalTimeFollowUpTimeout : EcucFloatParamDef
min = 0
max = 4
+parameter
EthTSynRxCrcValidated : EcucEnumerationParamDef
+literal
CRC_IGNORED :EcucEnumerationLiteralDef
+literal
CRC_NOT_VALIDATED :EcucEnumerationLiteralDef
+literal
CRC_VALIDATED :EcucEnumerationLiteralDef
+literal
CRC_OPTIONAL :EcucEnumerationLiteralDef
+subContainer
EthTSynCrcFlagsRxValidated : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
EthTSynCrcMessageLength :EcucBooleanParamDef
defaultValue = false
+parameter
EthTSynCrcDomainNumber :EcucBooleanParamDef
defaultValue = false
+parameter
EthTSynCrcCorrectionField :EcucBooleanParamDef
defaultValue = false
+parameter
EthTSynCrcSourcePortIdentity :EcucBooleanParamDef
defaultValue = false
+parameter
EthTSynCrcSequenceId :EcucBooleanParamDef
defaultValue = false
+parameter
EthTSynCrcPreciseOriginTimestamp :EcucBooleanParamDef
defaultValue = true

10.2.12 EthTSynCrcFlagsRxValidated

SWS Item	**ECUC_EthTSyn_00050 :**		
Container Name	EthTSynCrcFlagsRxValidated		
Description	This container collects definitions which parts of the Follow_Up message elements shall be included in CRC validation.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Configuration Parameters			

SWS Item	**ECUC_EthTSyn_00053 :**
Name	EthTSynCrcCorrectionField
Parent Container	EthTSynCrcFlagsRxValidated

Description	correctionField from the Follow_Up Message Header shall be included in CRC calculation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00052 :**		
Name	EthTSynCrcDomainNumber		
Parent Container	EthTSynCrcFlagsRxValidated		
Description	domainNumber from the Follow_Up Message Header shall be included in CRC calculation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00051 :**		
Name	EthTSynCrcMessageLength		
Parent Container	EthTSynCrcFlagsRxValidated		
Description	messageLength from the Follow_Up Message Header shall be included in CRC calculation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00056 :**		
Name	EthTSynCrcPreciseOriginTimestamp		
Parent Container	EthTSynCrcFlagsRxValidated		
Description	preciseOriginTimestamp from the Follow_Up Message Field shall be included in CRC calculation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00055 :**
Name	EthTSynCrcSequenceId

Parent Container	EthTSynCrcFlagsRxValidated		
Description	sequenceId from the Follow_Up Message Header shall be included in CRC calculation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_EthTSyn_00054 :**		
Name	EthTSynCrcSourcePortIdentity		
Parent Container	EthTSynCrcFlagsRxValidated		
Description	sourcePortIdentity from the Follow_Up Message Header shall be included in CRC calculation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.3 Published Information

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