

Document Title	Specification of Synchronized Time-Base Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	421

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R24-11

Document Change History			
Date	Release	Changed by	Description
2024-11-27	R24-11	AUTOSAR Release Management	<ul style="list-style-type: none"> • Offset Time Bases removed • Cloning of Time Bases enhanced • Several minor clarifications and corrections
2023-11-23	R23-11	AUTOSAR Release Management	<ul style="list-style-type: none"> • Support for a Disciplined HW Clock added • Time Validation enhanced: Fallback Virtual Local Time, rate validation and time progression monitoring added • Validation findings for "Secured Time Synchronization" incorporated • Type of Time Base no longer depends on the ID but on the newly introduced Type parameter
2022-11-24	R22-11	AUTOSAR Release Management	<ul style="list-style-type: none"> • Support for "Secured Time Synchronization" added • API StbM_GetCurrentTime modified to query the Time Tuple • Restorage of Global Time from persistent memory enhanced





2021-11-25	R21-11	AUTOSAR Release Management	<ul style="list-style-type: none"> • Support for CAN HW timestamping added • API for cloning of timebases added • Rate correction of the sync reception delay added • Several minor clarifications and corrections
2020-11-30	R20-11	AUTOSAR Release Management	<ul style="list-style-type: none"> • Update for Time Validation feature - support for time gateways - ring buffer added to pDelay data • New interface Set/GetBusProtocolParam added to access bus specific protocol parameters • Attribute "Variation" of the R-Port MeasurementNotification and DET Error from StbM_BusGetCurrentTime and StbM_BusSetGlobalTime APIs corrected • Timesync definitions moved to RS_TimeSync
2019-11-28	R19-11	AUTOSAR Release Management	<ul style="list-style-type: none"> • Time Validation (draft) • Multicore Distribution support (draft) • Clarification regarding behavior when Time Stamp or User Data is invalid • Clarification on StbM "Time Notifications" Feature • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	<ul style="list-style-type: none"> • Modifications to enhance the precision of Global Time Synchronization • Additional minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation





2017-12-08	4.3.1	AUTOSAR Release Management	<ul style="list-style-type: none"> • Corrections and clarification on how to apply rate correction • Clarifications on Time Base Status and Time Leap behavior • Additional minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2016-11-30	4.3.0	AUTOSAR Release Management	<ul style="list-style-type: none"> • Rate Correction added • Time precision measurement support added • Time/status notification mechanism added • Various enhancements and corrections
2015-07-31	4.2.2	AUTOSAR Release Management	<ul style="list-style-type: none"> • Config parameter argument added to StbM_Init • StbM_TimeStampRawType changed to uint32 • StbM_BusSetGlobalTime allow NULL as userDataPtr • "const" added to input arguments passed by pointer • Debugging support marked as obsolete
2014-10-31	4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> • Concept "Global Time Synchronization" incorporated to replace (and by that improve) original functionality and to support new functionality, e.g. support of CAN and Ethernet • Support for gateways to enable time domains spanning several busses • Due to deficiencies R4.0/1 content has been removed (e.g. customer API + polling of time-base providers). Exception: API to synchronize OS schedule tables



△

2014-03-31	4.1.3	AUTOSAR Release Management	<ul style="list-style-type: none"> • Clarification on Autonomous Time Maintenance
2013-10-31	4.1.2	AUTOSAR Release Management	<ul style="list-style-type: none"> • Parameter StbMMainFunctionPeriod added • Requirements StbM_0030 and 00035 removed • Restructuring of and clarification w.r.t. Service Interface related chapters • Parameters StbMFlexRayClusterRef / StbMTtcanClusterRef set to obsolete • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> • Added "Known Limitations" • Contradictions in error handling removed • Added chapter service interfaces • Added Subchapter 3.x due to SWS General Rollout • Reworked according to the new SWS_BSWGeneral
2011-12-22	4.0.3	AUTOSAR Administration	<ul style="list-style-type: none"> • Added functionality for absolute time provision
2010-09-30	3.1.5	AUTOSAR Administration	<ul style="list-style-type: none"> • SRS_General: SRS_BSW_00004 • Binding character of the Standardized AUTOSAR Interfaces mentioned in the SWS Documents. • Missing Port Driver DET Error Codes
2010-02-02	3.1.4	AUTOSAR Administration	<ul style="list-style-type: none"> • 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.

Contents

1	Introduction and functional overview	12
1.1	Use Cases	12
1.2	Functional Overview	12
2	Acronyms, Abbreviations and Definitions	16
2.1	Acronyms and Abbreviation	16
2.2	Definitions	16
3	Related documentation	20
3.1	Input documents & related standards and norms	20
3.2	Related specification	20
4	Constraints and assumptions	21
4.1	Limitations	21
4.1.1	OS ScheduleTable	21
4.1.2	Synchronized Time Base Identifier	21
4.1.3	Mode switches	21
4.1.4	Configuration	21
4.1.5	Fallback Virtual Local Time	21
4.1.6	Out of scope	22
4.2	Applicability to car domains	22
4.3	Conflicts	22
5	Dependencies to other modules	23
5.1	Code file structure	23
5.2	Header file structure	23
6	Requirements Tracing	24
7	Functional specification	31
7.1	Overview	31
7.1.1	Types of Time Bases	31
7.1.1.1	Synchronized Time Bases	32
7.1.1.2	Pure Local Time Bases	32
7.1.2	Roles of the StbM	32
7.1.2.1	Global Time Master	34
7.1.2.2	Time Slave	34
7.1.2.3	Time Gateway	34
7.2	Virtual Local Time	34
7.2.1	Primary Virtual Local Time	35
7.2.2	Fallback Virtual Local Time	37
7.3	Local Instance of the Global Time	39
7.3.1	Progressing the Local Instance of the Global Time by a free-running HW clock	40
7.3.1.1	Main Time Tuple	40

7.3.1.2	Main Time Triple	42
7.3.2	Progressing the Local Instance of the Global Time by a disciplined HW clock	43
7.3.3	Handling the Local Instance of the Global Time in absence of Primary Virtual Local Time	45
7.4	Synchronized Time Bases	48
7.4.1	Global Time Master	50
7.4.2	Time Slave	51
7.4.2.1	Calculation of the SyncLocal Time Tuple	53
7.4.2.2	Calculation of the Updated Rx Time Tuple	54
7.4.2.3	General	54
7.5	Pure Local Time Bases	55
7.6	Synchronization State	56
7.6.1	Global Time Master	57
7.6.2	Time Slaves	57
7.7	User Data	60
7.8	Startup behavior	61
7.8.1	Preconditions	61
7.8.2	Initialization	62
7.9	Shutdown behavior	63
7.10	Immediate Time Synchronization	64
7.11	Time Correction	65
7.11.1	Rate Correction Measurement (for Time Slaves)	65
7.11.1.1	Rate Measurement by Timesync Module	66
7.11.1.2	Rate Measurement by StbM	67
7.11.2	Rate and Offset Correction (for Time Slaves)	73
7.11.2.1	Offset Correction by Jump	74
7.11.2.2	Offset Correction by Rate Adaption	75
7.11.3	Time Extrapolation and Rate Correction for Global Time Masters	78
7.12	Time Base Cloning	81
7.13	Notification of Customers	86
7.13.1	Time Notifications	86
7.13.2	Status Notifications	90
7.14	Triggering Customers	94
7.15	Time Recording	96
7.15.1	General	96
7.15.2	Global Time Precision Measurement Support	96
7.15.2.1	Synchronized Time Base Record Table	97
7.15.2.2	Snapshot Conditions	99
7.16	Time Validation	101
7.16.1	Record Tables	102
7.16.1.1	Time Slave/Master	102
7.16.1.2	Pdelay Initiator/Responder	105
7.16.1.3	Common	107
7.16.2	Time Validation Snapshot Conditions	108

7.16.2.1	Time Slave/Master	108
7.16.2.2	Pdelay Initiator/Responder	110
7.16.2.3	Common	111
7.16.3	Monitoring the progression of the Local Instance of Global Time	112
7.16.4	Rate validation	114
7.16.4.1	Calculation of Rate Jitter	115
7.16.4.2	Calculation of Rate Wander	117
7.17	Freshness Value	120
7.18	Interaction with User Defined Timesync Module (CDD)	122
7.19	Multicore Distribution	123
7.20	Error Handling	123
7.21	Error Classification	123
7.21.1	Development Errors	124
7.21.2	Runtime Errors	124
7.21.3	Production Errors	124
7.21.4	Extended Production Errors	124
7.22	Version Check	124
8	API specification	125
8.1	Imported types	125
8.2	Type definitions	125
8.2.1	Extension to Std_ReturnType	125
8.2.2	StbM_ConfigType	126
8.2.3	StbM_MeasurementType	126
8.3	Function definitions	127
8.3.1	StbM_GetVersionInfo	127
8.3.2	StbM_Init	128
8.3.3	StbM_GetCurrentTime	129
8.3.4	StbM_GetCurrentSafeTime	130
8.3.5	StbM_GetCurrentVirtualLocalTime	131
8.3.6	StbM_GetFallbackVirtualLocalTime	132
8.3.7	StbM_SetGlobalTime	133
8.3.8	StbM_UpdateGlobalTime	134
8.3.9	StbM_SetUserData	136
8.3.10	StbM_BusSetGlobalTime	137
8.3.11	StbM_GetRateDeviation	139
8.3.12	StbM_SetRateCorrection	140
8.3.13	StbM_GetTimeLeap	141
8.3.14	StbM_GetTimeBaseStatus	142
8.3.15	StbM_CloneTimeBase	143
8.3.16	StbM_SetCloningOffset	144
8.3.17	StbM_StartTimer	145
8.3.18	StbM_GetSyncTimeRecordHead	146
8.3.19	StbM_TriggerTimeTransmission	147
8.3.20	StbM_GetTimeBaseUpdateCounter	148
8.3.21	StbM_GetMasterConfig	149

8.3.22	StbM_CanSetSlaveTimingData	150
8.3.23	StbM_FrSetSlaveTimingData	151
8.3.24	StbM_EthSetSlaveTimingData	151
8.3.25	StbM_CanSetMasterTimingData	153
8.3.26	StbM_FrSetMasterTimingData	153
8.3.27	StbM_EthSetMasterTimingData	154
8.3.28	StbM_EthSetPdelayInitiatorData	155
8.3.29	StbM_EthSetPdelayResponderData	156
8.3.30	StbM_GetBusProtocolParam	157
8.3.31	StbM_SetBusProtocolParam	158
8.3.32	StbM_GetTxFreshness	159
8.3.33	StbM_GetTxFreshnessTruncData	160
8.3.34	StbM_SPduTxConfirmation	162
8.3.35	StbM_GetRxFreshness	163
8.4	Callback notifications	164
8.5	Scheduled functions	164
8.5.1	StbM_MainFunction	164
8.6	Expected interfaces	165
8.6.1	Mandatory interfaces	165
8.6.2	Optional interfaces	165
8.6.3	Configurable interfaces	166
8.6.3.1	SyncTimeRecordBlockCallback	167
8.6.3.2	StatusNotificationCallback	167
8.6.3.3	<Customer>_TimeNotificationCallback	168
8.6.3.4	SPduTxConfirmationFct	169
8.6.3.5	GetTxFreshnessTruncDataFct	170
8.6.3.6	GetTxFreshnessFct	171
8.6.3.7	GetRxFreshnessFct	172
8.7	Service Interfaces	172
8.7.1	Provided Ports	173
8.7.1.1	GlobalTime_Master	173
8.7.1.2	GlobalTime_Slave	173
8.7.1.3	GlobalTime_StatusEvent	174
8.7.1.4	StartTimer	174
8.7.2	Required Ports	175
8.7.2.1	GlobalTime_TimeEvent	175
8.7.2.2	GlobalTime_Measurement	175
8.7.2.3	TimeBaseProviderNotification_Eth	176
8.7.2.4	TimeBaseProviderNotification_Fr	177
8.7.2.5	TimeBaseProviderNotification_Can	177
8.7.2.6	FreshnessManagement	178
8.7.3	Sender-Receiver Interfaces	178
8.7.3.1	StatusNotification	178
8.7.4	Client-Server-Interfaces	179
8.7.4.1	GlobalTime_Master	179
8.7.4.2	GlobalTime_Slave	182

8.7.4.3	StartTimer	185
8.7.4.4	TimeNotification	185
8.7.4.5	MeasurementNotification	186
8.7.4.6	TimeBaseProviderNotification_Eth	187
8.7.4.7	TimeBaseProviderNotification_Fr	188
8.7.4.8	TimeBaseProviderNotification_Can	189
8.7.4.9	FreshnessManagement	190
8.7.5	Implementation Data Types	193
8.7.5.1	StbM_PortIdType	194
8.7.5.2	StbM_SynchronizedTimeBaseType	194
8.7.5.3	StbM_TimeBaseStatusType	195
8.7.5.4	StbM_TimeBaseNotificationType	196
8.7.5.5	StbM_VirtualLocalTimeType	198
8.7.5.6	StbM_TimeStampShortType	199
8.7.5.7	StbM_TimeStampType	199
8.7.5.8	StbM_TimeTupleType	200
8.7.5.9	StbM_TimeTripleType	201
8.7.5.10	StbM_TimeDiffType	201
8.7.5.11	StbM_RateDeviationType	202
8.7.5.12	StbM_CloneConfigType	202
8.7.5.13	StbM_UserDataType	203
8.7.5.14	StbM_CustomerIdType	204
8.7.5.15	StbM_SyncRecordTableHeadType	204
8.7.5.16	StbM_SyncRecordTableBlockType	205
8.7.5.17	StbM_MasterConfigType	206
8.7.5.18	StbM_EthTimeMasterMeasurementType	206
8.7.5.19	StbM_FrTimeMasterMeasurementType	207
8.7.5.20	StbM_CanTimeMasterMeasurementType	208
8.7.5.21	StbM_EthTimeSlaveMeasurementType	209
8.7.5.22	StbM_FrTimeSlaveMeasurementType	210
8.7.5.23	StbM_CanTimeSlaveMeasurementType	211
8.7.5.24	StbM_PdelayInitiatorMeasurementType	212
8.7.5.25	StbM_PdelayResponderMeasurementType	213
8.7.5.26	StbM_TimeSyncType	214
8.7.5.27	StbM_ProtocolParamType	215
8.7.5.28	StbM_FreshnessArrayType	216
9	Sequence diagrams	217
9.1	StbM Initialization	217
9.2	Immediate Time Synchronisation	218
9.3	Explicit synchronization of OS ScheduleTable	220
9.4	Rx Time Tuple Processing Sequence	221
10	Configuration specification	226
10.1	How to read this chapter	226
10.2	Containers and configuration parameters	226
10.2.1	StbM	226

10.2.2	StbMGeneral	228
10.2.3	StbMSynchronizedTimeBase	231
10.2.4	StbMTimeCorrection	241
10.2.5	StbMLocalTimeClock	248
10.2.6	StbMDisciplinedClock	251
10.2.7	StbMFallbackTimeClock	252
10.2.8	StbMTimeRecording	255
10.2.9	StbMTimeValidation	257
10.2.10	StbMNotificationCustomer	261
10.2.11	StbMTriggeredCustomer	263
10.2.12	StbMFreshnessValueInformation	265
10.2.13	StbMFreshnessValue	270
10.3	Constraints	272
10.4	Published Information	272
A	Not applicable requirements	273
B	Change history of AUTOSAR traceable items	274
B.1	Traceable item history of this document according to AUTOSAR Release R24-11	274
B.1.1	Added Specification Items in R24-11	274
B.1.2	Changed Specification Items in R24-11	274
B.1.3	Deleted Specification Items in R24-11	275
B.1.4	Added Constraints in R24-11	275
B.1.5	Changed Constraints in R24-11	275
B.1.6	Deleted Constraints in R24-11	275
B.2	Traceable item history of this document according to AUTOSAR Release R23-11	275
B.2.1	Added Specification Items in R23-11	275
B.2.2	Changed Specification Items in R23-11	276
B.2.3	Deleted Specification Items in R23-11	276
B.2.4	Added Constraints in R23-11	276
B.2.5	Changed Constraints in R23-11	277
B.2.6	Deleted Constraints in R23-11	277

1 Introduction and functional overview

This document specifies the functionality, API and the configuration of the Synchronized Time-Base Manager (StbM) module. The purpose of the Synchronized Time-Base Manager is to provide Synchronized Time Bases to its customers, i.e., time bases, which are synchronized with time bases on other nodes of a distributed system.

1.1 Use Cases

Two main use cases are supported by the Synchronized Time-Base Manager:

- **Synchronization of RunnableEntities**

An arbitrary number of RunnableEntities must be executed synchronously. Synchronous means that they shall start with a well-defined and guaranteed relative offset (e.g. relative offset "0", means the execution shall occur at the same point in time). Such a requirement can be specified by the [1, AUTOSAR Timing Extensions] and must be fulfilled independently of the actual deployment of the software components. Typical examples of this use case are the sensor data read out or synchronous actuator triggering by different RunnableEntities.

- **Provision of absolute or relative time value**

The application (and other BSW modules) shall provide a central module that is responsible for the provision of information about absolute or relative time and progression of it.

Typical examples of this use case are:

- Sensor data fusion: Data from various sensor systems like radar or stereo multi-purpose cameras can be temporally correlated.
- Event data recording: In some cases, e.g. crash, it is desirable to store data about the events and the internal state of different ECUs. For a temporal correlation of these events and states a common time base is required.
- Access to synchronized calendar time for diagnostic events storage.

1.2 Functional Overview

Figure 1.1 illustrates how the Synchronized Time-Base Manager interacts with other modules.

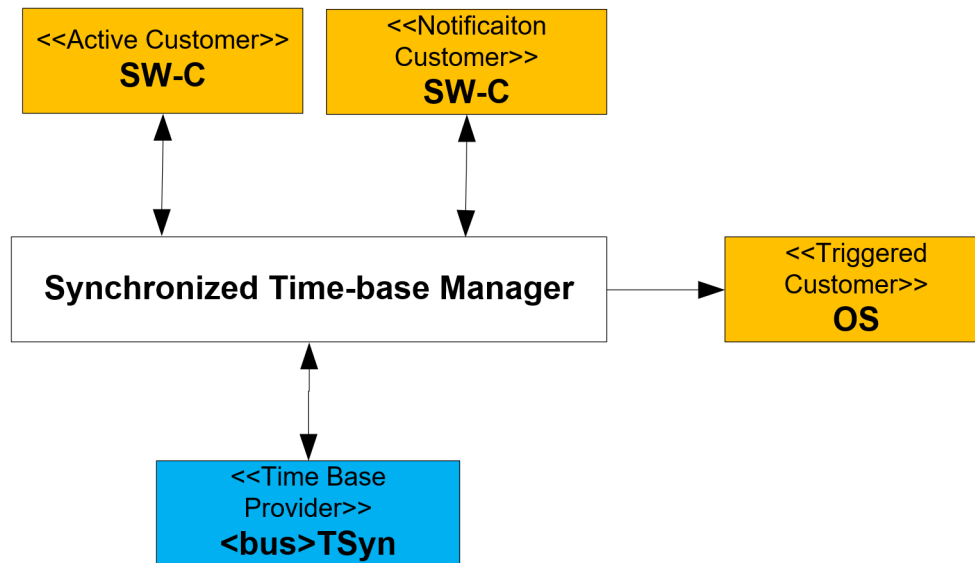


Figure 1.1: Synchronized Time-Base Manager as broker

The Synchronized Time-Base Manager itself does not provide means like network time protocols or time agreement protocols to synchronize its (local) Time Bases to Time Bases on other nodes. It interacts with the <Bus>TSyn modules of the BSW to achieve such synchronization. Those modules take as shown in [Figure 1.1](#) the role of a Time Base Provider and support above mentioned time protocols.

With the information retrieved from the provider modules, the Synchronized Time-Base Manager is able to synchronize its Time Bases to Time Bases on other nodes.

BSW modules and SW-C, which take the role of a customer, consume the time information provided and managed by the Synchronized Time-Base Manager. Three types of customers may be distinguished:

1. [Triggered Customer](#)
2. [Active Customer](#)
3. [Notification Customer](#)

Thus, the Synchronized Time-Base Manager acts as Time Base broker by offering the customers access to Synchronized Time Bases. Doing so, the Synchronized Time-Base Manager abstracts from the "real" Time Base provider.

Providing access to any Synchronized Time Base between the updates by the Time Base Providers is usually realized by using a Hardware Reference Clock; often in combination with a Software Counter which keeps track of the Hardware Reference Clock's overflows. Together Software Counter and Hardware Reference clock form the [Virtual Local Time](#) (despite the name the [Virtual Local Time](#) is an actually realized implementation).

This time is subsequently used to drive the time of the Time Bases, denoted as the *Local Instance of the Global Time*, taking account their Rate Deviations and Offsets to the underlying Virtual Local Time.

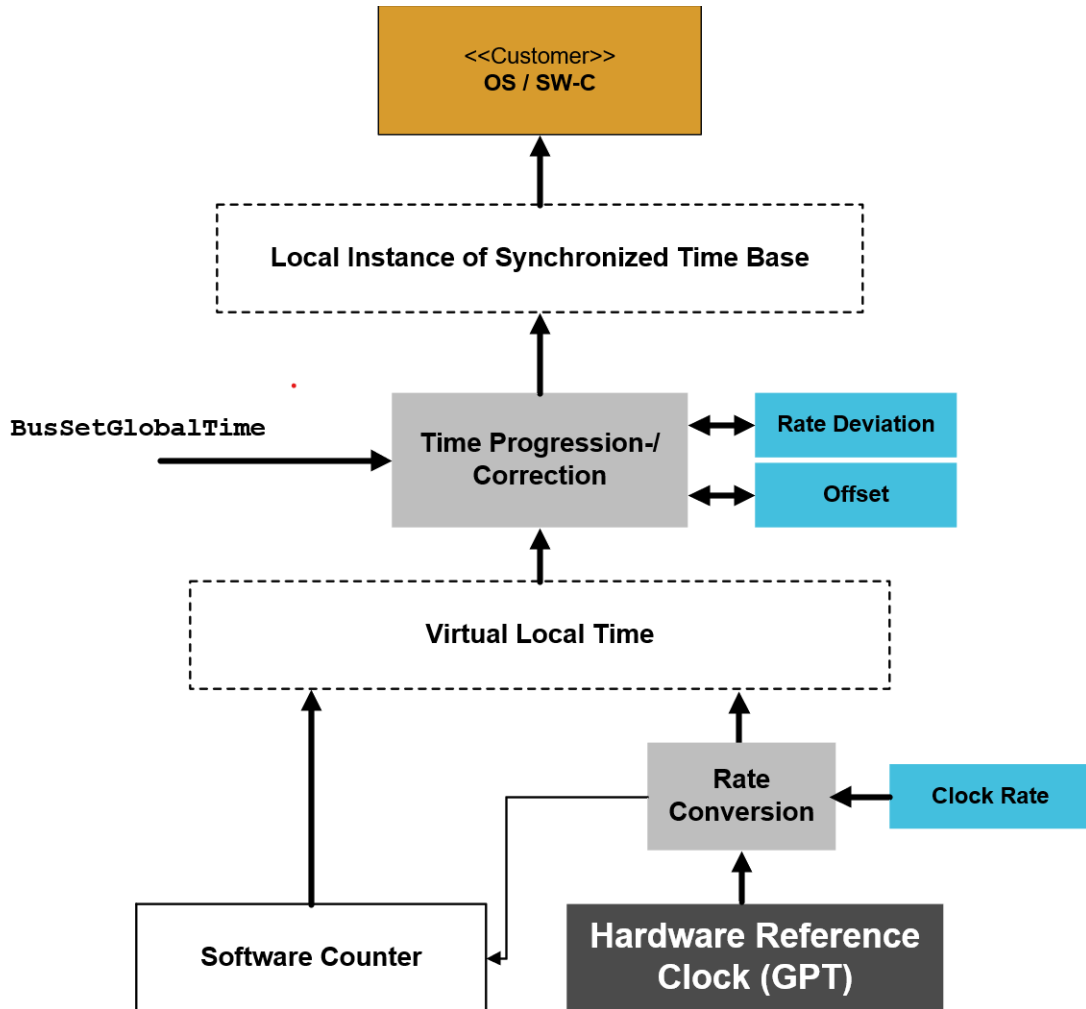


Figure 1.2: Abstract Working Principle of the Synchronized Time-Base Manager

The Synchronized Time-Base Manager will be the interface between the Freshness Value Manager (FVM) and the <Bus>TSyn modules. Once requested, the Freshness Value is provided to the <Bus>TSyn modules, and then used to secure the Time Synchronization Messages. The FVM can either be a SW-C or a CDD.

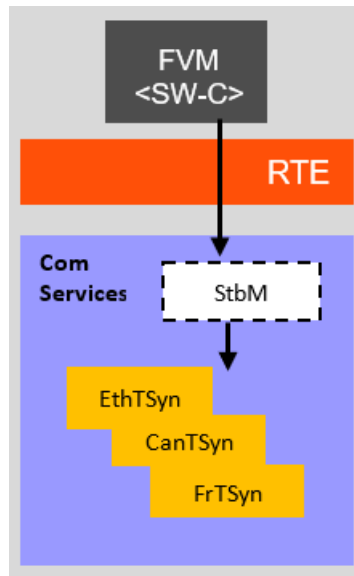


Figure 1.3: StbM as interface for the Freshness Value

The API for accessing the Synchronized Time Bases is provided to application software components as well as to other BSW modules:

- For the interaction with application software components, standardized AUTOSAR interfaces are specified in chapter 8 “[API specification](#)”.
- For the interaction with other BSW modules, respective interfaces are specified in chapter 8.7 “[Service Interfaces](#)”.

2 Acronyms, Abbreviations and Definitions

The glossary below includes acronyms and abbreviations relevant to the StbM module that are not included in

- the [2, AUTOSAR glossary]
- and [3, RS Time Synchronization]

2.1 Acronyms and Abbreviation

Abbreviation	Description
<Bus>TSyn	A bus specific Time Synchronization Provider module
CAN	Controller Area Network
CanTSyn	Time Synchronization Provider module for CAN
DET	Default Error Tracer
ECU	Electronic Control Unit
ETH	Ethernet
EthTSyn	Time Synchronization Provider module for Ethernet
FR	FlexRay
FrTSyn	Time Synchronization Provider module for FlexRay
FUP message	Follow-Up message for a Synchronized Time Base
FV	Freshness Value
FVM	Freshness Value Manager
GTM	Global Time Master
NvM	Non-volatile Memory
o_{oc}	Offset correction value
PTP	Precision Time Protocol
StbM	Synchronized Time-Base Manager
SYNC message	Time Synchronization message for a Synchronized Time Base
TG	Global Time
Timesync	Time Synchronization
TL	Local Instance of the Global Time
TL _{Sync}	Local Instance of the Global Time at re-synchronization
TSP	Time Synchronization Provider
TV	Virtual Local Time
TG _{Rx}	Global Time of Rx Time Tuple
TG _{URx}	Global Time of Updated Rx Time
r	(Applied) rate
r_{rc}	Rate (correction) as derived by rate measurement
r_{oc}	Rate offset as derived by Offset Correction by Rate Adaption

Table 2.1: Acronyms and Abbreviation

2.2 Definitions

Term	Definition
Active Time Base Customer	<p>A Time Base Customer that autonomously calls the Synchronized Time-Base Manager either</p> <ul style="list-style-type: none"> • to read time information for a Time Base from the Synchronized Time-Base Manager or • to update the Time Base maintained by the Synchronized Time-Base Manager according to application information
Applied Rate	<p>Rate that is applied for setting/calculating the Local Instance of the Global Time.</p> <ul style="list-style-type: none"> • For a Time Master of a Synchronized Time Base and for Pure Local Time Bases it is the same as the Rate Correction r_{rc} • For a Time Slave of a Synchronized Time Base it is calculated by adding r_{rc} and r_{oc}
Current Time Tuple	<p>Time Tuple which holds the current Local Instance of the Global Time and the corresponding current Virtual Local Time</p>
Disciplined HW Clock	<p>A HW clock, that can be adjusted in rate and offset to follow a reference clock.</p>
Fallback Virtual Local Time	<p>Secondary Virtual Local Time derived from a HW clock different than that one used for deriving the primary Virtual Local Time. The Fallback Virtual Local Time can be used to monitor the primary Virtual Local Time and as local replacement of the primary Virtual Local Time, if that one is not available.</p>
Global Time	<p>Time value that is provided by the Global Time Master, i.e., the reference time for the time synchronization across the network.</p>
hold-over	<p>A scenario, where the main time source is (typically) temporarily not available and a another time source provides a (possibly degraded) time information</p>
Local Instance of the Global Time	<p>Local extrapolation of the current value of the time between two time value samples received from the Global Time Master by means of the Virtual Local Time or a Disciplined HW Clock</p>
Main Time Triple	<p>Reference Time Triple $[TL_{Main}, TV_{Main}, TV_{Fallback}]$ which extends the Main Time Tuple $[TL_{Main}, TV_{Main}]$ by a third member $TV_{Fallback}$, i.e., the Fallback Virtual Local Time sampled at the point in time when the Main Time Tuple is updated. The Main Time Triple allows to calculate, Local Instance of the Global Time, if no Primary Virtual Local Time is available (but the Fallback Virtual Local Time is).</p>
Main Time Tuple	<p>Reference Time Tuple $[TL_{Main}, TV_{Main}]$ to calculate the Local Instance of the Global Time based on the Primary Virtual Local Time, if no Disciplined HW Clock is supported</p>

Term	Definition
Notification Time Base Customer	A Time Base Customer that is notified by the Synchronized Time-Base Manager, if the following Time Base related events occur: <ul style="list-style-type: none"> • Time Base status has changed (e.g. a timeout has occurred for a Time Base) • Time Base value has reached a given value, which has been previously set by the customer.
Offset Correction	Offset value needed to compensate a Time Offset
Offset Correction by Jump	A offset value is applied at re-synchronization. This is to compensate a Time Offset between StbM Local Instance of the Global Time TL and the time value TG_{URx} received from the Time Master.
Offset Correction Adaption Interval	Time interval StbMOffsetCorrectionAdaptionInterval during which an additional rate adaption r_{oc} is applied, if Offset Correction by Rate Adaption is configured.
Offset Correction by Rate Adaption	A rate offset is applied additionally to the Rate Correction . This is to smoothly reduce a Time Offset between StbM predicted time TL_{Sync} and the time value TG_{URx}
Rate Correction	Factor needed to compensate a Rate Deviation
Rate Deviation	A rate deviation value tells how much the frequency of a clock deviates from the rate of a reference clock
Rx Time Tuple	Time Tuple $[TG_{Rx}, TV_{Rx}]$ derived upon ingress of a Follow-up message and handed over by the BusTSyn modules to the StbM by StbM_BusSetGlobalTime
SYNC Ingress Processing Delay	Time interval on Timesync Module side between ingress of SYNC message and the forwarding of the Rx Time Tuple to the StbM by StbM_BusSetGlobalTime
SyncLocal Time Tuple	Time Tuple $[TL_{Sync}, TV_{Sync}]$ which is derived at TV_{Sync} <ul style="list-style-type: none"> • either based on [SWS_StbM_00355] • or derived from the Disciplined HW Clock, if supported.
Time Base Customer	The Synchronized Time-Base Manager supports 3 types of Time Base Customers: <ul style="list-style-type: none"> • Active Customer • Triggered Customer • Notification Customer
Time Offset	Time offset that remains after applying a Rate Correction

Term	Definition
Time Tuple	<p>The Timesync stack manages the time of a Time Base always via a Time Tuple structure, i.e., a <code>Global Time</code> sample and the corresponding sample of the <code>Virtual Local Time</code>, which are both sampled at the very same point in time. This is relevant for the following use cases:</p> <ol style="list-style-type: none"> 1. Timesync Modules provide the received <code>Global Time</code> in form of a Time Tuple to the StbM 2. Timesync Modules obtains the <code>Global Time</code> to transmit as a Time Tuple 3. The application sets the <code>Global Time</code>, which is immediately extended by the StbM to a Time Tuple by adding the current value of the <code>Virtual Local Time</code> 4. The application reads the current time as a Time Tuple <p>The StbM defines 5 special Time Tuples, which a sampled at well-defined points in time:</p> <ul style="list-style-type: none"> • <code>Main Time Tuple</code> (relevant for all use cases) • <code>Rx Time Tuple</code> (relevant for use case 1) • <code>Updated Rx Time Tuple</code> (relevant for use case 1) • <code>SyncLocal Time Tuple</code> (relevant for use case 1) • <code>Current Time Tuple</code> (relevant for use case 2, 3 and 4) <p>It is essential to guarantee to the integrity of the Time Tuple.</p>
Time Triple	<p>Derived from a <code>Time Tuple</code> by adding the corresponding <code>Fall-back Virtual Local Time</code> as a third member</p>
Triggered Time Base Customer	<p>A <code>Time Base Customer</code> that is triggered by the Synchronized Time-Base Manager. Thus, the Synchronized Time-Base Manager itself is aware of the required functionality of the customer and uses the defined interface of the customer to access it. This functionality is currently limited to synchronization of OS ScheduleTables.</p>
Updated Rx Time Tuple	<p>Time Tuple $[TG_{URx}, TV_{URx}]$ derived from the <code>Rx Time Tuple</code> by adding the rate corrected value of the <code>SYNC Ingress Processing Delay</code> when <code>StbM_BusSetGlobalTime</code> is called by the <code>BusTSyn</code> modules.</p>
Virtual Local Time	<p>Local time reference of the StbM derived from a HW clock used for progressing the <code>Local Instance of the Global Time</code></p>

Table 2.2: Definitions

3 Related documentation

3.1 Input documents & related standards and norms

- [1] Specification of Timing Extensions for Classic Platform
AUTOSAR_CP_TPS_TimingExtensions
- [2] Glossary
AUTOSAR_FO_TR_Glossary
- [3] Requirements on Time Synchronization
AUTOSAR_FO_RS_TimeSync
- [4] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral
- [5] Specification of Operating System
AUTOSAR_CP_SWS_OS
- [6] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral
- [7] System Template
AUTOSAR_CP_TPS_SystemTemplate
- [8] Specification of Secure Onboard Communication
AUTOSAR_CP_SWS_SecureOnboardCommunication
- [9] Complex Driver design and integration guideline
AUTOSAR_CP_EXP_CDDDesignAndIntegrationGuideline
- [10] Guide to BSW Distribution
AUTOSAR_CP_EXP_BSWDistributionGuide
- [11] IEEE Std 802.1AS-2011

3.2 Related specification

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

Thus, the specification SWS BSW General shall be considered as additional and required specification for StbM.

4 Constraints and assumptions

4.1 Limitations

The current module proposal has a number of limitations for the application of the Synchronized Time-Base Manager within an AUTOSAR system.

4.1.1 OS ScheduleTable

The Synchronized Time-Base Manager shall perform the functionality of synchronizing OS ScheduleTables with a respective Synchronized Time Base. However, the StbM considers only the case when the targeted OS ScheduleTable is explicitly synchronized. The implicit synchronization does not affect the StbM, because the synchronization mechanism bypasses the module (for more information about the difference between explicit and implicit synchronization, please refer to [5]). Thus, when talking in the following about synchronization of OS ScheduleTables, always the explicit one is meant.

4.1.2 Synchronized Time Base Identifier

The `StbMSynchronizedTimeBaseIdentifier` range (128 .. 65535) is currently reserved and might still be used by legacy applications (implementing Triggered Customers). The ID range will however be reassigned to new features in the next release. Legacy applications will then no longer be supported.

4.1.3 Mode switches

The Synchronized Time-Base Manager does not deal with mode switches during runtime.

4.1.4 Configuration

Postbuild configuration of the StbM is limited to enabling or disabling the functionality of a system wide Global Time Master for a Time Base (refer to `StbMIsSystemWideGlobalTimeMaster`).

4.1.5 Fallback Virtual Local Time

The Synchronized Time-Base Manager allows to configure a Fallback `Virtual Local Time`. If the Primary `Virtual Local Time` fails, the Fallback `Virtual Lo-`

`cal Time` provides only a `hold-over` capability to the local application. That means, that time information derived from that Fallback `Virtual Local Time` is not distributed on the network. In the next releases of the StbM the behavior of the Fallback `Virtual Local Time` might be extended to support additional use cases.

4.1.6 Out of scope

- Errors, which occurred during Global Time establishment and which are not caused by the module itself (e.g. loss of FlexRay global time is a FlexRay issue and is not an issue of the Synchronized Time-Base Manager).
- Errors, which occurred during interaction with customers.

Example: Calling the explicit OS `ScheduleTable` synchronization may cause an exception, because the delta between the submitted parameter `counterValue` and the OS internal counter is higher than the tolerance range of affected expiry points. Dealing with this exception is an OS issue, not an issue of the Synchronized Time-Base Manager.

4.2 Applicability to car domains

The concept is targeted at supporting time-critical and safety-related automotive applications such as airbag systems and braking systems. This doesn't mean that the concept has all that is required by such systems though, but crucial timing-related features that cannot be deferred to implementation are considered.

4.3 Conflicts

None.

5 Dependencies to other modules

5.1 Code file structure

For details refer to the chapter 5.1.6 "Code file structure" in [4, SWS BSW General].

5.2 Header file structure

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

In addition to the files defined in section 5.1.7 "Header file structure" of the [4, SWS BSW General], the StbM needs to include the file Os.h, CanIf.h, EthIf.h and Gpt.h.

[SWS_StbM_00065]

Upstream requirements: [SRS_BSW_00384](#)

[If a triggered customer is configured (refer to [StbMTriggeredCustomer](#)), StbM.c shall include Os.h to have access to the schedule table interface of the OS.]

[SWS_StbM_00246]

Upstream requirements: [SRS_BSW_00384](#)

[If time stamping via Ethernet shall be supported (refer to `EthIfGlobalTimeSupport`, which is referenced via [StbMLocalTimeHardware](#) or [StbMFallbackTimeHardware](#), if set to `EthTSynGlobalTimeDomain`), StbM.c shall include EthIf.h to have access to the interface of the EthIf module.]

[SWS_StbM_00538]

Status: DRAFT

Upstream requirements: [SRS_BSW_00384](#), [RS_TS_20070](#)

[If CAN hardware timestamping is supported (refer to configuration parameter `CanIfGlobalTimeSupport` in `CanIf`, which is referenced via [StbMLocalTimeHardware](#) or [StbMFallbackTimeHardware](#), if set to `CanTSynGlobalTimeDomain`), StbM.c shall include CanIf.h to have access to the interface of the CanIf module.]

[SWS_StbM_00426]

Upstream requirements: [RS_TS_00017](#), [RS_TS_00002](#)

[If time stamping via GPT shall be supported (which is referenced via [StbMLocalTimeHardware](#) or [StbMFallbackTimeHardware](#), if set to `GptChannelConfiguration`), StbM.c shall include Gpt.h to have access to the interface of the GPT module.]

6 Requirements Tracing

The following tables reference the requirements specified in [3, RS TimeSync] and [6, SRS BSWGeneral] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[RS_TS_00002]	The Implementation of Time Synchronization shall maintain its own Time Base independently of the acting role.	[SWS_StbM_00178] [SWS_StbM_00180] [SWS_StbM_00342] [SWS_StbM_00413] [SWS_StbM_00426] [SWS_StbM_00433] [SWS_StbM_00512] [SWS_StbM_00574] [SWS_StbM_00575] [SWS_StbM_00579]
[RS_TS_00003]	The TS shall initialize the Local Time Base with a configurable startup value	[SWS_StbM_00170]
[RS_TS_00004]	The Implementation of Time Synchronization shall initialize the Global Time Base with a configurable startup value.	[SWS_StbM_00171] [SWS_StbM_00578]
[RS_TS_00005]	The Implementation of Time Synchronization shall allow customers to have access to the Synchronized Time Base	[SWS_StbM_00142] [SWS_StbM_00173] [SWS_StbM_00195] [SWS_StbM_00240] [SWS_StbM_00244] [SWS_StbM_00247] [SWS_StbM_00248] [SWS_StbM_00262] [SWS_StbM_00263] [SWS_StbM_00267] [SWS_StbM_00434] [SWS_StbM_00436] [SWS_StbM_91013] [SWS_StbM_91027] [SWS_StbM_91028]
[RS_TS_00006]	The Implementation of Time Synchronization shall provide time information to TSP modules	[SWS_StbM_00173] [SWS_StbM_00195] [SWS_StbM_00434] [SWS_StbM_00436] [SWS_StbM_00437] [SWS_StbM_91006] [SWS_StbM_91027] [SWS_StbM_91029]
[RS_TS_00007]	The Implementation of Time Synchronization shall synchronize the Time Base of a Time Slave, on reception of a Time Master value	[SWS_StbM_00179] [SWS_StbM_00233] [SWS_StbM_00438] [SWS_StbM_00529] [SWS_StbM_00573]
[RS_TS_00008]	The Implementation of Time Synchronization shall continuously maintain its Time Bases based on a Time Base reference clock	[SWS_StbM_00178] [SWS_StbM_00180] [SWS_StbM_00413] [SWS_StbM_00433] [SWS_StbM_00437] [SWS_StbM_00512] [SWS_StbM_00515] [SWS_StbM_00539] [SWS_StbM_00574] [SWS_StbM_00575] [SWS_StbM_91006] [SWS_StbM_91029]
[RS_TS_00009]	The Implementation of Time Synchronization shall maintain the synchronization status of a Time Base	[SWS_StbM_00179] [SWS_StbM_00181] [SWS_StbM_00182] [SWS_StbM_00183] [SWS_StbM_00184] [SWS_StbM_00185] [SWS_StbM_00187] [SWS_StbM_00239] [SWS_StbM_00305] [SWS_StbM_00399] [SWS_StbM_00425] [SWS_StbM_00438] [SWS_StbM_00529] [SWS_StbM_00540] [SWS_StbM_00568] [SWS_StbM_00569] [SWS_StbM_00570] [SWS_StbM_00571] [SWS_StbM_00572] [SWS_StbM_00573] [SWS_StbM_91003]
[RS_TS_00010]	The Implementation of Time Synchronization shall allow customer on master side to set the Global Time	[SWS_StbM_00213] [SWS_StbM_00240] [SWS_StbM_00244] [SWS_StbM_00300] [SWS_StbM_00342] [SWS_StbM_00385] [SWS_StbM_00579]





Requirement	Description	Satisfied by
[RS_TS_00011]	The Implementation of Time Synchronization shall allow customers on master side to trigger time transmission by the TSP module	[SWS_StbM_00240] [SWS_StbM_00344] [SWS_StbM_00346] [SWS_StbM_00347] [SWS_StbM_00350] [SWS_StbM_00351] [SWS_StbM_00414]
[RS_TS_00014]	The Implementation of Time Synchronization shall allow customers to read User Data propagated via the TSP modules.	[SWS_StbM_00173] [SWS_StbM_00195] [SWS_StbM_00243] [SWS_StbM_00247] [SWS_StbM_00248] [SWS_StbM_00434] [SWS_StbM_00436] [SWS_StbM_91027]
[RS_TS_00015]	The Implementation of Time Synchronization shall allow customers to set User Data propagated via the TSP modules.	[SWS_StbM_00218] [SWS_StbM_00240] [SWS_StbM_00243] [SWS_StbM_00244] [SWS_StbM_00381] [SWS_StbM_00398] [SWS_StbM_00427]
[RS_TS_00016]	The Implementation of Time Synchronization shall notify customers about status events	[SWS_StbM_00277] [SWS_StbM_00279] [SWS_StbM_00280] [SWS_StbM_00284] [SWS_StbM_00285] [SWS_StbM_00286] [SWS_StbM_00287] [SWS_StbM_00288] [SWS_StbM_00290] [SWS_StbM_00299] [SWS_StbM_00345] [SWS_StbM_00526]
[RS_TS_00017]	The Implementation of Time Synchronization shall notify customers about elapsed pre-defined time span.	[SWS_StbM_00247] [SWS_StbM_00270] [SWS_StbM_00271] [SWS_StbM_00272] [SWS_StbM_00273] [SWS_StbM_00274] [SWS_StbM_00275] [SWS_StbM_00276] [SWS_StbM_00288] [SWS_StbM_00301] [SWS_StbM_00335] [SWS_StbM_00336] [SWS_StbM_00337] [SWS_StbM_00409] [SWS_StbM_00421] [SWS_StbM_00426] [SWS_StbM_00432] [SWS_StbM_91004]
[RS_TS_00018]	The Implementation of Time Synchronization shall support rate correction	[SWS_StbM_00352] [SWS_StbM_00353] [SWS_StbM_00355] [SWS_StbM_00356] [SWS_StbM_00359] [SWS_StbM_00361] [SWS_StbM_00362] [SWS_StbM_00364] [SWS_StbM_00366] [SWS_StbM_00367] [SWS_StbM_00368] [SWS_StbM_00370] [SWS_StbM_00371] [SWS_StbM_00372] [SWS_StbM_00373] [SWS_StbM_00374] [SWS_StbM_00375] [SWS_StbM_00376] [SWS_StbM_00377] [SWS_StbM_00378] [SWS_StbM_00390] [SWS_StbM_00395] [SWS_StbM_00396] [SWS_StbM_00397] [SWS_StbM_00400] [SWS_StbM_00412] [SWS_StbM_00422] [SWS_StbM_00424] [SWS_StbM_00431] [SWS_StbM_00440] [SWS_StbM_00527] [SWS_StbM_00576] [SWS_StbM_00577] [SWS_StbM_00580] [SWS_StbM_00581] [SWS_StbM_00583] [SWS_StbM_00586] [SWS_StbM_00587] [SWS_StbM_00588]
[RS_TS_00019]	The Implementation of Time Synchronization shall support damping offset correction	[SWS_StbM_00356]
[RS_TS_00021]	The Implementation of Time Synchronization shall provide interfaces to query the synchronization status	[SWS_StbM_00262]
[RS_TS_00024]	The Implementation of Time Synchronization shall support storage of the Time Base value at shutdown if configured as Time Master	[SWS_StbM_00172] [SWS_StbM_00555]





Requirement	Description	Satisfied by
[RS_TS_00025]	The Implementation of Time Synchronization shall provide fault detection mechanisms	[SWS_StbM_00031] [SWS_StbM_00183] [SWS_StbM_00187] [SWS_StbM_00199] [SWS_StbM_00540]
[RS_TS_00029]	The configuration of the Time Synchronization implementation shall allow the implementation to behave as a (vehicle wide) Time Master	[SWS_StbM_00195] [SWS_StbM_00213] [SWS_StbM_00244] [SWS_StbM_00408] [SWS_StbM_00490] [SWS_StbM_00491] [SWS_StbM_00492] [SWS_StbM_91001] [SWS_StbM_91002] [SWS_StbM_91027]
[RS_TS_00030]	The configuration of the Time Synchronization implementation shall allow the implementation to behave as a Time Slave	[SWS_StbM_00195] [SWS_StbM_00233] [SWS_StbM_00248] [SWS_StbM_00484] [SWS_StbM_00485] [SWS_StbM_00486] [SWS_StbM_91027]
[RS_TS_00031]	The configuration of the Time Synchronization implementation shall allow the implementation to behave as a Time Gateway	[SWS_StbM_00195] [SWS_StbM_00233] [SWS_StbM_00248] [SWS_StbM_00484] [SWS_StbM_00485] [SWS_StbM_00486] [SWS_StbM_00490] [SWS_StbM_00491] [SWS_StbM_00492] [SWS_StbM_91027]
[RS_TS_00032]	The Implementation of Time Synchronization shall trigger registered customers	[SWS_StbM_00020] [SWS_StbM_00022] [SWS_StbM_00077] [SWS_StbM_00084] [SWS_StbM_00092] [SWS_StbM_00093] [SWS_StbM_00107] [SWS_StbM_00142] [SWS_StbM_00302] [SWS_StbM_00303]
[RS_TS_00033]	The Implementation of Time Synchronization shall use a time format with a resolution of 1 ns	[SWS_StbM_00437]
[RS_TS_00034]	The Implementation of Time Synchronization shall provide measurement data to the application	[SWS_StbM_00233] [SWS_StbM_00247] [SWS_StbM_00306] [SWS_StbM_00307] [SWS_StbM_00308] [SWS_StbM_00309] [SWS_StbM_00310] [SWS_StbM_00312] [SWS_StbM_00313] [SWS_StbM_00314] [SWS_StbM_00315] [SWS_StbM_00317] [SWS_StbM_00319] [SWS_StbM_00320] [SWS_StbM_00322] [SWS_StbM_00323] [SWS_StbM_00331] [SWS_StbM_00332] [SWS_StbM_00339] [SWS_StbM_00382] [SWS_StbM_00384] [SWS_StbM_00387] [SWS_StbM_00428] [SWS_StbM_00458] [SWS_StbM_00459] [SWS_StbM_00460] [SWS_StbM_00461] [SWS_StbM_00462] [SWS_StbM_00463] [SWS_StbM_00465] [SWS_StbM_00466] [SWS_StbM_00467] [SWS_StbM_00468] [SWS_StbM_00469] [SWS_StbM_00470] [SWS_StbM_00471] [SWS_StbM_00472] [SWS_StbM_00473] [SWS_StbM_00474] [SWS_StbM_00475] [SWS_StbM_00476] [SWS_StbM_00477] [SWS_StbM_00478] [SWS_StbM_00479] [SWS_StbM_00480] [SWS_StbM_00481] [SWS_StbM_00482] [SWS_StbM_00483] [SWS_StbM_00484] [SWS_StbM_00485] [SWS_StbM_00486] [SWS_StbM_00487] [SWS_StbM_00490] [SWS_StbM_00491] [SWS_StbM_00492] [SWS_StbM_00493] [SWS_StbM_00496] [SWS_StbM_00497] [SWS_StbM_00500] [SWS_StbM_00501] [SWS_StbM_00503] [SWS_StbM_00504] [SWS_StbM_00505] [SWS_StbM_00506] [SWS_StbM_00507] [SWS_StbM_00508] [SWS_StbM_00509] [SWS_StbM_00510] [SWS_StbM_00511] [SWS_StbM_00522]





Requirement	Description	Satisfied by
		[SWS_StbM_00523] [SWS_StbM_00524] [SWS_StbM_00525] [SWS_StbM_00557] [SWS_StbM_00558] [SWS_StbM_00559] [SWS_StbM_00560] [SWS_StbM_00562] [SWS_StbM_00624]
[RS_TS_00035]	The Implementation of Time Synchronization shall provide a system service interface to applications	[SWS_StbM_00142] [SWS_StbM_00240] [SWS_StbM_00244] [SWS_StbM_00247] [SWS_StbM_00248] [SWS_StbM_00275] [SWS_StbM_00276] [SWS_StbM_00286] [SWS_StbM_00287] [SWS_StbM_00288] [SWS_StbM_00290]
[RS_TS_00036]	The Implementation of Time Synchronization shall provide a bus independent customer interface	[SWS_StbM_00241]
[RS_TS_00037]	The configuration of the Time Synchronization implementation shall allow the interaction with different types of customers	[SWS_StbM_00020] [SWS_StbM_00022] [SWS_StbM_00093] [SWS_StbM_00277] [SWS_StbM_00278] [SWS_StbM_00279] [SWS_StbM_00282] [SWS_StbM_00285] [SWS_StbM_00303] [SWS_StbM_00526]
[RS_TS_00038]	The Implementation of Time Synchronization shall copy Time Base information upon user request	[SWS_StbM_00240] [SWS_StbM_00530] [SWS_StbM_00531] [SWS_StbM_00532] [SWS_StbM_00533] [SWS_StbM_00534] [SWS_StbM_00535] [SWS_StbM_00536] [SWS_StbM_00584] [SWS_StbM_00585] [SWS_StbM_00629] [SWS_StbM_91011] [SWS_StbM_91012] [SWS_StbM_91030]
[RS_TS_00039]	The implementation of Time Synchronization shall provide Freshness Value (FV) to TSP modules required to secure the time information	[SWS_StbM_00541] [SWS_StbM_00542] [SWS_StbM_00543] [SWS_StbM_00551] [SWS_StbM_00552] [SWS_StbM_00553] [SWS_StbM_00554] [SWS_StbM_00564] [SWS_StbM_00565] [SWS_StbM_00566] [SWS_StbM_00567] [SWS_StbM_91014] [SWS_StbM_91015] [SWS_StbM_91016] [SWS_StbM_91017] [SWS_StbM_91018] [SWS_StbM_91019] [SWS_StbM_91021] [SWS_StbM_91022] [SWS_StbM_91023] [SWS_StbM_91024] [SWS_StbM_91025] [SWS_StbM_91026]
[RS_TS_00040]	Monitoring of the Synchronization Process Monitoring	[SWS_StbM_00596] [SWS_StbM_00597] [SWS_StbM_00606] [SWS_StbM_00616] [SWS_StbM_00622]
[RS_TS_00041]	Global Time Progress Monitoring	[SWS_StbM_00589] [SWS_StbM_00590] [SWS_StbM_00591] [SWS_StbM_00592] [SWS_StbM_00593] [SWS_StbM_00594] [SWS_StbM_00595] [SWS_StbM_00599] [SWS_StbM_00600] [SWS_StbM_00601] [SWS_StbM_00602] [SWS_StbM_00616] [SWS_StbM_00622]
[RS_TS_00042]	Continuous Time Progression Assurance in Error Cases	[SWS_StbM_00589] [SWS_StbM_00590] [SWS_StbM_00591] [SWS_StbM_00592] [SWS_StbM_00593] [SWS_StbM_00594] [SWS_StbM_00595] [SWS_StbM_00601] [SWS_StbM_00602] [SWS_StbM_00607]





Requirement	Description	Satisfied by
[RS_TS_00043]	Providing Relevant Information to SWCs/Adaptive Applications	[SWS_StbM_00603] [SWS_StbM_00604] [SWS_StbM_00608] [SWS_StbM_00609] [SWS_StbM_00610] [SWS_StbM_00611] [SWS_StbM_00612] [SWS_StbM_00613] [SWS_StbM_00614] [SWS_StbM_00615] [SWS_StbM_00616] [SWS_StbM_00617] [SWS_StbM_00618] [SWS_StbM_00619] [SWS_StbM_00620] [SWS_StbM_00621] [SWS_StbM_00622] [SWS_StbM_00623]
[RS_TS_20069]	The TimeSync over Ethernet module shall provide read / write access to bus protocol specific parameters	[SWS_StbM_00240] [SWS_StbM_00247] [SWS_StbM_00516] [SWS_StbM_00517] [SWS_StbM_91007] [SWS_StbM_91008] [SWS_StbM_91009] [SWS_StbM_91010]
[RS_TS_20070]	The Timesync over CAN module shall support hardware and software timestamping	[SWS_StbM_00538] [SWS_StbM_00539]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_StbM_00052]
[SRS_BSW_00172]	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	[SWS_StbM_00057] [SWS_StbM_00407]
[SRS_BSW_00301]	All AUTOSAR Basic Software Modules shall only import the necessary information	[SWS_StbM_00051] [SWS_StbM_00058] [SWS_StbM_00059]
[SRS_BSW_00305]	Data types naming convention	[SWS_StbM_00142]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_StbM_00041] [SWS_StbM_00196] [SWS_StbM_00197] [SWS_StbM_00214] [SWS_StbM_00215] [SWS_StbM_00219] [SWS_StbM_00220] [SWS_StbM_00234] [SWS_StbM_00235] [SWS_StbM_00264] [SWS_StbM_00268] [SWS_StbM_00269] [SWS_StbM_00296] [SWS_StbM_00298] [SWS_StbM_00340] [SWS_StbM_00341] [SWS_StbM_00348] [SWS_StbM_00349] [SWS_StbM_00379] [SWS_StbM_00380] [SWS_StbM_00386] [SWS_StbM_00391] [SWS_StbM_00392] [SWS_StbM_00394] [SWS_StbM_00405] [SWS_StbM_00406] [SWS_StbM_00415] [SWS_StbM_00416] [SWS_StbM_00444] [SWS_StbM_00445] [SWS_StbM_00448] [SWS_StbM_00449] [SWS_StbM_00451] [SWS_StbM_00452] [SWS_StbM_00455] [SWS_StbM_00456] [SWS_StbM_00457] [SWS_StbM_00488] [SWS_StbM_00489] [SWS_StbM_00494] [SWS_StbM_00495] [SWS_StbM_00498] [SWS_StbM_00499] [SWS_StbM_00502] [SWS_StbM_00503] [SWS_StbM_00518] [SWS_StbM_00519] [SWS_StbM_00520] [SWS_StbM_00521] [SWS_StbM_00537] [SWS_StbM_00544] [SWS_StbM_00545] [SWS_StbM_00546] [SWS_StbM_00547] [SWS_StbM_00548] [SWS_StbM_00549] [SWS_StbM_00550] [SWS_StbM_00556] [SWS_StbM_00598] [SWS_StbM_00605] [SWS_StbM_00625] [SWS_StbM_00626] [SWS_StbM_00627] [SWS_StbM_00628] [SWS_StbM_00630]





Requirement	Description	Satisfied by
[SRS_BSW_00327]	Error values naming convention	[SWS_StbM_00041]
[SRS_BSW_00333]	For each callback function it shall be specified if it is called from interrupt context or not	[SWS_StbM_00107] [SWS_StbM_00273] [SWS_StbM_00285]
[SRS_BSW_00337]	Classification of development errors	[SWS_StbM_00041] [SWS_StbM_00094]
[SRS_BSW_00339]	Reporting of production relevant error status	[SWS_StbM_00058] [SWS_StbM_00059]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_StbM_00052]
[SRS_BSW_00360]	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	[SWS_StbM_00273] [SWS_StbM_00285]
[SRS_BSW_00373]	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	[SWS_StbM_00057]
[SRS_BSW_00384]	The Basic Software Module specifications shall specify at least in the description which other modules they require	[SWS_StbM_00065] [SWS_StbM_00246] [SWS_StbM_00538]
[SRS_BSW_00385]	List possible error notifications	[SWS_StbM_00041]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_StbM_00041] [SWS_StbM_00094] [SWS_StbM_00196] [SWS_StbM_00197] [SWS_StbM_00214] [SWS_StbM_00215] [SWS_StbM_00219] [SWS_StbM_00220] [SWS_StbM_00234] [SWS_StbM_00235] [SWS_StbM_00264] [SWS_StbM_00268] [SWS_StbM_00269] [SWS_StbM_00296] [SWS_StbM_00298] [SWS_StbM_00340] [SWS_StbM_00341] [SWS_StbM_00348] [SWS_StbM_00349] [SWS_StbM_00379] [SWS_StbM_00380] [SWS_StbM_00386] [SWS_StbM_00391] [SWS_StbM_00392] [SWS_StbM_00394] [SWS_StbM_00405] [SWS_StbM_00406] [SWS_StbM_00415] [SWS_StbM_00416] [SWS_StbM_00444] [SWS_StbM_00445] [SWS_StbM_00448] [SWS_StbM_00449] [SWS_StbM_00451] [SWS_StbM_00452] [SWS_StbM_00455] [SWS_StbM_00456] [SWS_StbM_00457] [SWS_StbM_00488] [SWS_StbM_00489] [SWS_StbM_00494] [SWS_StbM_00495] [SWS_StbM_00498] [SWS_StbM_00499] [SWS_StbM_00502] [SWS_StbM_00503] [SWS_StbM_00518] [SWS_StbM_00519] [SWS_StbM_00520] [SWS_StbM_00521] [SWS_StbM_00537] [SWS_StbM_00544] [SWS_StbM_00545] [SWS_StbM_00546] [SWS_StbM_00547] [SWS_StbM_00548] [SWS_StbM_00549] [SWS_StbM_00550] [SWS_StbM_00556] [SWS_StbM_00598] [SWS_StbM_00605] [SWS_StbM_00625] [SWS_StbM_00626] [SWS_StbM_00627] [SWS_StbM_00628] [SWS_StbM_00630]
[SRS_BSW_00406]	API handling in uninitialized state	[SWS_StbM_00100] [SWS_StbM_00121]





Requirement	Description	Satisfied by
[SRS_BSW_00407]	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	[SWS_StbM_00066]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_StbM_00052] [SWS_StbM_00249]
[SRS_BSW_00429]	Access to OS is restricted	[SWS_StbM_00020] [SWS_StbM_00092]
[SRS_BSW_00457]	Callback functions of Application software components shall be invoked by the Basis SW	[SWS_StbM_00273] [SWS_StbM_00285]
[SRS_BSW_00459]	It shall be possible to concurrently execute a service offered by a BSW module in different partitions	[SWS_StbM_00513] [SWS_StbM_00514]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Overview

A Global Time network contains of a Time Master and at least one Time Slave. The Time Master is distributing via Time Synchronization messages the Global Time Base to the connected Time Slaves for each Time Domain. For CAN and Ethernet, the Time Slave corrects the received Global Time Base by considering the Time Stamp at the transmitter side and the own generated receiver Time Stamp. For FlexRay, the Time Synchronization mechanism is based on the local time of the FlexRay bus.

The local instance of the Time Base (derived from a HW reference clock) will be updated with the latest received valid value of the Global Time Base and runs autonomously until the next value of the Global Time Base is received.

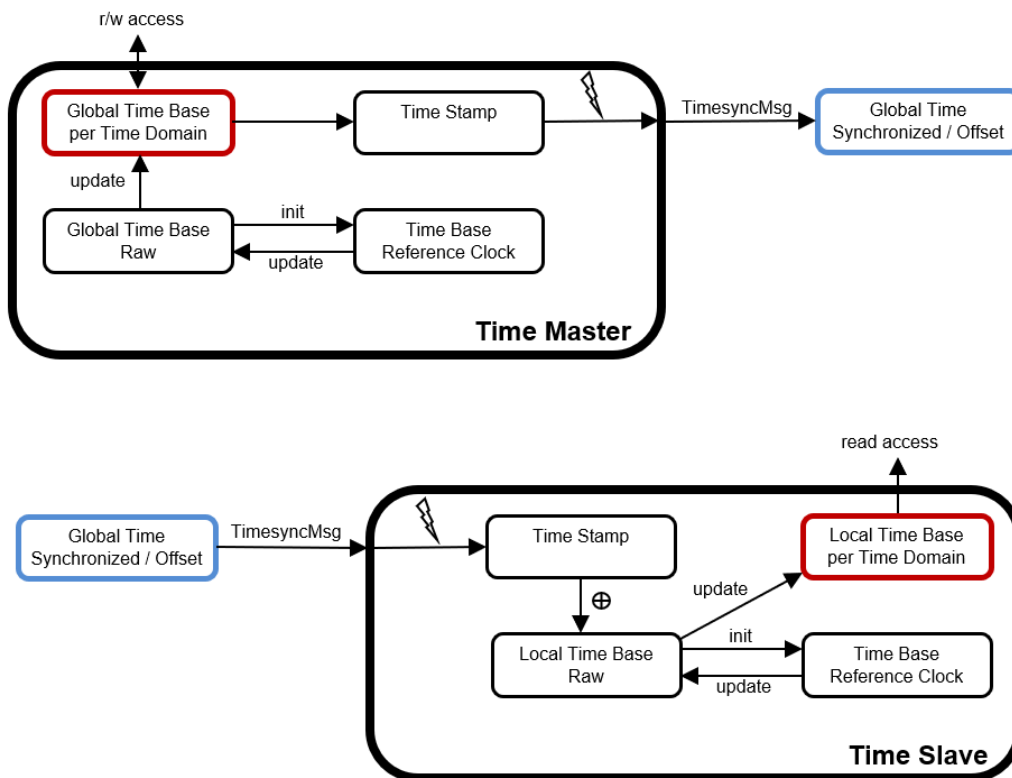


Figure 7.1: Global Time Base Distribution

7.1.1 Types of Time Bases

The type of a Time Base is defined by the Time Base specific configuration parameter `StbMSynchronizedTimeBaseType` as

- Synchronized Time Base (refer to chapter 7.1.1.1 “Synchronized Time Bases”)

- or Pure Local Time Base (refer to chapter [7.1.1.2 “Pure Local Time Bases”](#))

7.1.1.1 Synchronized Time Bases

Each Time Base has assigned a **Time Base identifier** in the range between 0 and 127. Time Base identifiers are configured in the StbM (refer to parameter `StbMSynchronizedTimeBaseIdentifier`).

Additionally, for each Synchronized Time Base the Timesync modules define a **Time Domain identifier**. The Time Domain identifier is actually sent/received by the Timesync modules in the Timesync messages on the network.

The Time Domain identifier is configured in the Timesync modules and it **is independent of the Time Base identifiers** in the StbM. Time Domain identifiers are in the range between 0 and 15 for CAN and FlexRay and between 0 and 127 for Ethernet.

Since the Time Base identifier in StbM is independent of the Time Domain identifier in the Timesync modules the following examples represent valid configurations:

- a Global Time Master contains a single Synchronized Time Base with Time Base identifier 0 that is transmitted as Time Domain 0 on Ethernet and Time Domain 1 on CAN
- a Time Slave receives Time Domain 0 on Ethernet and stores it as Time Base 0 whereas it also receives Time Domain 1 on CAN and stores it as Time Base 2.

The actual linking of a **Time Domain** to a **Time Base** is done by a parameter `<bus>TSynSynchronizedTimeBaseRef` per Time Domain in the Timesync modules.

7.1.1.2 Pure Local Time Bases

For details of Pure Local Time Bases refer to chapter [7.5 “Pure Local Time Bases”](#).

7.1.2 Roles of the StbM

Depending on its configuration the StbM may take one of the following three roles for a Time Base:

- Global Time Master
- Time Slave
- Time Gateway

In each role specific functionality is supported or not supported.

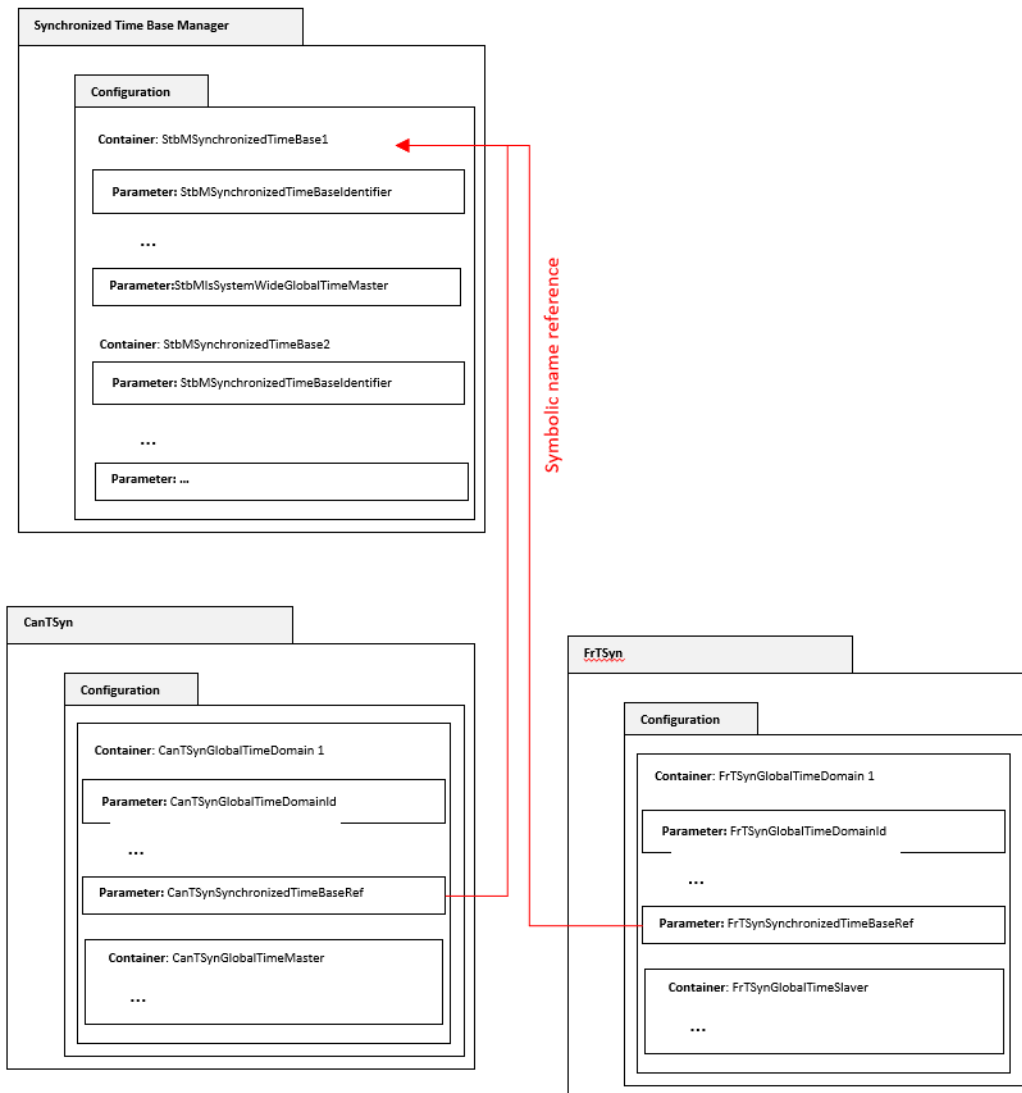


Figure 7.2: Configuration of the StbM Role per Time Base

Example: In Figure 7.2 the Time Base `StbMSynchronizedTimeBase1` is referenced by two Time Domains `CanTSynGlobalTimeDomain 1` and `FrTSynGlobalTimeDomain 1` from within a `CanTSyn` and a `FrTSyn Timesync` module, respectively.

`CanTSynGlobalTimeDomain 1` is configured as a Time Master and `FrTSynGlobalTimeDomain 1` as a Time Slave. This makes the StbM a Time Gateway for Time Base `StbMSynchronizedTimeBase1`.

If Time Base `StbMSynchronizedTimeBase1` would have been referenced by only one of the Time Domains - `CanTSynGlobalTimeDomain 1` or `FrTSynGlobalTimeDomain 1` - the StbM would have become a Time Master or a Time Slave for Time Base `StbMSynchronizedTimeBase1`, respectively.

Note: For system level representation of roles refer to figure 9.1 ("Big Picture of AUTOSAR global time synchronization") in [7, TPS System Template]

7.1.2.1 Global Time Master

A Global Time Master is the system wide origin for a given Time Base. Its Time Base values are distributed via the network to the Time Slaves.

[SWS_StbM_00408]

Upstream requirements: [RS_TS_00029](#)

[[StbM_GetMasterConfig](#) shall return the value of the configuration parameter [Stb-MIsSystemWideGlobalTimeMaster](#) for the Time Base [timeBaseId](#). This is to check, if the StbM is configured as system wide Global Time Master for a specific Time Base.]

7.1.2.2 Time Slave

In the role of a Time Slave the StbM updates its internally maintained local Time Base based on Global Time Base values, which are provided by the corresponding Timesync module.

7.1.2.3 Time Gateway

A Time Gateway in the StbM is a Time Base which is referenced by one Time Slave and one or more Time Masters. The Time Slave, which references a StbM Time Gateway receives Timesync messages on the corresponding bus and passes the received Time Base values to the StbM. Every Time Master referencing the Time Gateway retrieves the Gateway Time Base values from the StbM and transmits those on the bus. Depending on configuration the reception on slave side can or cannot automatically trigger the transmission on the master side automatically.

So, Timesync messages are not routed directly through an AUTOSAR Time Gateway. This is because routing delays need to be compensated.

7.2 Virtual Local Time

The [Virtual Local Time](#) is derived from a hardware reference clock (see [Figure 7.1](#)). The following hardware reference clocks are supported:

- OS counter
- GPT counter
- Ethernet free-running counter (used for ingress and egress timestamping)

Each Synchronized and Pure Local Time Base has to have one [Virtual Local Time](#) assigned to progress the [Local Instance of the Global Time](#) for that Time Bases. That [Virtual Local Time](#) is denoted as the **Primary Virtual Local Time** of the corresponding Time Base (refer to chapter [7.2.1 “Primary Virtual Local Time”](#)).

Optionally (e.g. when Time Validation is supported), each Synchronized Time Base may have a secondary [Virtual Local Time](#) assigned, denoted as the **Fallback Virtual Local Time** of the Time Base (refer to chapter [7.2.2 “Fallback Virtual Local Time”](#)).

In this document, if not explicitly stated otherwise, the term [Virtual Local Time](#) refers to the **Primary Virtual Local Time**.

7.2.1 Primary Virtual Local Time

The Primary [Virtual Local Time](#) is configured by the configuration container [StbMLocalTimeClock](#) referenced by the Time Base. Each Time Base might have a different Primary [Virtual Local Time](#) configured.

[SWS_StbM_00512]

Upstream requirements: [RS_TS_00008](#), [RS_TS_00002](#)

[If [StbMLocalTimeHardware](#) references a Gpt Channel as local time source for a Synchronized Time Base, the StbM shall derive the Virtual Local Time from the value of the corresponding GPT timer.

The elapsed timer value shall be read via `Gpt_GetTimeElapsed`.]

[SWS_StbM_00352]

Upstream requirements: [RS_TS_00018](#)

[The StbM shall use the factor ($\text{StbMClockPrescaler}/\text{StbMClockFrequency}$) to convert the time of its local hardware reference clock to the actual time of the Virtual Local Time (refer to [StbM_VirtualLocalTimeType](#)), if the Virtual Local Time is derived from a GPT or OsCounter (refer to [StbMLocalTimeHardware](#)).]

Note: Rationale is that a tick duration of the hardware reference clock does not necessarily have to match the resolution of the Virtual Local Time.

[SWS_StbM_00515]

Upstream requirements: [RS_TS_00008](#)

[If the range of the corresponding HW reference counter is less than that of the Virtual Local Time (refer to [StbM_VirtualLocalTimeType](#)), the StbM shall extend the range accordingly.]

Note: Depending on the HW reference clock one way of extending the range is to count overflows of the HW reference clock.

[SWS_StbM_00178]

Upstream requirements: [RS_TS_00008](#), [RS_TS_00002](#)

[If `EthIfGlobalTimeSupport` is set to `TRUE` for a Synchronized Time Base, then the StbM shall get the current value of the [Virtual Local Time](#) from the freerunning HW counter of the corresponding Ethernet Controller via `EthIf_GetCurrentTimeTuple`.

If `EthIf_GetCurrentTimeTuple` returns either `ETH_UNCERTAIN` or `ETH_INVALID` for member `timeQuality` of parameter `currentTimeTuplePtr`, the StbM shall ignore the time value `timestampClockValue` returned by `EthIf_GetCurrentTimeTuple`.]

[SWS_StbM_00539]

Status: DRAFT

Upstream requirements: [RS_TS_20070](#), [RS_TS_00008](#)

[If `CanIfGlobalTimeSupport` is set to `TRUE` for a Synchronized Time Base, then the StbM shall derive the current value of the [Virtual Local Time](#) (see [StbM_VirtualLocalTimeType](#)) from the freerunning HW counter from the corresponding CAN Controller via `CanIf_GetCurrentTime`.

If `CanIf_GetCurrentTime` returns `E_NOT_OK`, the time value returned by `CanIf_GetCurrentTime` shall be ignored.]

Note: If `CanIf_GetCurrentTime` or `EthIf_GetCurrentTimeTuple` fail, this means the corresponding [Virtual Local Time](#) is not available. Hence, related Time Bases cannot be interpolated anymore (unless a Fallback [Virtual Local Time](#) is configured and available). APIs for Time Bases, which depend on that [Virtual Local Time](#), would return `E_NOT_OK`.

Note: `EthIfGlobalTimeSupport` and `CanIfGlobalTimeSupport` may be referenced via [StbMLocalTimeHardware](#), if set to `EthTSynGlobalTimeDomain` or `CanTSynGlobalTimeDomain`, respectively.

[SWS_StbM_00437]

Upstream requirements: [RS_TS_00006](#), [RS_TS_00008](#), [RS_TS_00033](#)

[`StbM_GetCurrentVirtualLocalTime` shall return the value of the Primary [Virtual Local Time](#) of the associated Time Base.

If the [Virtual Local Time](#) could not be determined (e.g., the underlying hardware counter has not been activated yet), then `StbM_GetCurrentVirtualLocalTime` shall return `E_NOT_OK`.]

Note: `StbM_GetCurrentVirtualLocalTime` is called by the Timesync modules with an established protection against interruptions.

7.2.2 Fallback Virtual Local Time

The StbM optionally supports a Fallback `Virtual Local Time`. Like a Primary `Virtual Local Time` a Fallback `Virtual Local Time` is derived from a hardware reference clock, i.e., from an Os counter, a GPT counter, an Ethernet free running counter or any other hardware reference clock, if available. The Fallback `Virtual Local Time` is configured by the configuration container `StbMFallbackTimeClock` which is referenced by the Time Bases.

The Fallback `Virtual Local Time` supports 2 use cases,

- Monitoring the progression of the `Local Instance of the Global Time` (refer to 7.16.3 “Monitoring the progression of the Local Instance of Global Time”)
- and Time extrapolation when the Primary `Virtual Local Time` is absent (refer to 7.3.3 “Handling the Local Instance of the Global Time in absence of Primary `Virtual Local Time`”).

For Time Validation (safe time use case) the integrator must ensure that the clock source for `Virtual Local Time` and Fallback `Virtual Local Time` are not only different but independent.

[SWS_StbM_00593]

Status: DRAFT

Upstream requirements: RS_TS_00041, RS_TS_00042

[If `StbMFallbackTimeClock` is configured, then the StbM shall provide a Fallback `Virtual Local Time`.]

Note: For configuration of `StbMFallbackTimeClock` also refer to [SWS_StbM_CONSTR_00004].

[SWS_StbM_00589]

Status: DRAFT

Upstream requirements: RS_TS_00041, RS_TS_00042

[If `StbMFallbackTimeHardware` references a Gpt Channel as Fallback `Virtual Local Time` source for a Synchronized Time Base, the StbM shall derive this Fallback `Virtual Local Time` from the value of the corresponding GPT timer. The elapsed timer value shall be read via `Gpt_GetTimeElapsed`.]

[SWS_StbM_00590]

Status: DRAFT

Upstream requirements: RS_TS_00041, RS_TS_00042

[The StbM shall use the factor (`StbMFallbackTimeClockPrescaler / StbMFallbackTimeClockFrequency`) to convert the time of its local hardware reference clock to the actual time of the Fallback `Virtual Local Time` (refer to `StbM_VirtualLocalTimeType`, if the Fallback `Virtual Local Time` is derived from a GPT or OsCounter (refer to `StbMFallbackTimeHardware`)]

Rationale: A tick duration of the hardware reference clock does not necessarily have to match the resolution of the Fallback `Virtual Local Time`.

[SWS_StbM_00591]

Status: DRAFT

Upstream requirements: RS_TS_00041, RS_TS_00042

[If the range of the corresponding HW reference counter is less than that of the Fallback `Virtual Local Time` (refer to `StbM_VirtualLocalTimeType`) the StbM shall extend the range accordingly.]

Note: Depending on the HW reference clock one way of extending the range is to count overflows of the HW reference clock.

[SWS_StbM_00592]

Status: DRAFT

Upstream requirements: RS_TS_00041, RS_TS_00042

[If `EthIfGlobalTimeSupport` is set to TRUE for a Synchronized Time Base, then the StbM shall derive the current value of the Fallback `Virtual Local Time` from the freerunning HW counter from the corresponding Ethernet Controller via `EthIf_GetCurrentTimeTuple`.

If `EthIf_GetCurrentTimeTuple` returns either `ETH_UNCERTAIN` or `ETH_INVALID` for member `timeQuality` of parameter `currentTimeTuplePtr`, then the StbM shall ignore the time value `timestampClockValue` returned by `EthIf_GetCurrentTimeTuple`.]

[SWS_StbM_00594]

Status: DRAFT

Upstream requirements: RS_TS_00041, RS_TS_00042

[If `CanIfGlobalTimeSupport` is set to TRUE for a Synchronized Time Base, then the StbM shall derive the current value of the Fallback `Virtual Local Time` from the freerunning HW counter from the corresponding CAN Controller via `CanIf_GetCurrentTime`.

If `CanIf_GetCurrentTime` returns `E_NOT_OK`, the time value returned by `CanIf_GetCurrentTime` shall be ignored.]

Note: `EthIfGlobalTimeSupport` and `CanIfGlobalTimeSupport` may be referenced via `StbMFallbackTimeHardware`, if set to `EthTSynGlobalTimeDomain` or `CanTSynGlobalTimeDomain`, respectively.

Note: If `CanIf_GetCurrentTime` or `EthIf_GetCurrentTime` fail, this means the corresponding Fallback `Virtual Local Time` is not available. Hence, related Time Bases cannot be interpolated anymore and APIs for Time Bases, which depend on that Fallback `Virtual Local Time`, would return `E_NOT_OK`.

[SWS_StbM_00595]

Status: DRAFT

Upstream requirements: [RS_TS_00041](#), [RS_TS_00042](#)

[For Synchronized Time Bases `StbM_GetFallbackVirtualLocalTime` shall return the value of the Fallback `Virtual Local Time` of the associated Time Base.

If the Fallback `Virtual Local Time` could not be determined (e.g., the underlying hardware counter has not been activated yet), `StbM_GetFallbackVirtualLocalTime` shall return `E_NOT_OK`.]

7.3 Local Instance of the Global Time

The Synchronized Time-Base Manager has to progress the `Local Instance of the Global Time` (TL) of a Synchronized or Pure Local Time Base between the updates

- from the Timesync Modules (for a Time Slave of a Synchronized Time Base)
- or from the application (for a Global Time Master or a Pure Local Time Base)

Progression of TL can be done

- by SW calculation based on a free-running HW Clock (refer to chapter [7.3.1 “Progressing the Local Instance of the Global Time by a free-running HW clock”](#)).
- or by a `Disciplined HW Clock`, which is adjustable in rate and offset (refer to chapter [7.3.2 “Progressing the Local Instance of the Global Time by a disciplined HW clock”](#))

The type of progression can be configured per Time Base.

7.3.1 Progressing the Local Instance of the Global Time by a free-running HW clock

This chapter specifies how progression of the [Local Instance of the Global Time](#) (TL) is done based on a free-running HW clock.

The StbM derives TL from a on a free-running HW clock by doing a rate correction in SW according to [[SWS_StbM_00355](#)].

[SWS_StbM_00355]

Upstream requirements: [RS_TS_00018](#)

[For a Time Base if

- the Primary [Virtual Local Time](#) $TV_{Primary}$ is available (i.e., [StbM_GetCurrentVirtualLocalTime](#) returns `E_OK`)
- and a [Disciplined HW Clock](#) is not supported (i.e., [StbMDisciplinedClock](#) is not configured),

then StbM shall calculate [Local Instance of the Global Time](#) (TL) of the Time Base, based on the [Main Time Tuple](#) and the [Applied Rate](#) according to:

$$TL = TL_{Main} + r * (TV_{Primary} - TV_{Main}) \quad (7.1)$$

]

7.3.1.1 Main Time Tuple

It is obvious that the precision of the extrapolation by the StbM in [[SWS_StbM_00355](#)] depends on rounding effects and the granularity of the HW counters from which the Primary [Virtual Local Time](#) ($TV_{Primary}$) is derived.

In addition, the precision of a Time Base depends on the handling of the [Main Time Tuple](#) [TL_{Main} , TV_{Main}], i.e.,

- when and how is it interpolated by the StbM
- for a Time Slave or Time Gateway: how is it received and processed by the Timesync Modules and how it was transmitted by the Timesync Modules of the Global Time Master

The [Main Time Tuple](#) is managed by the StbM. Each time TL_{Main} is updated, TV_{Main} has to be updated as well and vice versa.

[SWS_StbM_00433]

Upstream requirements: [RS_TS_00008](#), [RS_TS_00002](#)

[If for a Time Base

- a [Disciplined HW Clock](#) is not supported (i.e., [StbMDisciplinedClock](#) is not configured)
- or a [Fallback Virtual Local Time](#) is configured (i.e., [StbMFallbackTime-Clock](#) is configured),

when

- for a Global Time Master
 - a new Global Time or a new Rate Correction value is set by the application (refer [\[SWS_StbM_00342\]](#))
 - or a Time Base gets cloned from another Time Base (see [\[SWS_StbM_00534\]](#))
- or for a Time Slave
 - a new valid [Rx Time Tuple](#) is obtained from a Timesync Module (refer [\[SWS_StbM_00440\]](#), [\[SWS_StbM_00588\]](#), [\[SWS_StbM_00400\]](#) or [\[SWS_StbM_00356\]](#))
 - or the [Offset Correction by Rate Adaption](#) interval has elapsed (see [\[SWS_StbM_00353\]](#)),

then the StbM shall update the [Main Time Tuple](#) for the Time Base.]

If there has been no update for more than 3s, the StbM may also update the [Main Time Tuple](#). The 3s interval is derived from the value range of 32 bit results (e.g., when calculating the [Virtual Local Time](#) difference, i.e., 4.29 sec) with some safety margin. This is to prevent too frequent updates of the [Main Time Tuple](#), which would lead to accumulation of rounding errors.

If requesting a Global Time by the application would always lead to an update of the Main Tuple, the high frequency of those requests would influence the precision due to the aforementioned rounding effects as well. It is therefore necessary to ensure that updates of the [Main Time Tuple](#) don't happen unnecessarily often.

So, the [Main Time Tuple](#) should not be updated:

- on every invocation of [StbM_MainFunction](#)
- every time a Global Time value is requested by [StbM_GetCurrentTime](#)

7.3.1.2 Main Time Triple

[SWS_StbM_00600]

Status: DRAFT

Upstream requirements: RS_TS_00041

[If a Fallback [Virtual Local Time](#) is enabled (i.e., [StbMFallbackTimeClock](#) is configured), then the StbM shall extend the Main Time Tuple to a [Main Time Triple](#) (TV_{Main} , $TV_{Fallback_Main}$)]

Note: The StbM maintains the [Main Time Triple](#) and derives the [Main Time Tuple](#) from the [Main Time Triple](#) when needed by reading only the relevant members of the [Main Time Triple](#).

[SWS_StbM_00602]

Status: DRAFT

Upstream requirements: RS_TS_00041, RS_TS_00042

[When StbM updates the [Main Time Tuple](#) (refer to [[SWS_StbM_00433](#)]), then the StbM shall **immediately** update the Fallback [Virtual Local Time](#) of the [Main Time Triple](#) ($TV_{Fallback_Main}$) by the current value of the Fallback [Virtual Local Time](#).

If Fallback [Virtual Local Time](#) could not be determined (e.g., the underlying hardware counter has not been activated yet), when the [Main Time Triple](#) is updated, then the StbM shall set the Fallback [Virtual Local Time](#) of the [Main Time Triple](#) ($TV_{Fallback_Main}$) to 0.]

Note: "immediately" means that the values of the [Main Time Tuple](#) and the Fallback [Virtual Local Time](#) are sampled at the very same point in time. Ideally, this is done in an atomic, i.e., a non-interruptable sequence. However, since the Fallback [Virtual Local Time](#) is first of all used for plausibility checks and a [hold-over](#) scenario, timing constraints for sampling the Fallback [Virtual Local Time](#) are not as tight as for the [Main Time Tuple](#) itself.

Note: A [Virtual Local Time](#) of zero is a reasonable choice for "stuck" or unavailable counters, because no [Virtual Local Time](#) should ever return zero in a reasonable set-up.

[SWS_StbM_00601]

Status: DRAFT

Upstream requirements: RS_TS_00041, RS_TS_00042

[If Primary [Virtual Local Time](#) could not be determined (e.g., the underlying hardware counter has not been activated yet), when the [Main Time Triple](#) is updated then the StbM shall set the Primary [Virtual Local Time](#) of the [Main Time Tuple](#) (TV_{Main}) to 0.]

7.3.2 Progressing the Local Instance of the Global Time by a disciplined HW clock

This chapter specifies progression of the [Local Instance of the Global Time](#) (TL) if supported by a [Disciplined HW Clock](#), i.e., a HW clock that is incremented by the underlying hardware and which can be adjusted in offset and rate.

A [Disciplined HW Clock](#) is currently only supported for Synchronized and Pure Local Time Bases which reference an underlying Ethernet HW clock (PHC).

The StbM will use the following 2 APIs to adjust the value of the [Disciplined HW Clock](#) while it is progressing:

- `EthIf_SetPhcTime` for setting the absolute value, e.g.,
 - on Global Time Master side after a reset of the ECU when restoring the time from NvM [[SWS_StbM_00578](#)]
 - or on Global Time Master side when `StbM_SetGlobalTime` or `StbM_UpdateGlobalTime` are being called [[SWS_StbM_00579](#)]
- `EthIf_SetPhcCorrection` for smoothly adapting the rate of the HW clock and correcting smaller offsets according to [[SWS_StbM_00586](#)]
 - on Time Slave side
 - * after receiving a new `Rx Time Tuple` (TG_{Rx}) via `StbM_BusSetGlobalTime`
 - * or when the adaption interval for Offset Correction by Rate Adaption has elapsed
 - or on Global Time Master side when `StbM_SetRateCorrection` is being called.

The StbM will use following 2 APIs to read the value of the [Disciplined HW Clock](#) while it is progressing:

- `EthIf_GetPhcTime` to derive the current [Local Instance of the Global Time](#) (TL)
- `EthIf_GetCurrentTimeTuple` to derive the current Time Tuple [TL, TV], i.e., the current Local Instance of the Global Time (TL) and the correlated / cross-timestamped [Virtual Local Time](#) (TV)

[SWS_StbM_00574]

Status: DRAFT

Upstream requirements: [RS_TS_00008](#), [RS_TS_00002](#)

[If

- a [Disciplined HW Clock](#) is supported (`StbMDisciplinedClock` is configured),

- and StbM uses the same HW clock unit for
 - for deriving the `Virtual Local Time` (`StbMLocalTimeHardware->EthTSynGlobalTimeDomain->EthIfClkUnit`)
 - and for deriving the `Local Instance of the Global Time` (`StbMDisciplinedClockHardwareRef->EthTSynGlobalTimeDomain->EthIfClkUnit`),

then StbM shall derive the current Time Tuple [TL, TV] by calling `EthIf_GetCurrentTimeTuple`

- to read the `Local Instance of the Global Time` (TL) from the member `disciplinedClockValue`
- and to read the `Virtual Local Time` (TV) from the member `timestampClockValue`

of parameter `currentTimeTuplePtr`]

[SWS_StbM_00575]

Status: DRAFT

Upstream requirements: [RS_TS_00008](#), [RS_TS_00002](#)

[If

- a `Disciplined HW Clock` is supported (`StbMDisciplinedClock` is configured),
- and StbM does not use the same HW clock unit for
 - for deriving the `Virtual Local Time` (`StbMLocalTimeHardware->EthTSynGlobalTimeDomain->EthIfClkUnit`)
 - and for deriving the `Local Instance of the Global Time` (`StbMDisciplinedClockHardwareRef->EthTSynGlobalTimeDomain->EthIfClkUnit`),

then StbM shall derive the current Time Tuple [TL, TV] by consecutive calls to

- `EthIf_GetPhcTime` to retrieve the Global Time of the Tuple (TL) from the parameter `timeStampPtr`
- and `StbM_GetCurrentVirtualLocalTime` to retrieve the `Virtual Local Time` of the Tuple (TV).

The sequence of the 2 calls, `StbM_GetCurrentVirtualLocalTime` and `EthIf_GetPhcTime`, shall not be interrupted.]

7.3.3 Handling the Local Instance of the Global Time in absence of Primary Virtual Local Time

If the **Primary Virtual Local Time** is available, the **Local Instance of the Global Time** of a Time Base will be derived based on the **Primary Virtual Local Time** of that Time Base (refer to chapter 7.2.1 “**Primary Virtual Local Time**”).

If the **Primary Virtual Local Time** of a Time Base fails, the StbM can still extrapolate the **Local Instance of the Global Time** of the Time Base using a **Fallback Virtual Local Time**, if configured for the Time Base. The **Fallback Virtual Local Time** provides a **hold-over** capability for extrapolating the **Local Instance of the Global Time** locally.

"Locally" means, that the value of the **Local Instance of the Global Time** is only provided to the local application upon request, but is not distributed on the network. This is because the Timesync modules currently still call the `StbM_GetCurrentVirtualLocalTime`. Since that call will return `E_NOT_OK`, when the **Primary Virtual Local Time** is not available, the Timesync module cannot properly process any transmit or receive request.

Figure 7.3 illustrates how the **Local Instance of the Global Time** of a Time Base is derived, if the **Primary Virtual Local Time** fails and afterwards recovers from an outage.

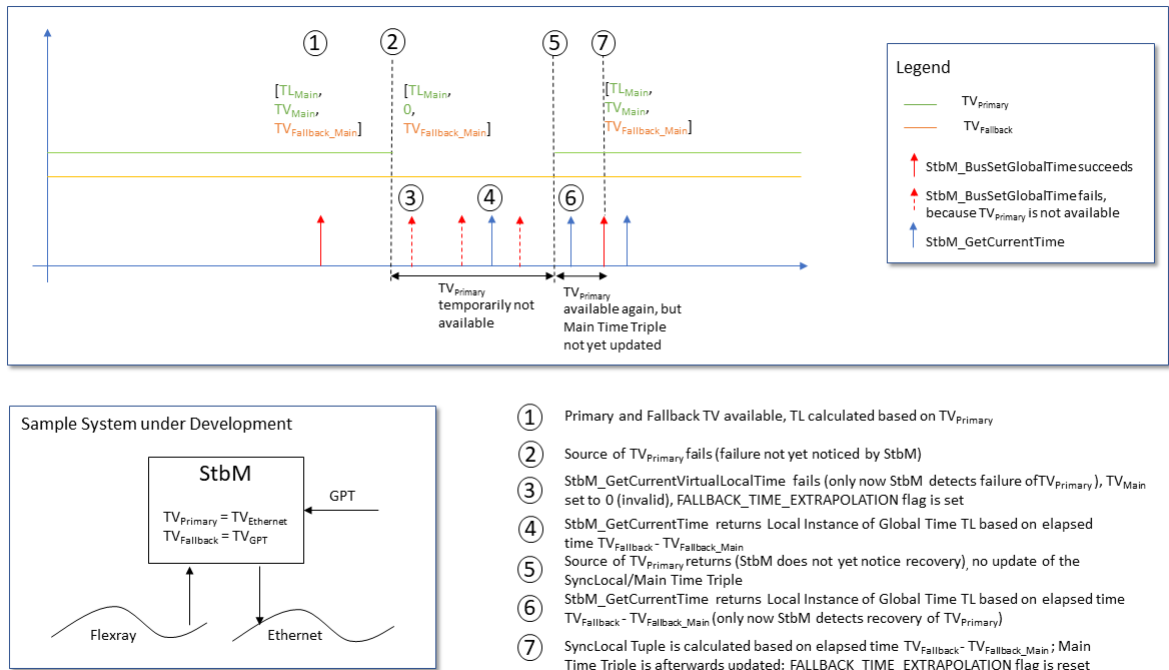


Figure 7.3: Temporary Outage Of Primary Virtual Local Time

[SWS_StbM_00606]*Status:* DRAFT*Upstream requirements:* [RS_TS_00040](#)

[For a Synchronized or Pure Local Time Base if

- the Primary [Virtual Local Time](#) is not available (i.e., [StbM_GetCurrentVirtualLocalTime](#) returns `E_NOT_OK`)
- and the Fallback [Virtual Local Time](#) is configured (i.e., [StbMFallbackTimeClock](#) is configured)
- and the Fallback [Virtual Local Time](#) is available (refer to [\[SWS_StbM_00595\]](#)),

then the StbM shall calculate [Local Instance of the Global Time](#) (TL) of the Time Base based on the [Main Time Triple](#) without applying rate correction according to the following formula:

$$TL = TL_{Main} + (TV_{Fallback} - TV_{Fallback_Main}) \quad (7.2)$$

]

Rationale: Extrapolation with the Fallback [Virtual Local Time](#) currently does not support [Rate Correction](#) to keep things simpler, because no [Rate Correction](#) needs to be measured and calculated (refer to chapter 7.11 “[Time Correction](#)”). For plausibility checks and [hold-over](#) operation such a simplified method is considered good enough.

[SWS_StbM_00607]*Status:* DRAFT*Upstream requirements:* [RS_TS_00042](#)

[For a Synchronized or Pure Local Time Base if

- the Fallback [Virtual Local Time](#) is configured (i.e., [StbMFallbackTimeClock](#) is configured)
- and the Primary [Virtual Local Time](#) is not available
- and the Fallback [Virtual Local Time](#) is available,

when the StbM is required to retrieve the [Virtual Local Time](#), then the StbM shall set the `timeBaseStatus` bit [FALLBACK_TIME_EXTRAPOLATION](#) of the Time Base]

[SWS_StbM_00611]*Status:* DRAFT*Upstream requirements:* [RS_TS_00043](#)

[For a Synchronized or Pure Local Time Base if

- the Fallback `Virtual Local Time` is configured (i.e., `StbMFallbackTime-Clock` is configured)
- and the Primary `Virtual Local Time` has become available again (e.g. after an outage)
- and the Fallback `Virtual Local Time` is available,

when the StbM has updated the `Main Time Triple` (refer to [SWS_StbM_00433]), then the StbM

- shall resume the extrapolation of the Time Base based on the Primary `Virtual Local Time`
- and reset the `timeBaseStatus` bit `FALLBACK_TIME_EXTRAPOLATION` of the Time Base

]

[SWS_StbM_00609]

Status: DRAFT

Upstream requirements: RS_TS_00043

[For a Time Slave of a Synchronized Time Base if

- the Fallback `Virtual Local Time` is configured (i.e., `StbMFallbackTime-Clock` is configured)
- and the Primary `Virtual Local Time` has become available again (e.g. after an outage)
- and the Fallback `Virtual Local Time` is available,

when `StbM_BusSetGlobalTime` is called, then the StbM shall derive the `SyncLocal Time Tuple` as follows:

- calculate TL_{Sync} according to [SWS_StbM_00606]
- and set TV_{Sync} to $TV_{Primary}$

before the `Main Time Triple` is updated.]

Rationale: The `SyncLocal Time Tuple` is required to be calculated for Time Leap detection. It is calculated before the `Main Time Tuple` because if `Offset Correction by Rate Adaption` is enabled the new `Main Time Tuple` depends on the `SyncLocal Time Tuple` (refer to chapter 7.11.2).

Note: Time leaps are expected after a recovery of the Primary `Virtual Local Time` as the `SyncLocal Time Tuple` is calculated in [SWS_StbM_00609]

- based on a TL_{Main} which has not been updated for longer time
- and without rate correction.

[SWS_StbM_00610]

Status: DRAFT

Upstream requirements: [RS_TS_00043](#)

[For a Synchronized or Pure Local Time Base if

- the Fallback [Virtual Local Time](#) is configured (i.e., [StbMFallbackTime-Clock](#) is configured)
- and the Primary [Virtual Local Time](#) is not available
- and the Fallback [Virtual Local Time](#) has become available again (e.g. after an outage),

when the [Virtual Local Time](#) is to be derived, then the StbM shall resume the fallback extrapolation of the [Local Instance of the Global Time](#) of the Time Base according to [\[SWS_StbM_00606\]](#).]

7.4 Synchronized Time Bases

[SWS_StbM_00180]

Upstream requirements: [RS_TS_00008](#), [RS_TS_00002](#)

[After initialization the StbM shall maintain the Local Time of each Time Base autonomously via a hardware reference clock (referenced by [StbMLocalTimeClock](#)).]

Note: While no Global Time Base value has yet been set/received ([GLOBAL_TIME_BASE](#) bit is not yet set), the StbM shall maintain the Local Time of each Time Base (i.e., progress the time) starting at the value restored from [NvM](#) or at value 0 (depending on setting of [StbMStoreTimebaseNonVolatile](#)).

Note: Progressing the time means that the Virtual Local Time as part of the Main Time Tuple needs to be retrieved once the Global Time part of the Main Time Tuple was either set to 0 or to the value restored from [NvM](#).

[SWS_StbM_00173]

Upstream requirements: [RS_TS_00005](#), [RS_TS_00006](#), [RS_TS_00014](#)

[For Synchronized Time Bases if

- the Primary [Virtual Local Time](#) is available and valid,
- or Fallback [Virtual Local Time](#) is available (i.e., [StbMFallbackTime-Clock](#) is configured) and valid

then [StbM_GetCurrentTime](#) shall return

- [E_OK](#).

- and the current Time Tuple [TL; TV],
- and the current status of the Time Base `timeBaseStatus`,
- and the User Data.

]

[SWS_StbM_00434]*Upstream requirements:* [RS_TS_00005](#), [RS_TS_00006](#), [RS_TS_00014](#)

[For Synchronized Time Bases if

- the Primary [Virtual Local Time](#) is not available,
- and Fallback [Virtual Local Time](#) is not configured (i.e., [StbMFallbackTimeClock](#) is configured) or not valid

then [StbM_GetCurrentTime](#) shall return `E_NOT_OK`.]**[SWS_StbM_00436]***Upstream requirements:* [RS_TS_00005](#), [RS_TS_00006](#), [RS_TS_00014](#)[For Synchronized Time Bases if the Primary [Virtual Local Time](#) is available then StbM shall set the current Time Tuple [TL, TV] with

- TV set to the Primary [Virtual Local Time](#)
- and TL set to the corresponding [Local Instance of the Global Time](#)

]

[SWS_StbM_00608]*Status:* DRAFT*Upstream requirements:* [RS_TS_00043](#)

[For Synchronized Time Bases If

- the Primary [Virtual Local Time](#) is not available
- and Fallback [Virtual Local Time](#) is configured (i.e., [StbMFallbackTimeClock](#) is configured)
- and the Fallback [Virtual Local Time](#) is available,

then the StbM shall set the current Time Tuple [TL, TV] with

- TV set to the Fallback [Virtual Local Time](#),
- and TL set to the corresponding [Fallback Local Instance of the Global Time](#) calculated according to [\[SWS_StbM_00606\]](#)

]

7.4.1 Global Time Master

[SWS_StbM_00342]

Upstream requirements: RS_TS_00010, RS_TS_00002

[For a Global Time Master of a Synchronized Time Bases,
if a *Disciplined HW Clock* is not supported (i.e., *StbMDisciplinedClock* is not configured),
when *StbM_SetGlobalTime* or *StbM_UpdateGlobalTime* is called,
then the StbM shall update the *Main Time Tuple* [TL_{Main} , TV_{Main}] of the corresponding Synchronized Time Base.

StbM shall set

- TL_{Main} to the value of the parameter *timeStamp* of *StbM_SetGlobalTime* or *StbM_UpdateGlobalTime*, respectively
- and TV_{Main} to the value of the *Primary Virtual Local Time* as sampled **immediately** after entering *StbM_SetGlobalTime* or *StbM_UpdateGlobalTime*, respectively

]

Note: Sampling the *Virtual Local Time* **immediately** after entering *StbM_SetGlobalTime* or *StbM_UpdateGlobalTime* improves the precision. In order to further improve precision it may be beneficial for applications to call *StbM_SetGlobalTime* or *StbM_UpdateGlobalTime* with locked interrupts.

[SWS_StbM_00579]

Status: DRAFT

Upstream requirements: RS_TS_00010, RS_TS_00002

[For a Global Time Master of a Synchronized Time Bases,
if a *Disciplined HW Clock* is supported (*StbMDisciplinedClock* is configured),
when *StbM_SetGlobalTime* or *StbM_UpdateGlobalTime* is called,
then the StbM shall adjust the value of the *Disciplined HW Clock* to the value as set by parameter *timeStamp* of *StbM_SetGlobalTime* or *StbM_UpdateGlobalTime* by calling *EthIf_SetPhcTime*.
StbM shall set parameter *timeStampPtr* of *EthIf_SetPhcTime* to the value of *timeStamp* of *StbM_SetGlobalTime* or *StbM_UpdateGlobalTime*, respectively.]

[SWS_StbM_00516]

Upstream requirements: [RS_TS_20069](#)

[For a Time Master of a Synchronized Time Base on invocation of `StbM_SetBusProtocolParam`, the StbM shall forward the values provided in argument `protocolParam` by calling `EthTSyn_SetProtocolParam`.

If

- the corresponding Time Base is not mapped to Ethernet or
- member `protocolType` of argument `protocolParam` is not set to `STBM_TIMESYNC_ETHERNET`

`StbM_SetBusProtocolParam` shall return `E_NOT_OK`.]

[SWS_StbM_00172]

Upstream requirements: [RS_TS_00024](#)

[For each Synchronized Time Base configured

- for a Time Master
- and to be stored non-volatile (i.e., `StbMStoreTimebaseNonVolatile == STORAGE`)

the StbM shall cyclically, with a cycle interval of `StbMCyclicBackupInterval`, store a backup time to `NvM`.

The StbM shall calculate the value of the backup time as:

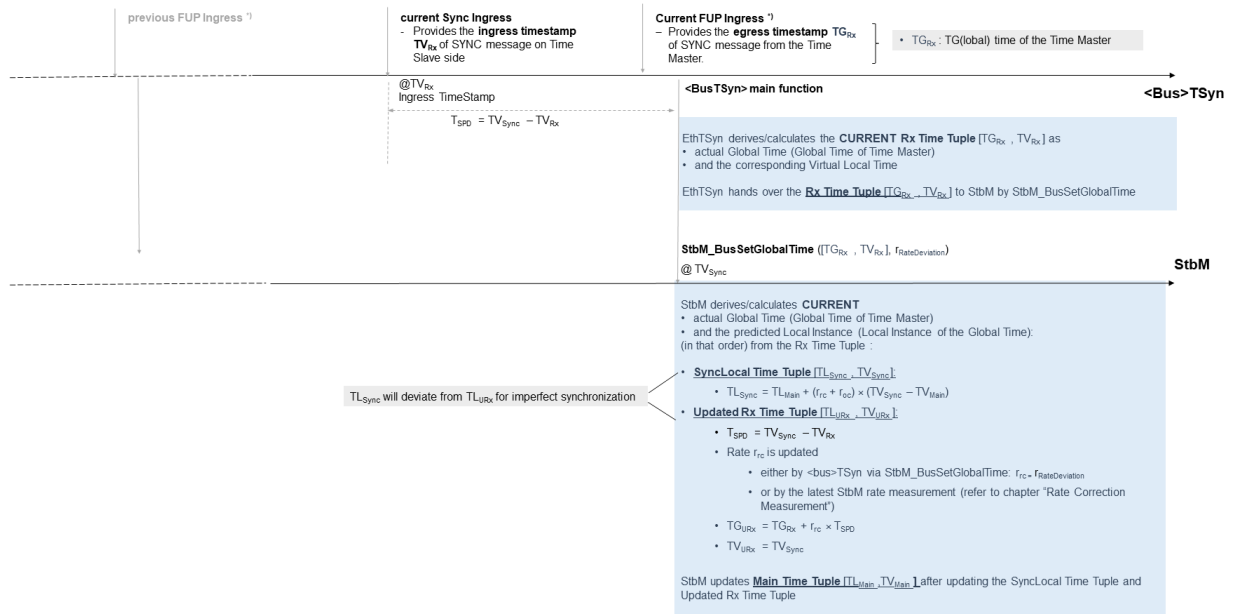
- current time of the Time Base + `StbMCyclicBackupInterval`.

The initial value of the backup time in the `NvM` shall be set to 0.]

Rationale: By adding the offset value `StbMCyclicBackupInterval` to the current time, when storing the backup value to `NvM`, it is ensured that the Global Time increases (strictly) monotonously even in the case of a reset, when the backup time is restored from `NvM`.

7.4.2 Time Slave

Figure 7.4 illustrates the steps to update a Synchronized Time Base, when a new Timesync message is received from a Time Master.



^{*)} Follow-up messages are only supported by CanTSyn and EthTSyn (2-step synchronization). FRTSyn derives the Rx Time Tuple directly from the SYNC message

Figure 7.4: Rx Time Tuple Processing Overview

Once a new Time Tuple (denoted as **Rx Time Tuple** $[TG_{Rx}, TV_{Rx}]$) is derived by the Timesync Module, the Timesync Module forwards this Time Tuple via `StbM_BusSetGlobalTime` to the StbM for further processing.

The following steps are also illustrated in the sequence diagrams [Figure 9.5](#) and [Figure 9.6](#). Upon invocation of `StbM_BusSetGlobalTime`, the StbM records the current value of the **Virtual Local Time** as TV_{Sync} and calculates two new Time Tuples:

- the **SyncLocal Time Tuple** $[TL_{Sync}, TV_{Sync}]$ which is the Time Tuple at that point in time (TV_{Sync}). The **SyncLocal Time Tuple** does not consider the current update of the **Rx Time Tuple** $[TG_{Rx}, TV_{Rx}]$ (refer to chapter 7.4.2.1 "Calculation of the SyncLocal Time Tuple").

Hence, the **SyncLocal Time Tuple** is a **prediction of the Global Time** by the StbM at TV_{Sync} **based on 'previous' update of the Global Time** TG_{Rx} received from the Global Time Master and the calculated rate and offset correction.

- the **Updated Rx Time Tuple** $[TG_{URx}, TV_{URx}]$ which is calculated at the very same point in time as the **SyncLocal Time Tuple**, i.e., $TV_{URx} = TV_{Sync}$. It is derived from the **Rx Time Tuple** by considering the delay from the ingress of the SYNC message at TV_{Rx} until `StbM_BusSetGlobalTime` is actually being called at TV_{Sync} . This delay $TV_{Sync} - TV_{Rx}$ is denoted **SYNC Ingress Processing Delay** (T_{SPD}) (refer to chapter 7.4.2.2 "Calculation of the Updated Rx Time Tuple").

The *SYNC Ingress Processing Delay* needs to be rate corrected to improve the precision when calculating the *Updated Rx Time Tuple* ($[TG_{URx}]$, TV_{URx}).

Hence, the *Updated Rx Time Tuple* is **derived directly from the 'current', i.e., latest update of the Global Time** TG_{Rx} received from the Global Time Master.

7.4.2.1 Calculation of the SyncLocal Time Tuple

[SWS_StbM_00438]

Upstream requirements: [RS_TS_00007](#), [RS_TS_00009](#)

[For a Synchronized Time Base

if StbM does not use a *Disciplined HW Clock* to maintain the *Local Instance of the Global Time* (i.e., container *StbMDisciplinedClock* is configured),

when *StbM_BusSetGlobalTime* is called,

then the StbM derive the *SyncLocal Time Tuple* $[TL_{Sync}, TV_{Sync}]$

- by retrieving TV_{Sync} by calling *StbM_GetCurrentVirtualLocalTime*
- and calculating TL_{Sync} according to [\[SWS_StbM_00355\]](#).

]

[SWS_StbM_00573]

Status: DRAFT

Upstream requirements: [RS_TS_00007](#), [RS_TS_00009](#)

[For a Synchronized Time Base,

if StbM uses a *Disciplined HW Clock* to maintain the *Local Instance of the Global Time* (i.e., container *StbMDisciplinedClock* is configured),

when *StbM_BusSetGlobalTime* is called,

then the StbM shall retrieve the current *SyncLocal Time Tuple* $[TL_{Sync}, TV_{Sync}]$ from the underlying *Disciplined HW Clock* according to [\[SWS_StbM_00574\]](#) or [\[SWS_StbM_00575\]](#), respectively.]

7.4.2.2 Calculation of the Updated Rx Time Tuple

[SWS_StbM_00529]

Upstream requirements: [RS_TS_00007](#), [RS_TS_00009](#)

[For a Synchronized Time Base

when `StbM_BusSetGlobalTime` is called,

then the StbM shall calculate TG_{URx} of the `Updated Rx Time Tuple` of a Synchronized Time Base as follows:

- calculate the `SYNC Ingress Processing Delay` as time interval $T_{SPD} = TV_{Sync} - TV_{Rx}$ with
 - TV_{Sync} as derived for the `SyncLocal Time Tuple` according to [\[SWS_StbM_00573\]](#)
 - TV_{Rx} as received via `StbM_BusSetGlobalTime`
- apply the current rate correction r_{rc}
 - either as received by the current `StbM_BusSetGlobalTime` call according to [\[SWS_StbM_00576\]](#) or [\[SWS_StbM_00577\]](#)
 - or as calculated according to [\[SWS_StbM_00361\]](#)
 to the `SYNC Ingress Processing Delay`
- calculate TG_{URx} by adding the rate corrected `SYNC Ingress Processing Delay` to the Global Time of the `Rx Time Tuple` TG_{Rx} , i.e., $TG_{URx} = TG_{Rx} + r_{rc} * T_{SPD}$

The StbM shall calculate the `Updated Rx Time Tuple` at the same point in time as the `SyncLocal Time Tuple`, i.e., at $TV_{URx} = TV_{Sync}$

With

- TV_{Sync} as derived by [\[SWS_StbM_00438\]](#) or [\[SWS_StbM_00573\]](#), respectively

]

7.4.2.3 General

[SWS_StbM_00179]

Upstream requirements: [RS_TS_00007](#), [RS_TS_00009](#)

[For a Synchronized Time Base each invocation of `StbM_BusSetGlobalTime` shall update the User Data of the Time Base with the data provided by input parameter `userDataPtr`.]

[SWS_StbM_00517]

Upstream requirements: [RS_TS_20069](#)

[For Synchronized Time Bases on invocation of [StbM_GetBusProtocolParam](#), the StbM shall read the structure values referenced by argument [protocolParam](#) by calling [EthTSyn_GetProtocolParam](#), if member [protocolType](#) of argument [protocolParam](#) is set to [STBM_TIMESYNC_ETHERNET](#).

If

- the corresponding Time Base is not mapped to Ethernet or
- member [protocolType](#) of argument [protocolParam](#) is not set to [STBM_TIMESYNC_ETHERNET](#)

[StbM_GetBusProtocolParam](#) shall return [E_NOT_OK](#).]

7.5 Pure Local Time Bases

A Pure Local Time Base will only locally be set and read. A Pure Local Time Base behaves like a Synchronized Time Base since it progresses in time, however it is not synchronized via Timesync modules. So, only a subset of APIs is supported by Pure Local Time Base.

[SWS_StbM_00413]

Upstream requirements: [RS_TS_00008](#), [RS_TS_00002](#)

[After initialization the StbM shall maintain the Time of each Pure Local Time Base autonomously via a hardware reference clock (referenced by [StbMLocalTimeClock](#)).]

Note: While no Time Base value has yet been set ([GLOBAL_TIME_BASE](#) bit is not yet set), the StbM shall maintain the time value of each Pure Local Time Base (i.e., progress the time) starting at the value 0.

[SWS_StbM_00398]

Upstream requirements: [RS_TS_00015](#)

[For Pure Local Time Bases [StbM_GetCurrentTime](#) shall return the User Data as set by [StbM_SetGlobalTime](#), [StbM_UpdateGlobalTime](#) or [StbM_SetUserData](#).]

[SWS_StbM_00399]

Upstream requirements: [RS_TS_00009](#)

[For Pure Local Time Bases the StbM shall

- allow to change
 - the `GLOBAL_TIME_BASE` bit (according to [SWS_StbM_00571] and [SWS_StbM_00572])
 - and the `RATE_CORRECTED` bit (according to [SWS_StbM_00570], [SWS_StbM_00535])
- and keep all other bits of the Time Base status `timeBaseStatus` at their initial state 0.

]

[SWS_StbM_00571]*Upstream requirements:* [RS_TS_00009](#)

[For Pure Local Time Bases upon a valid invocation of `StbM_SetGlobalTime` or `StbM_UpdateGlobalTime` the StbM shall set the `GLOBAL_TIME_BASE` bit of the status of the Time Base for the corresponding Time Base to 1.]

[SWS_StbM_00572]*Upstream requirements:* [RS_TS_00009](#)

[For Pure Local Time Bases if a clone request is processed, the StbM shall copy the `GLOBAL_TIME_BASE` bit of the status of the Source Time Base to the `GLOBAL_TIME_BASE` bit of the corresponding Destination Time Base.]

Note: Refer to [SWS_StbM_00532] and to [SWS_StbM_00530] for when exactly a `DEFERRED` and `IMMEDIATE` clone request is processed.

Note: Assumption is, that APIs `StbM_SetGlobalTime` or `StbM_UpdateGlobalTime` are not used for Pure Local Time Bases, which are the destination of a clone request to avoid conflicting settings. With that, the `GLOBAL_TIME_BASE` bit cannot be set to 1 independent of a clone request.

7.6 Synchronization State

[SWS_StbM_00262]*Upstream requirements:* [RS_TS_00005](#), [RS_TS_00021](#)

[For Synchronized and Pure Local Time Bases `StbM_GetTimeBaseStatus` shall return the status of the corresponding Time Base via `syncTimeBaseStatus`.]

Note: Pure Local Time Bases support only a limited set of status flags according to [SWS_StbM_00399].

7.6.1 Global Time Master

[SWS_StbM_00181]

Upstream requirements: [RS_TS_00009](#)

[On a valid invocation of `StbM_SetGlobalTime` and `StbM_UpdateGlobalTime` the StbM shall set the `GLOBAL_TIME_BASE` bit within `timeBaseStatus` of the corresponding Time Base.]

7.6.2 Time Slaves

Usually, a Time Slave starts its local Time Base from 0. So, after initialization the 1st check against `StbMTimeLeapFutureThreshold` / `StbMTimeLeapPastThreshold` would most likely always fail and the `TIMELEAP_FUTURE` / `TIMELEAP_PAST` bit would be always set. To avoid this, threshold monitoring will start only after a first valid Time Base value has been received.

The state machine below illustrates the detection for a time leap into the future. The behavior for a detection of a time leap into the past would look fully symmetrically.

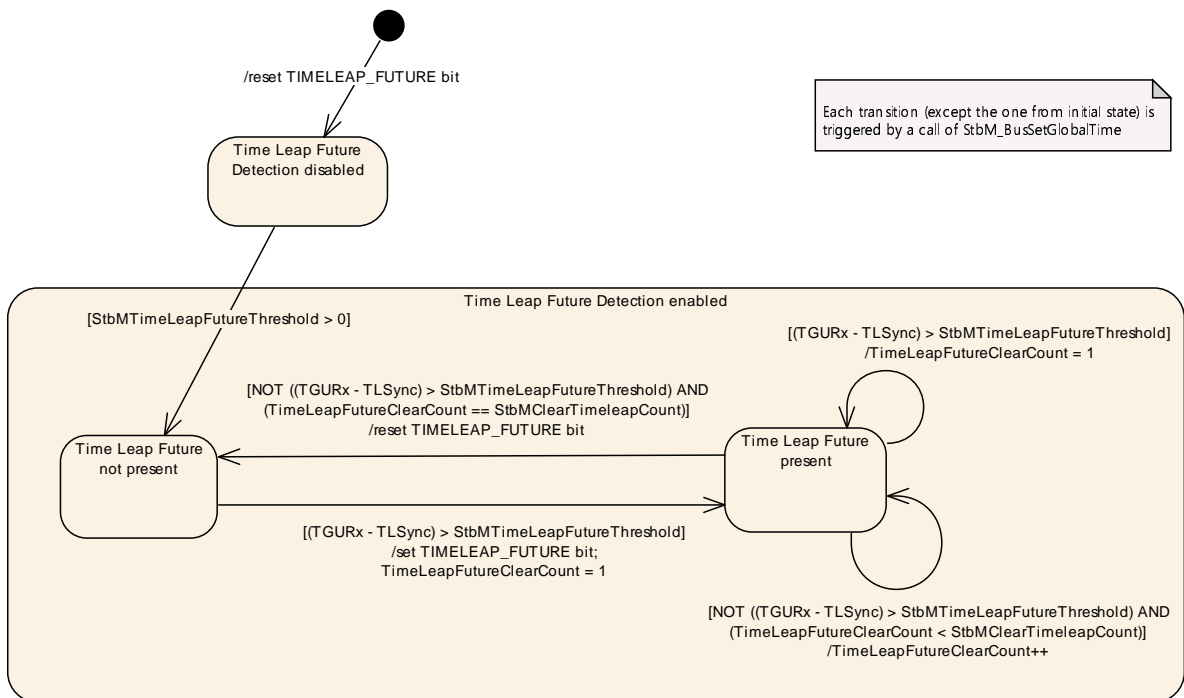


Figure 7.5: "Time Leap Future" Detection State Machine

[SWS_StbM_00182]

Upstream requirements: [RS_TS_00009](#)

[For Synchronized Time Bases for which the StbM is configured as Time Slave or Time Gateway, an invocation of [StbM_BusSetGlobalTime](#) shall check, if the Global Time difference between the Updated Rx Time (i.e., the updated Time Base value) and the Synclocal Time (i.e., the current Time Base value) exceeds the configured threshold of [StbMTimeLeapFutureThreshold](#), i.e., $TG_{URx} - TL_{Sync} > StbMTimeLeapFutureThreshold$, if at least one Time Base value has been successfully received before.

With:

- TL_{Sync} = Global Time part of the Synclocal Time Tuple
- TG_{URx} = Global Time part of the Updated Rx Time Tuple

In case the threshold is exceeded the StbM shall set the [TIMELEAP_FUTURE](#) bit within `timeBaseStatus` of the Time Base.

If the next [StbMClearTimeleapCount](#) updates are within the threshold of [StbMTimeLeapFutureThreshold](#) the StbM shall clear the [TIMELEAP_FUTURE](#) bit within `timeBaseStatus` of the Time Base.

A threshold of 0 shall deactivate this check.]

[SWS_StbM_00305]

Upstream requirements: [RS_TS_00009](#)

[For Synchronized Time Bases for which the StbM is configured as Time Slave or Time Gateway, an invocation of [StbM_BusSetGlobalTime](#) shall check, if the Global Time difference between the Synclocal Time (i.e., the current Time Base value) and the Received Time (i.e., the updated Time Base value) exceeds the configured threshold of [StbMTimeLeapPastThreshold](#), i.e., $TL_{Sync} - TG_{URx} > StbMTimeLeapPastThreshold$, if at least one Time Base value has been successfully received before.

With:

- TL_{Sync} = Global Time part of the Synclocal Time Tuple
- TG_{URx} = Global Time part of the Updated Rx Time Tuple

In case the threshold is exceeded the StbM shall set the [TIMELEAP_PAST](#) bit within `timeBaseStatus` of the Time Base.

If the next [StbMClearTimeleapCount](#) updates are within the threshold of [StbMTimeLeapPastThreshold](#) the StbM shall clear the [TIMELEAP_PAST](#) bit within `timeBaseStatus` of the Time Base.

A threshold of 0 shall deactivate this check.]

Note: After a longer timeout a time leap is likely to be detected (either `StbMTimeLeapFutureThreshold` or `StbMTimeLeapPastThreshold` is exceeded), although the time drift was within the acceptable range. A time leap could also occur if a Time Slave continues operating while a Time Master performs a restart.

Additional measures could be taken on application level to cope with those situations.

Note: If set, a `TIMELEAP_FUTURE/TIMELEAP_PAST` bit remains set while a timeout is active (i.e., while the `TIMEOUT` bit is set) and also beyond, if `StbMClearTimeLeapCount` updates within the threshold of `StbMTimeLeapFutureThreshold/StbMTimeLeapPastThreshold` have not yet happened.

[SWS_StbM_00425]

Upstream requirements: [RS_TS_00009](#)

[For Synchronized Time Bases for which the StbM is configured as Time Slave or Time Gateway `StbM_GetTimeLeap()` shall return the Global Time difference between the Updated Rx Time and the Synclocal Time, i.e., $TG_{URx} - TL_{Sync}$, which is calculated upon each, except the very first, valid invocation of `StbM_BusSetGlobalTime` for the corresponding Time Base.

With

- TL_{Sync} = Global Time part of the Synclocal Time Tuple
- TG_{URx} = Global Time part of the Updated Rx Time Tuple

If the calculated time difference exceeds the value range of the `timeJump` parameter of `StbM_GetTimeLeap` the returned time difference shall be limited to either the maximum negative or the maximum positive value of the type of `timeJump` (refer to `StbM_TimeDiffType`).

`StbM_GetTimeLeap` shall return `E_NOT_OK` until the second valid invocation of `StbM_BusSetGlobalTime` for the corresponding Time Base.]

[SWS_StbM_00183]

Upstream requirements: [RS_TS_00025](#), [RS_TS_00009](#)

[For Synchronized Time Bases for which the StbM is configured as Time Slave or Time Gateway, the StbM shall observe the timeout `StbMSyncLossTimeout`. The timeout shall be measured from last invocation of `StbM_BusSetGlobalTime`.

If the timeout occurs, the StbM shall set the `TIMEOUT` bit within `timeBaseStatus` of the Time Base.

An invocation of `StbM_BusSetGlobalTime` shall clear the `TIMEOUT` bit.]

[SWS_StbM_00540]

Upstream requirements: [RS_TS_00025](#), [RS_TS_00009](#)

[The StbM shall check for a timeout of a Time Base within [StbM_MainFunction](#)

- either based on the Virtual Local Time
- or by counting the main function calls

]

[SWS_StbM_00187]

Upstream requirements: [RS_TS_00025](#), [RS_TS_00009](#)

[For Synchronized Time Bases for which the StbM is configured as Time Gateway, the StbM shall set the [SYNC_TO_GATEWAY](#) bit within [timeBaseStatus](#) of the Time Base when a timeout occurs (refer to [\[SWS_StbM_00183\]](#)).]

[SWS_StbM_00184]

Upstream requirements: [RS_TS_00009](#)

[Every invocation of [StbM_BusSetGlobalTime](#) shall set the [SYNC_TO_GATEWAY](#) bit within [timeBaseStatus](#) of the Time Base to the value of the [SYNC_TO_GATEWAY](#) bit within [timeBaseStatus](#) of the [timeTuplePtr](#) argument passed to [StbM_BusSetGlobalTime](#).]

[SWS_StbM_00185]

Upstream requirements: [RS_TS_00009](#)

[For Synchronized Time Bases for which the StbM is configured as Time Slave or Time Gateway, an invocation of [StbM_BusSetGlobalTime](#) shall set the [GLOBAL_TIME_BASE](#) bit within [timeBaseStatus](#) of the Time Base. Once set, the bit is never cleared.]

7.7 User Data

User Data is part of each Global Time Base. User Data are set by the system-wide Time Master of each Time Base and distributed as part of the Timesync messages.

User Data are used for application specific purposes, e.g.

- for debugging purposes during integration
- and/or to characterize the Time Base, e.g.,
 - regarding the quality of the underlying clock source

- or regarding the progress of time (e.g., if Global Time has been restored from non-volatile memory after reset).

Note, that the status information, if contained in the User Data, is from the Time Master's point of view, whereas the `timeBaseStatus` in the `StbM` (`StbM_TimeBaseStatusType`) refers to the status of the Time Base from a Time Slave's point of view..

User Data consist of up to three bytes. Due to the frame format of various Timesync messages it is not possible to transmit all three bytes on every bus system. It is the responsibility of the system designer to only use those User Data bytes that can be distributed inside the vehicle network.

[SWS_StbM_00381]

Upstream requirements: [RS_TS_00015](#)

[All functions that are setting User Data shall only set as many User Data bytes as defined within the `userDataLength` element of the `StbM_UserDataType` structure.

If `userDataLength` is equal to 0, no User Data bytes shall be set. User Data bytes that are not set shall remain at their previous value.]

7.8 Startup behavior

This chapter describes the actions, which shall be performed during `StbM_Init`.

`StbM_Init` shall establish the initial state of the module to prepare the module for the actual functionality of providing Global Time Bases to the customers.

7.8.1 Preconditions

Required basic software modules for the Synchronized Time-Base Manager must be available (running) before the Synchronized Time-Base Manager accesses them.

Details of `StbM` initialization are considered implementation specific.

If `StbM` relies on the GPT driver, assumption is, that GPT is initialized by GPT driver before `StbM_Init`. `StbM_Init` starts the GPT timer, which is selected by `StbM_LocalTimeHardware` or `StbM_FallbackTimeHardware` and configured in `GPT_CH_MODE_CONTINUOUS` mode with `GptChannelTickValueMax` as target time. Timer overflows are counted by the notification function `Gpt_Notification_<channel>` for updating the Virtual Local Time. This timer is not stopped/reconfigured before ECU shutdown.

7.8.2 Initialization

[SWS_StbM_00170]

Upstream requirements: [RS_TS_00003](#)

[On invocation of `StbM_Init` each configured Time Base (refer to `StbMSynchronizedTimeBase`) shall be initialized with zero and its synchronization status `timeBaseStatus` shall be set to 0.]

[SWS_StbM_00345]

Upstream requirements: [RS_TS_00016](#)

[For each Time Base the StbM shall initialize the corresponding event status `NotificationEvents` with 0.]

[SWS_StbM_00344]

Upstream requirements: [RS_TS_00011](#)

[For each Time Base the StbM shall initialize the corresponding update counter `timeBaseUpdateCounter` with 0.]

[SWS_StbM_00171]

Upstream requirements: [RS_TS_00004](#)

[For each Synchronized Time Base configured

- for a Global Time Master
- and to be stored non-volatile (i.e., `StbMStoreTimebaseNonVolatile == STORAGE`),

the StbM shall load the stored `backup time` from `NvM` (refer to [\[SWS_StbM_00172\]](#) and [\[SWS_StbM_00555\]](#)).

Immediately after loading the stored `backup time` from `NvM`, the StbM shall store a new `backup time` (= loaded (old) `backup time` + `StbMCyclicBackupInterval`) to `NvM`.

In case the restorage is not successful, the Time Base shall start with zero.]

Note: The further details on the `NvM` handling are intentionally left open. The implementer could choose e.g. between

- the ReadAll/WriteAll functionality from `NvM`
- or explicit `NvM`-Block configuration and synchronization

Also, block restore could be

- via callback

- or via constant.

[SWS_StbM_00578]

Status: DRAFT

Upstream requirements: [RS_TS_00004](#)

[For each Synchronized Time Base

- configured as a Global Time Master
- and to be stored non-volatile (i.e., `StbMStoreTimebaseNonVolatile == STORAGE`)
- and using a `Disciplined HW Clock` (i.e., `StbMDisciplinedClock` is configured),

when the stored backup time has been successfully loaded from `NvM`, then the `StbM` shall call `EthIf_SetPhcTime` to set the value of the `Disciplined HW Clock` to the backup time read from `NvM`.

If loading from `NvM` fails, then the `StbM` shall call `EthIf_SetPhcTime` to set the value of the `Disciplined HW Clock` to 0.]

[SWS_StbM_00306]

Upstream requirements: [RS_TS_00034](#)

[If `StbMTimeRecordingSupport` is set to `TRUE`, the `StbM` shall initialize all Block Elements of the measurement recording tables with zero.]

Note: For details on the measurement recording tables (Synchronized Time Base Record Table) refer to chapter 7.15.2 “Global Time Precision Measurement Support”

[SWS_StbM_00427]

Upstream requirements: [RS_TS_00015](#)

[For each Time Base the `StbM` shall initialize all of the corresponding User Data bytes with 0.]

7.9 Shutdown behavior

[SWS_StbM_00555]

Upstream requirements: [RS_TS_00024](#)

[Upon a graceful shutdown, for each Synchronized Time Base configured

- for a Time Master
- and to be stored non-volatile (i.e., `StbMStoreTimebaseNonVolatile == STORAGE`),

the StbM shall store the following value as `backup time` to NvM:

- current time of the Time Base

]

7.10 Immediate Time Synchronization

All Timesync Modules are working independently of the StbM regarding the handling of the bus-specific Time Synchronization protocol (i.e., autonomous transmission of Timesync messages on the bus).

Nevertheless it is necessary, that the StbM provides an interface, based on a `timeBaseUpdateCounter`, to allow the Timesync Modules to detect, if a Time Base has been updated or not and thus may perform an immediate transmission of Timesync messages, e.g. to speed up re-synchronization.

`StbM_GetTimeBaseUpdateCounter` allows the Timesync Modules to detect, whether a Time Base should be transmitted immediately in the subsequent `<Bus>TSyn_MainFunction` cycle.

[SWS_StbM_00414]

Upstream requirements: [RS_TS_00011](#)

[`StbM_GetTimeBaseUpdateCounter` shall return the value of the `timeBaseUpdateCounter` of the corresponding Time Base.]

[SWS_StbM_00351]

Upstream requirements: [RS_TS_00011](#)

[For Synchronized Time Bases, the `timeBaseUpdateCounter` of a Time Base shall have the value range 0 to 255.]

[SWS_StbM_00350]

Upstream requirements: [RS_TS_00011](#)

[For Synchronized Time Bases on a valid invocation of

- `StbM_SetGlobalTime`,
- `StbM_BusSetGlobalTime`,
- or `StbM_TriggerTimeTransmission`,

the StbM shall increment the `timeBaseUpdateCounter` of the corresponding Time Base by 1 (one).

At 255 the `timeBaseUpdateCounter` shall wrap around to 0.]

Note: `StbM_UpdateGlobalTime` can be used instead of `StbM_SetGlobalTime`, if the StbM shall not increment the `timeBaseUpdateCounter` of the corresponding Time Base.

7.11 Time Correction

The Synchronized Time-Base Manager provides the ability for Time Slaves to perform

- Rate Correction (refer to chapters 7.11.1 “Rate Correction Measurement (for Time Slaves)” and 7.11.2 “Rate and Offset Correction (for Time Slaves)”) and
- Offset Correction (refer to chapter 7.11.2)

of their Time Base(s).

For Global Time Masters the StbM provides the ability to perform

- Rate Correction (refer to chapter 7.11.3 “Time Extrapolation and Rate Correction for Global Time Masters”)

of their Time Base(s).

If a rate deviation between the Primary `Virtual Local Time` (TV_{Primary}) and the current Global Time from the Time Master is detected, `Rate Correction` is applied to reduce said rate deviation.

If an offset between the rate corrected Primary `Virtual Local Time` and the current current Global Time from the Time Master is detected, `Offset Correction` is applied to reduce said offset.

Rate and Offset Correction can be configured individually for each Time Base.

7.11.1 Rate Correction Measurement (for Time Slaves)

Rate Correction Measurement determines the rate ratio between the `Virtual Local Time` and the `Global Time`.

The rate correction measurement can be done

- by the StbM itself (refer to chapter 7.11.1.2 “Rate Measurement by StbM”)or
- by the underlying TimeSync module (refer to chapter 7.11.1.1 “Rate Measurement by Timesync Module”)

The measured rate correction factor r_{rc} is used by the StbM to correct the Time Base's time whenever it is determined (e.g., in the scope of `StbM_GetCurrentTime`, refer to [Figure 7.6](#)).

Note: Applying rate correction is inaccurate for short intervals (and for small rate deviation values).

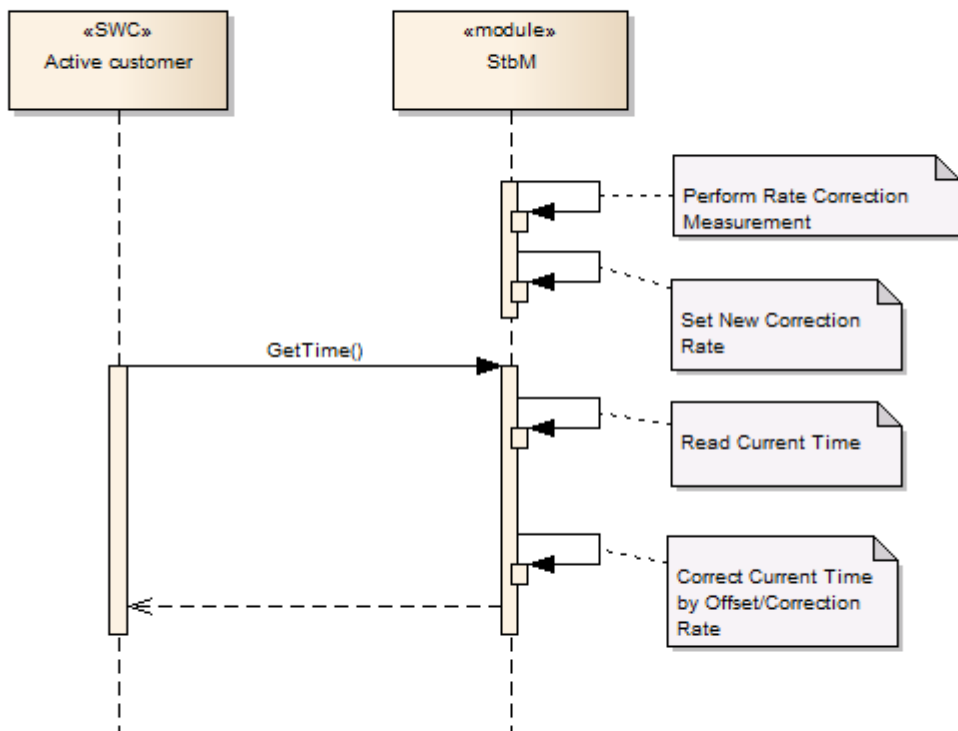


Figure 7.6: Rate Correction

7.11.1.1 Rate Measurement by Timesync Module

Currently only the Timesync module EthTSyn supports a rate measurement.

[SWS_StbM_00576]

Status: DRAFT
Upstream requirements: [RS_TS_00018](#)

[If

- for a Synchronized Time Base `StbMRateSource` is configured to `RATE_SOURCE_TIMESYNC_MODULE`, i.e., the rate is provided by the Timesync module
- and `measureDataPtr->rateDeviation->rateDeviationStatus` is `ETH_RATE_OK`,

when `StbM_BusSetGlobalTime` is called, then the StbM shall set the rate correction r_{rc} to the value of parameter (`measureDataPtr ->rateDeviation.rateDeviationValue / 241`) + 1.]

[SWS_StbM_00577]

Status: DRAFT
Upstream requirements: [RS_TS_00018](#)

[If

- for a Synchronized Time Base `StbMRateSource` is configured to `RATE_SOURCE_TIMESYNC_MODULE`, i.e., the rate is provided by the Timesync module
- and `measureDataPtr->rateDeviation.rateDeviationStatus` is `ETH_RATE_NOT_AVAILABLE` or `ETH_RATE_INVALID`,

when `StbM_BusSetGlobalTime` is called, then the StbM shall

- ignore the value of parameter `measureDataPtr->rateDeviation.rateDeviationValue`
- and not change the rate correction r_{rc}

]

7.11.1.2 Rate Measurement by StbM

[SWS_StbM_00377]

Upstream requirements: [RS_TS_00018](#)

[If for a Time Base

- `StbMRateSource` is configured to `RATE_SOURCE_STBM`, i.e., the rate is provided by the StbM and
- the measurement duration `StbMRateCorrectionMeasurementDuration` is greater than 0,

the StbM itself shall measure the Rate Correction value and use the measured value r_{rc} for Rate Correction.]

[SWS_StbM_00376]

Upstream requirements: [RS_TS_00018](#)

[For Rate Correction measurements, the StbM shall evaluate the [TIMELEAP_FUTURE](#) and [TIMELEAP_PAST](#) flags during measurements. The StbM shall discard the measurement, if any of these flags is set.]

[SWS_StbM_00375]

Upstream requirements: [RS_TS_00018](#)

[For Rate Correction measurements, the StbM shall evaluate state changes of the [SYNC_TO_GATEWAY](#) flag during measurements. The StbM shall discard the measurement if the flag state changes.]

[SWS_StbM_00374]

Upstream requirements: [RS_TS_00018](#)

[For Rate Correction measurements, the StbM shall evaluate

- the [TIMEOUT](#) flag and
- the [TIME_PROGRESSION_INCONSISTENCY](#) flag and
- the [FALLBACK_TIME_EXTRAPOLATION](#) flag

during measurements.

If any of these flags is set, then the StbM shall discard the measurement.]

[SWS_StbM_00373]

Upstream requirements: [RS_TS_00018](#)

[For Rate Correction, the StbM shall evaluate,

- the [TIMELEAP_FUTURE](#)/[TIMELEAP_PAST](#) flags
- the [TIME_PROGRESSION_INCONSISTENCY](#) flag and
- the [FALLBACK_TIME_EXTRAPOLATION](#) flag

at the start of a measurement.

If any of these flags is set, then the StbM shall not start a Rate Correction measurement.]

[SWS_StbM_00372]

Upstream requirements: [RS_TS_00018](#)

[Unless Rate Correction is disabled [[SWS_StbM_00377](#)], the StbM shall perform Rate Correction measurements to determine the rate deviation of each configured Time Base.]

[SWS_StbM_00371]

Upstream requirements: [RS_TS_00018](#)

[The StbM shall perform Rate Correction measurements continuously. The end of a measurement marks the start of the next measurement.]

The start and end of measurements are always triggered by and aligned to the reception of time values for Synchronized Time Bases.]

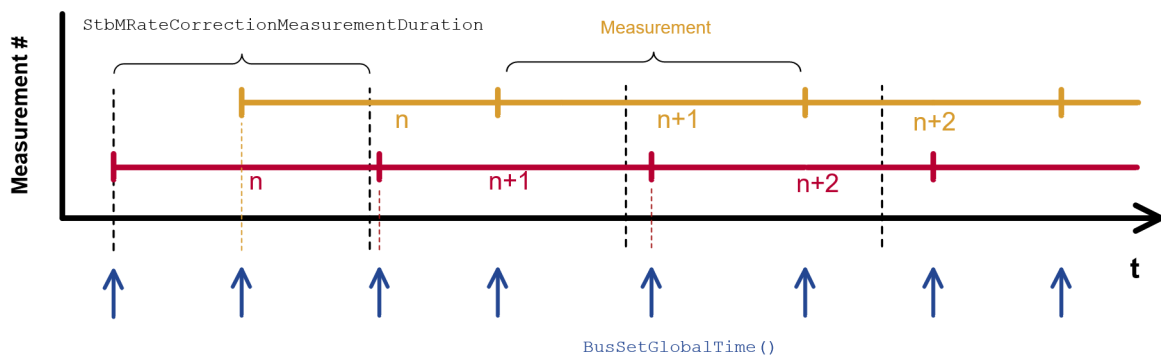


Figure 7.7: Visualization of two parallel measurements

[SWS_StbM_00370]

Upstream requirements: [RS_TS_00018](#)

[During runtime the StbM shall determine the timespan of a Rate Correction measurement on the basis of the Virtual Local Time.]

Note: Simply counting `StbM_BusSetGlobalTime` calls (caused by incoming Timesync messages) and deriving the timespan, which has passed from the cycle time, may lead to incorrect results, because the Timesync cycle time is allowed to vary.

The Global Time is only suitable as a time reference for determining the timespan of a Rate Correction measurement, if time leap detection is configured appropriately - otherwise time leaps may shorten or lengthen the time interval unacceptably.

Instead the timespan should be determined either

- based on the Virtual Local Time or
- by counting invocations of the main function `StbM_MainFunction`

In the latter case, when determining the number of invocations based on `StbMMainFunctionPeriod` and `StbMRateCorrectionMeasurementDuration`, it has to be ensured, that the resulting timespan is not shorter than `StbMRateCorrectionMeasurementDuration`.

Note: For implementation details of the timespan measurement refer to Note after [\[SWS_StbM_00370\]](#).

[SWS_StbM_00368]

Upstream requirements: [RS_TS_00018](#)

[The StbM shall perform as many simultaneous Rate Correction measurements as configured by parameter `StbMRateCorrectionsPerMeasurementDuration` for each configured Time Base.]

[SWS_StbM_00367]

Upstream requirements: [RS_TS_00018](#)

[Simultaneous Rate Correction measurements shall be started with a defined offset (t_{0_n}) to yield Rate Corrections evenly distributed over the measurement duration:

- $t_{0_n} = n * (\text{StbMRateCorrectionMeasurementDuration} / \text{StbMRateCorrectionsPerMeasurementDuration})$

with

- n : zero-based index of the current measurement

]

Note: If a Rate Correction measurement start is delayed e.g. due to a late reception of time values for Synchronized Time Bases (refer also to [\[SWS_StbM_00371\]](#)) such, that it would coincide with the start of a later simultaneous Rate Correction measurement, then the delayed measurement should be discarded and only the most recent one should be started. That is, only one of the simultaneous measurements is started at any reception of time values for Synchronized Time Bases.

Note: The implementation can, e.g., be realized by storing the relevant time snapshots in chained lists. Alternatively, measurements can be seen as objects, which store their relevant data and can be used independently.

[SWS_StbM_00366]

Upstream requirements: [RS_TS_00018](#)

[At the start of a Rate Correction measurement, the StbM shall store the Updated Rx Time Tuple. The elements of the stored Time Tuple have the following denotation:

- TG_{Start} - Global Time part of the Updated Rx Time Tuple
- TV_{Start} - Virtual Local Time part of the Updated Rx Time Tuple

]

Note: This is equivalent to an atomic Time Tuple assignment: $[TG_{Start};TV_{Start}] = [TG_{Rx};TV_{Rx}]$

[SWS_StbM_00364]

Upstream requirements: [RS_TS_00018](#)

[At the end of the Rate Correction measurement, the StbM shall store the Updated Rx Time Tuple. The elements of the stored Time Tuple have the following denotation:

- TG_{Stop} - Global Time part of the Updated Rx Time Tuple
- TV_{Stop} - Virtual Local Time part of the Updated Rx Time Tuple

]

Note: This is equivalent to an atomic Time Tuple assignment: $[TG_{Stop};TV_{Stop}] = [TG_{Rx};TV_{Rx}]$

[SWS_StbM_00361]

Upstream requirements: [RS_TS_00018](#)

[At the end of a Rate Correction measurement, the StbM shall calculate the resulting correction rate (r_{rc}) for Synchronized Time Bases as shown:

$$r_{rc} = \frac{TG_{Stop} - TG_{Start}}{TV_{Stop} - TV_{Start}} \quad (7.3)$$

]

Note: To determine the resulting rate deviation the value 1 has to be subtracted from r_{rc} .

[SWS_StbM_00362]

Upstream requirements: [RS_TS_00018](#)

[The StbM shall use the same value for the rate correction r_{rc} until a new valid (refer to [\[SWS_StbM_00569\]](#)) value has been calculated.]

Note: A newly calculated Rate Correction r_{rc} is only applied to following time calculations.

[SWS_StbM_00527]

Upstream requirements: [RS_TS_00018](#)

[For Synchronized Time Bases the StbM shall calculate the rate deviation r_{Dev} as $r_{rc} - 1$.]

[SWS_StbM_00397]

Upstream requirements: [RS_TS_00018](#)

[For Synchronized Time Bases and if `StbMIsSystemWideGlobalTimeMaster` is set to `False`, the StbM shall return on invocation of `StbM_GetRateDeviation` the rate deviation r_{Dev} , which has been calculated for that Time Base.

If no rate deviation has been calculated, `StbM_GetRateDeviation` shall return `E_NOT_OK`.]

[SWS_StbM_00412]

Upstream requirements: [RS_TS_00018](#)

[For a Synchronized Time Base the StbM shall use $r_{rc} = 1$, if a valid correction rate (r_{rc}) has not yet been calculated or is not being calculated (refer [[SWS_StbM_00377](#)]) but shall be applied.]

[SWS_StbM_00569]

Upstream requirements: [RS_TS_00009](#)

[For Synchronized Time Bases if

- `StbMIsSystemWideGlobalTimeMaster` is set to `False`
- and `StbMRateCorrectionThreshold` is greater than 0,

then the StbM shall check if the absolute value of the calculated rate deviation exceeds the configured value of `StbMRateCorrectionThreshold`.

If the threshold is exceeded, then the StbM shall set the `RATE_EXCEEDED` bit within `timeBaseStatus` of the Time Base and shall consider the calculated rate deviation as not valid.

If the threshold is not exceeded, then the StbM shall clear the `RATE_EXCEEDED` bit within `timeBaseStatus` of the Time Base.]

[SWS_StbM_00570]

Upstream requirements: [RS_TS_00009](#)

[If a valid (refer to [[SWS_StbM_00569](#)]) rate correction value has been calculated for a Time Base, then the StbM shall set the bit `RATE_CORRECTED` within `timeBaseStatus` of the Time Base to 1.]

Note: Only `StbM_Init` will reset the bit `RATE_CORRECTED`, i.e., once set, the flag remains set until a reset of the StbM.

7.11.2 Rate and Offset Correction (for Time Slaves)

The Time Slave of a Synchronized Time Base adjusts its `Local Instance of the Global Time` to the Global Time as provided by a Time Master. For doing so, the StbM has to

- eliminate the difference between the frequency of the local clock and the frequency of the clock of the Time Master, referred to as rate deviation r_{Dev}
- eliminate the `Time Offset` $o_{SyncDiff}$ between the `Updated Rx Time Tuple` (TG_{URx}) and the Global Time of the `SyncLocal Time Tuple` (TL_{Sync})

while interpolating the `Local Instance of the Global Time`.

This chapter specifies how the StbM

- eliminates r_{Dev} by applying the rate correction factor r_{rc} as derived in chapter [Section 7.11](#)
- and eliminates the time offset $o_{SyncDiff}$ in two alternative ways:
 - “**Offset Correction by Jump**”: the time of the `Local Instance of the Global Time` (TL) ‘jumps’ to the value of the `Updated Rx Time` (TG_{URx}), i.e., it jumps by the `Time Offset` $o_{SyncDiff}$ (refer to [Figure 9.7](#)).
 - “**Offset Correction by Rate Adaption**”: the applied `Rate Correction` r_{rc} is adapted by an additional rate offset r_{oc} such that the existing `Time Offset` $o_{SyncDiff}$ is steadily reduced to zero within a configured time span denoted as `Offset Correction Adaption Interval` (as configured by `StbMOffsetCorrectionAdaptionInterval`, refer to [Figure 9.7](#)).

[SWS_StbM_00359]

Upstream requirements: [RS_TS_00018](#)

[When `StbM_BusSetGlobalTime` is called and if it is not the the very first call of `StbM_BusSetGlobalTime`, then the StbM shall calculate `Time Offset` $o_{SyncDiff}$ between the Updated Received Time and the Synclocal Time as

- $o_{SyncDiff} = TG_{URx} - TL_{Sync}$

]

For Synchronized Time Bases `Rate Correction` and `Offset Correction by Rate Adaption` is only supported if `Rate Correction` is enabled. If not enabled, StbM will only do `Offset Correction by Jump`.

7.11.2.1 Offset Correction by Jump

[SWS_StbM_00588]

Status: DRAFT

Upstream requirements: [RS_TS_00018](#)

[For Synchronized Time Bases

when a [Time Offset](#) o_{SyncDiff} is calculated according to [\[SWS_StbM_00359\]](#),

if

- [Rate Correction](#) is disabled, i.e., [StbMRateSource](#) is not configured

then the StbM shall

- apply an [Offset Correction by Jump](#), i.e.,
 - set value $o_{\text{oc}} = o_{\text{SyncDiff}}$
 - and update the [Main Time Tuple](#) to the [Updated Rx Time Tuple](#)
- and apply no [Rate Correction](#), i.e., set $r = 1$.

]

For Synchronized Time Base, which have Rate Correction enabled, Offset Correction by Rate Adaption can be switched on or off.

[SWS_StbM_00400]

Upstream requirements: [RS_TS_00018](#)

[For Synchronized Time Bases

when a [Time Offset](#) o_{SyncDiff} is calculated according to [\[SWS_StbM_00359\]](#),

and if

- [Rate Correction](#) is enabled, i.e., [StbMRateSource](#) is configured
- and [StbMOffsetCorrectionJumpThreshold](#) is set to 0

then the StbM shall

- apply an [Offset Correction by Jump](#), i.e.,
 - set value $o_{\text{oc}} = o_{\text{SyncDiff}}$
 - and update the [Main Time Tuple](#) to the [Updated Rx Time Tuple](#)
- and apply no [Offset Correction by Rate Adaption](#), i.e.,
 - set $r_{\text{oc}} = 0$
 - and with that set $r = r_{\text{rc}}$.

」

If the *Time Offset* is too big to be smoothly reduced to zero within a configured time span, the StbM will do a *Offset Correction by Jump*.

[SWS_StbM_00440]

Upstream requirements: RS_TS_00018

[For Synchronized Time Bases

when a *Time Offset* o_{SyncDiff} is calculated according to [SWS_StbM_00359]

and if

- *Rate Correction* is enabled, i.e., *StbMRateSource* is configured
- and absolute value $\text{abs}(o_{\text{SyncDiff}})$ is equal or greater than *StbMOffsetCorrectionJumpThreshold*,

then the StbM shall

- apply an *Offset Correction by Jump*, i.e.,
 - set value $o_{\text{oc}} = o_{\text{SyncDiff}}$
 - and update the *Main Time Tuple* to the *Updated Rx Time Tuple*
- and apply no *Offset Correction by Rate Adaption*, i.e.,
 - set $r_{\text{oc}} = 0$
 - and with that set $r = r_{\text{rc}}$

」

7.11.2.2 Offset Correction by Rate Adaption

[SWS_StbM_00356]

Upstream requirements: RS_TS_00018, RS_TS_00019

[For Synchronized Time Bases

when a *Time Offset* o_{SyncDiff} is calculated according to [SWS_StbM_00359],

and if

- *Rate Correction* is enabled, i.e., *StbMRateSource* is configured
- and absolute value $\text{abs}(o_{\text{SyncDiff}})$ is less than *StbMOffsetCorrectionJumpThreshold*,

then the StbM shall

- start the `Offset Correction Adaption Interval`
- calculate a new value for the `Offset Correction by Rate Adaption` as $r_{oc} = O_{SyncDiff} / StbMOffsetCorrectionAdaptionInterval$,
- update the `Applied Rate` r accordingly, i.e., $r = r_{rc} + r_{oc}$
- and apply no `Offset Correction by Jump`, i.e.,
 - set value $o_{oc} = 0$
 - and update the `Main Time Tuple` to the `SyncLocal Time Tuple`

]

[SWS_StbM_00353]

Upstream requirements: RS_TS_00018

[For a Synchronized Time Base

if StbM does not use a `Disciplined HW Clock` to maintain the `Local Instance of the Global Time` (container `StbMDisciplinedClock` not configured),

and when the `Offset Correction Adaption Interval` expires (refer to [SWS_StbM_00356] for start condition),

then the StbM shall in that order:

1. use r as derived by [SWS_StbM_00356], i.e., $r = r_{rc} + r_{oc}$
2. update the `Main Time Tuple` [TL_{Main} , TV_{Main}] in an atomic sequence as:
 - $TV_{Main} = TV_{Main} + StbMOffsetCorrectionAdaptionInterval$
 - and $TL_{Main} = TL_{Main} + r * StbMOffsetCorrectionAdaptionInterval$
3. and set $r_{oc} = 0$, i.e., no longer apply an `Offset Correction by Rate Adaption`.

]

Note: This assumes no update of the `Main Time Tuple` has occurred due to e.g. another Sync/FollowUp message pair being processed by the TimeSync module.

Note: "atomic sequence" means, that the two members of the Time Tuple need to be updated by an atomic, i.e., non-interruptable, operation, such that it remains consistent.

Note: If `Local Instance of the Global Time` TL is not derived from a disciplined HW clock the StbM needs to update the `Main Time Tuple` according to [SWS_StbM_00353] after expiration of `StbMOffsetCorrectionAdaptionInterval`

- either in the next `StbM_GetCurrentTime` (i.e., `StbM_GetCurrentTime` will calculate first a new `Main Time Tuple` and then based on that the Time Tuple [TL, TV])
- or in the next Main Function

depending on which of the two events occurs first.

If `Local Instance of the Global Time TL` is derived from a `Disciplined HW Clock` the StbM needs to update that HW clock immediately once a new `r` has been set after `Offset Correction Adaption Interval` has expired.

Imprecisions arising from using Rate Adaption have to be considered by the user.

[SWS_StbM_00587]

Status: DRAFT

Upstream requirements: RS_TS_00018

[If StbM uses a `Disciplined HW Clock` to maintain the `Local Instance of the Global Time` (container `StbMDisciplinedClock` configured),

and when the `Offset Correction Adaption Interval` expires (refer to [SWS_StbM_00356] for start condition),

then the StbM shall and no longer apply an `Offset Correction by Rate Adaption`, i.e., set $r_{oc} = 0$, and with that update the `Applied Rate` $r = r_{rc}$]

[SWS_StbM_00586]

Status: DRAFT

Upstream requirements: RS_TS_00018

[If StbM uses a `Disciplined HW Clock` to maintain the `Local Instance of the Global Time` (container `StbMDisciplinedClock` configured),

when

- – a new `Rx Time Tuple` (TG_{Rx}) via `StbM_BusSetGlobalTime` has been received
- or the `Offset Correction Adaption Interval` has expired
- and
 - a new `Applied Rate` r (if `Rate Correction` is enabled)
 - and a new `Offset Correction` O_{oc}

are calculated

then the StbM shall update the `Disciplined HW Clock` by calling `EthIf_SetPhcCorrection` with

- parameter `rateDeviation` of `EthIf_SetPhcCorrection` set to $r - 1$

- and parameter `offset` of `EthIf_SetPhcCorrection` set to `ooc`

]

Note: It is assumed, that the `r` and `ooc` change simultaneously and that the StbM will call `EthIf_SetPhcCorrection` only once within `StbM_BusSetGlobalTime` updating the rate and offset at the same time.

7.11.3 Time Extrapolation and Rate Correction for Global Time Masters

Rate correction in Global Time Masters can be applied to Synchronized Time Bases and Pure Local Time Bases.

Offset Correction is not supported by Global Time Masters

Use cases are setting the rate of a Pure Local Time Base to the rate of a received Synchronized Time Base or adjusting the rate of Synchronized Time Bases to external time sources (e.g., GPS).

Rate correction is applied by setting a correction factor which the StbM uses to correct the Time Base's time whenever it is read (e.g., in the scope of `StbM_GetCurrentTime`).

As a Global Time Master the StbM derives the `Local Instance of the Global Time` (TL) for a Time Base

- by extrapolation based on the `Main Time Tuple`, the current value of the `Virtual Local Time` and the `Applied Rate` value according to [SWS_StbM_00355], if no `Disciplined HW Clock` is supported.
- by reading the `Disciplined HW Clock`, if `Disciplined HW Clock` is supported.

[SWS_StbM_00424]

Upstream requirements: RS_TS_00018

[For a Global Time Master of a Synchronized Time Base if a `Disciplined HW Clock` is not supported (`StbMDisciplinedClock` not configured),

then the StbM shall calculate the `Local Instance of the Global Time` (TL) of a Time Base according to [SWS_StbM_00355], where the `Applied Rate` `r` is set to $r = r_{rc}$.]

[SWS_StbM_00581]

Status: DRAFT

Upstream requirements: [RS_TS_00018](#)

[For a Global Time Master of a Synchronized Time Base and for a Pure Local Time Base if `StbMAllowMasterRateCorrection` equals `FALSE`, then the StbM shall set the value of the `Rate Correction` $r_{rc} = 1$.]

[SWS_StbM_00431]

Upstream requirements: [RS_TS_00018](#)

[For the Time Master of a Synchronized Time Base and a Pure Local Time Base if a valid correction rate (r_{rc}) has not yet been set, then the StbM shall set $r_{rc} = 1$.]

[SWS_StbM_00396]

Upstream requirements: [RS_TS_00018](#)

[If the absolute value of the parameter `rateDeviation` of `StbM_SetRateCorrection` is greater than `StbMMasterRateDeviationMax`, then when `StbM_SetRateCorrection` is called, the StbM shall limit the rate deviation value as handed over by parameter `rateDeviation` to

- either (`+StbMMasterRateDeviationMax`), if `rateDeviation` is positive
- or (`-StbMMasterRateDeviationMax`), if `rateDeviation` is negative

]

[SWS_StbM_00395]

Upstream requirements: [RS_TS_00018](#)

[If `StbMAllowMasterRateCorrection` equals `FALSE`, then `StbM_SetRateCorrection` shall do nothing and return `E_NOT_OK`.]

[SWS_StbM_00583]

Status: DRAFT

Upstream requirements: [RS_TS_00018](#)

[For a Global Time Master of a Synchronized Time Base

if

- `StbMAllowMasterRateCorrection` equals `TRUE`
- and a `Disciplined HW Clock` is not supported (`StbMDisciplinedClock` not configured),

when `StbM_SetRateCorrection` is called,

then the StbM shall in that order

1. get the current `Virtual Local Time` (TV_{Temp})
2. calculate the `Local Instance of the Global Time` (TL_{Temp}) for TV_{Temp} according to [\[SWS_StbM_00424\]](#)
3. update the current `Main Time Tuple` with the values of the temporary Time Tuple [TL_{Temp} , TV_{Temp}]
4. update the `Applied Rate` r by setting it to $r_{rc} = (\text{rateDeviation} + 1)$ with `rateDeviation` of `StbM_SetRateCorrection` limited according to [\[SWS_StbM_00396\]](#)

]

[SWS_StbM_00580]*Status:* DRAFT*Upstream requirements:* [RS_TS_00018](#)

[For a Global Time Master of a Synchronized Time Base

if

- `StbMAllowMasterRateCorrection` equals TRUE
- and a `Disciplined HW Clock` is supported (`StbMDisciplinedClock` is configured),

when `StbM_SetRateCorrection` is called,

then the StbM shall

- update the `Applied Rate` r by setting it to $r_{rc} = (\text{rateDeviation} + 1)$ with `rateDeviation` of `StbM_SetRateCorrection` limited according to [\[SWS_StbM_00396\]](#)
- then update the `Disciplined HW Clock` by calling `EthIf_SetPhcCorrection` with
 - parameter `rateDeviation` (of `EthIf_SetPhcCorrection`) set to `rateDeviation` (of `StbM_SetRateCorrection`)
 - and parameter `offset` of `EthIf_SetPhcCorrection` set to 0

]

Note: If aligning the rate of one Time Base to the rate of another one, it is possible to get the `Rate Deviation` via `StbM_GetRateDeviation` and pass that value as argument to `StbM_SetRateCorrection`.

[SWS_StbM_00568]

Upstream requirements: [RS_TS_00009](#)

[Upon a valid invocation of [StbM_SetRateCorrection](#) the StbM shall set the [RATE_CORRECTED](#) bit of the status (`timeBaseStatus`) of the corresponding Time Base to 1.]

Note: The bit [RATE_CORRECTED](#) can only be reset by [StbM_Init](#) or by a clone operation (refer to [\[SWS_StbM_00535\]](#)).

[SWS_StbM_00422]

Upstream requirements: [RS_TS_00018](#)

[

- For Synchronized Time Bases, if [StbMIsSystemWideGlobalTimeMaster](#) equals `TRUE`
- and for Pure Local Time Bases

on invocation of [StbM_GetRateDeviation](#) the StbM shall return the current rate deviation.

If

- no rate deviation has been set,
- or rate deviation has been reset during clone operation (refer to [\[SWS_StbM_00535\]](#))

then [StbM_GetRateDeviation](#) shall return `E_NOT_OK`.]

7.12 Time Base Cloning

The StbM provides an API to clone a Time Base (denoted as Source Time Base) by copying its current value, User Data and rate correction to another Time Base (denoted as Destination Time Base).

The clone API is called for the Destination Time Base to copy data from its Source Time Base. The StbM acts as a Global Time Master for the Destination Time Base. The StbM supports cloning for Synchronized and Pure Local Time Bases.

The StbM will clone the Time Base only if the Source Time Base's current status matches certain criteria (e.g., if no timeleap or timeout is present).

Other than copying Time Bases on application level the cloning API avoids loss of precision when copying the Time Bases.

Possible use cases for cloning are fallback and redundancy scenarios as well as reverse synchronization.

[SWS_StbM_00530]

Upstream requirements: [RS_TS_00038](#)

[For Synchronized Time Bases and Pure Local Time Bases upon invocation of [StbM_CloneTimeBase](#), the StbM shall

- first determine
 - the Destination Time Base (given by input parameter [timeBaseId](#)) and
 - the Source Time Base (given by configuration parameter [StbMSourceTimeBase](#) of the given Destination Time Base)
- then derive a [Source;Destination] time tuple from the Time Bases
- and then check the [DEFERRED_COPY](#) flag in [cloneCfg](#).

If the [DEFERRED_COPY](#) flag is set, the StbM shall

- store the clone request together with the parameters passed by [StbM_CloneTimeBase](#) as 'deferred' and
- replace any pending deferred clone request of the same [Source;Destination] tuple by the actual deferred clone request
- return `E_OK`.

If the [DEFERRED_COPY](#) flag is not set, the StbM shall

- remove any pending deferred clone request of the same [Source;Destination] tuple and
- immediately process the clone request.

]

[SWS_StbM_00531]

Upstream requirements: [RS_TS_00038](#)

[For Synchronized Time Bases, which are configured as a source of a clone operation, the StbM shall check on every change of the [TIMEOUT](#) bit in the [timeBaseStatus](#) if a pending deferred clone request exists.

If

- a pending deferred clone request exists
- and the flag [RETRIGGER_DEFERRED_COPY](#) of the [cloneCfg](#) parameter is not set

then the StbM shall remove the deferred clone request.]

Note: As a consequence any pending deferred clone request is removed when

- the related Source Time Base enters `TIMEOUT` state
- and the flag `RETRIGGER_DEFERRED_COPY` of the `cloneCfg` parameter is not set.

Furthermore, if the Source Time Base is already in `TIMEOUT` state when

- `StbM_CloneTimeBase` is invoked with the `DEFERRED_COPY` flag being set
- and the `RETRIGGER_DEFERRED_COPY` flag being not set,

then the effect of removing the request when leaving the `TIMEOUT` state later on is equivalent to not storing the pending deferred clone request at all.

[SWS_StbM_00532]

Upstream requirements: [RS_TS_00038](#)

[For Synchronized Time Bases, which are configured as a source of a clone operation, the StbM shall check on every invocation of `StbM_BusSetGlobalTime` if a pending deferred clone request exists.

If such a pending deferred clone request exists it shall be processed once `StbM_BusSetGlobalTime` has been processed completely, i.e., the Time Base Status has been updated, the Updated Rx Time Tuple has been subject to rate correction calculations and the Main Time Tuple has been overwritten by the Updated Rx Time Tuple.

After processing the pending deferred clone request, if the flag `RETRIGGER_DEFERRED_COPY` of the `cloneCfg` parameter is not set, then the StbM shall remove the deferred clone request.]

Note: While the flag `RETRIGGER_DEFERRED_COPY` of the `cloneCfg` parameter is set, deferred cloning is permanent, i.e., the application does not need to cyclically re-invoke `StbM_CloneTimeBase`.

Note: [Figure 7.4](#) illustrates the sequence of Time Tuple updates upon invocation of `StbM_BusSetGlobalTime`

[SWS_StbM_00533]

Upstream requirements: [RS_TS_00038](#)

[To process an immediate or deferred clone request the StbM shall first mask (logical AND) the current Time Base Status with the `statusMask` parameter of the clone request.

If the masked value is equal to the `statusValue` parameter of the clone request, the StbM shall perform the clone operation.

When processing an immediate clone request in the course of `StbM_CloneTimeBase`, the StbM shall return `E_OK` if the clone request was successfully performed, otherwise the StbM shall return `E_NOT_OK`.]

[SWS_StbM_00534]

Upstream requirements: [RS_TS_00038](#)

[If a `Disciplined HW Clock` is not supported (`StbMDisciplinedClock` not configured), when performing the clone operation, then the StbM shall copy the Time Base value as follows:

If both, Source and Destination Time Base, are using the same Virtual Local Time Source, then the Main Time Tuple shall be copied, otherwise the StbM shall

- establish a protection against interruptions and run the next two steps directly afterwards:
- retrieve the current Virtual Local Time TV_{Source} for the Source Time Base
- retrieve the current Virtual Local Time $TV_{Destination}$ for the Destination Time Base
- the protection against interruptions can be removed now
- determine a temporary Time Tuple $[TL_{Source};TV_{Source}]$
- create a Time Tuple $[TL_{Source};TV_{Destination}]$ which then replaces the Main Time Tuple of the Destination Time Base.

]

[SWS_StbM_00585]

Status: DRAFT

Upstream requirements: [RS_TS_00038](#)

[If a `Disciplined HW Clock` is supported (`StbMDisciplinedClock` is configured),

when performing the clone operation the StbM shall copy the Time Base value.

The StbM shall copy TL_{Source} to $TL_{Destination}$ by executing the following to steps in an atomic, i.e., uninterruptable sequence

- retrieve the current `Local Instance of the Global Time` TL_{Source} for the Source Time Base by calling `EthIf_GetPhcTime`
- set the `Local Instance of the Global Time` $TL_{Destination}$ for the Destination Time Base by calling `EthIf_SetPhcTime`

]

[SWS_StbM_00584]

Status: DRAFT

Upstream requirements: [RS_TS_00038](#)

[When performing the clone operation the StbM shall copy the User Bytes of the Source Time Base to the ones of the Destination Time Base]

[SWS_StbM_00535]

Upstream requirements: [RS_TS_00038](#)

[When performing the clone operation the StbM shall check the flag [APPLY_RATE](#) of the `cloneCfg` parameter of the clone request.

If

- the [APPLY_RATE](#) flag is set
- and a valid rate correction has been calculated for the Source Time Base (i.e., the bit [RATE_CORRECTED](#) in the Time Base Status of the Source Time Base is set),

then the StbM shall copy the Rate Correction value and the value of the [RATE_CORRECTED](#) flag of the Source Time Base to the Time Base Status (`timeBaseStatus`) of the Destination Time Base.

If

- the [APPLY_RATE](#) flag is set
- and no valid rate correction has been calculated for the Source Time Base (i.e., the bit [RATE_CORRECTED](#) in the Time Base Status of the Source Time Base is not set),

then the StbM shall

- copy the value of the [RATE_CORRECTED](#) flag of the Source Time Base to the Time Base Status (`timeBaseStatus`) of the Destination Time Base.
- and reset the rate correction of the Destination Time Base, i.e., set $r_{rc} = 1$.

]

Note: The Destination Time Base will apply the Rate Correction value only if [StbMAllowMasterRateCorrection](#) is set to `TRUE`.

[SWS_StbM_00536]

Upstream requirements: [RS_TS_00038](#)

[When performing the clone operation the StbM shall check the flag [IMMEDIATE_TX](#) of the `cloneCfg` parameter of the clone request.

If the flag is set, StbM shall increment the `timeBaseUpdateCounter` of the Destination Time Base after having copied the Time Base value and User Data to the Destination Time Base (refer to [SWS_StbM_00534]) to force an immediate transmission of the Time Base on the bus.]

Note: If `IMMEDIATE_TX` flag is not set, the Destination Time Base will be transmitted on the bus with the next cyclic transmission of the corresponding Timesync module(s).

For a Pure Local Time Base the `IMMEDIATE_TX` flag has no effect since Pure Local Time Bases don't have a `timeBaseUpdateCounter`.

[SWS_StbM_00629] Adding Offset when Cloning

Upstream requirements: [RS_TS_00038](#)

[When performing the clone operation the StbM shall check the flag `APPLY_OFFSET` of the `cloneCfg` parameter of the clone request.

If the flag is set and an Offset value has been set, then the StbM shall add the Offset value as set by `StbM_SetCloningOffset` for the Destination Time Base to the time value being written to the Destination Time Base.]

7.13 Notification of Customers

The StbM allows Notification Customers (i.e., SW-Cs or other BSW modules) either to register to be notified of status change events for a Time Base or to be notified if an alarm expires.

7.13.1 Time Notifications

The StbM allows Notification Customers to register to be notified if a Customer specific alarm expires.

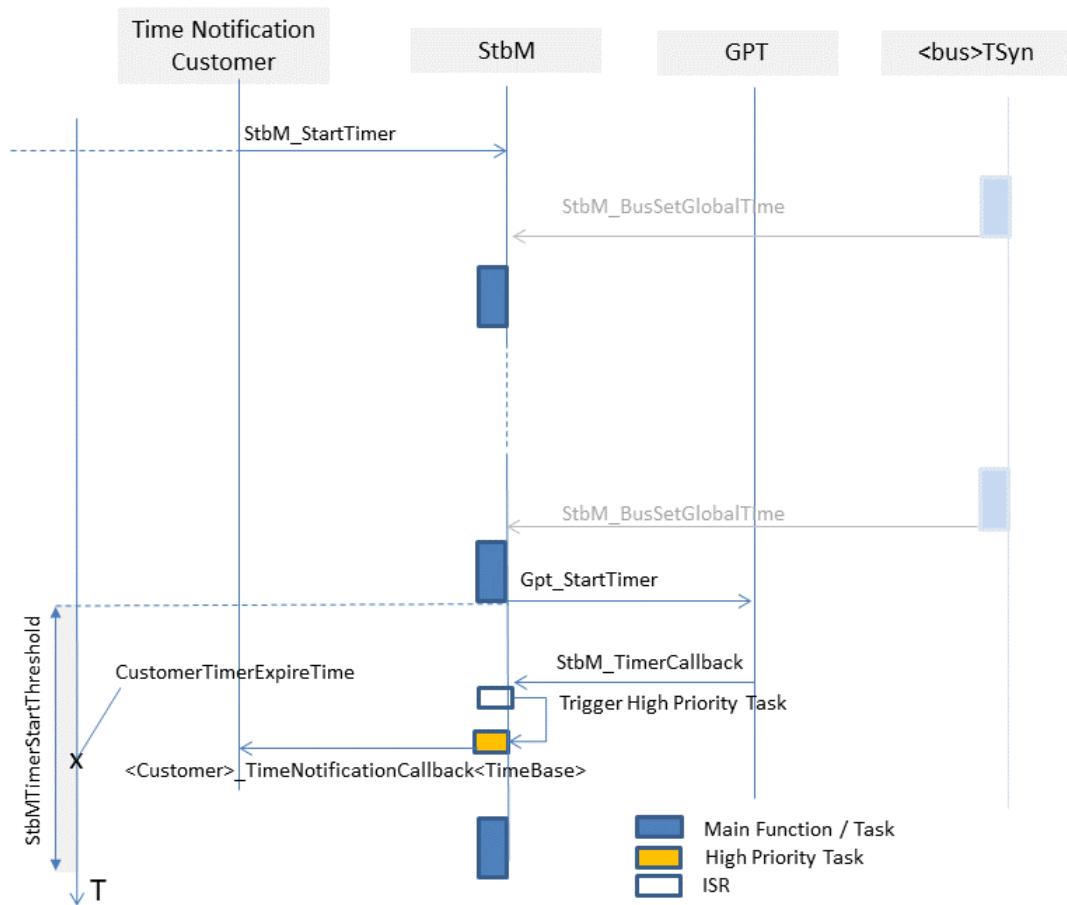


Figure 7.8: Basic mechanism of Time Notification

[SWS_StbM_00421]

Upstream requirements: [RS_TS_00017](#)

[If any `StbMNotificationCustomer` is configured, the StbM shall use one additional GPT source (referenced by `StbMGptTimerRef`), which is not used for other purposes.]

[SWS_StbM_00270]

Upstream requirements: [RS_TS_00017](#)

[On invocation of `StbM_StartTimer` for a Time Notification Customer of a Time Base the StbM shall calculate the time `CustomerTimerExpireTime` when that Customer Timer will expire based on the corresponding Time Base. If a Customer Timer for the same Customer is already running, `StbM_StartTimer` shall return `E_NOT_OK`.]

[SWS_StbM_00335]

Upstream requirements: [RS_TS_00017](#)

[For currently active Time Notification Customers the StbM shall cyclically calculate and monitor in its [StbM_MainFunction](#) the difference between the current value of the corresponding Time Base and the expiration time [CustomerTimerExpireTime](#).]

Note: Cyclic recalculation accounts for asynchronous updates of the Time Base e.g. by [StbM_BusSetGlobalTime](#).

[SWS_StbM_00336]

Upstream requirements: [RS_TS_00017](#)

[A time interval [StbMTimerStartThreshold](#) before a Customer Timer expires, the StbM shall calculate the time difference between [CustomerTimerExpireTime](#) and the current value of the corresponding Time Base.

The StbM shall then start a GPT timer via [Gpt_StartTimer](#) to be notified, when the time difference has elapsed.]

Note: [StbMTimerStartThreshold](#) should be set to a value greater than [StbM_MainFunctionPeriod](#) to account for the jitter of the [StbM_MainFunction](#).

If the GPT timer expires for a Time Notification Customer, [StbM_TimerCallback](#) is called by the GPT.

[SWS_StbM_00271]

Upstream requirements: [RS_TS_00017](#)

[Upon invocation of [StbM_TimerCallback](#), the StbM shall calculate the time difference between [CustomerTimerExpireTime](#) and the current value of the corresponding Time Base.

If the calculated time difference exceeds the value range of the [deviationTime](#) parameter of [<Customer>_TimeNotificationCallback<TimeBase>](#) the returned time difference shall be limited to either the maximum negative or the maximum positive value of type [StbM_TimeDiffType](#)).

If [StbMTimeNotificationCallback](#) is not NULL,

- the StbM shall call the function [<Customer>_TimeNotificationCallback<TimeBase>](#)

else

- the StbM shall call the service operation [NotifyTime](#) of the required port [GlobalTime_TimeEvent](#)

to inform the corresponding Time Notification Customer and return the value of the calculated time difference in the parameter [deviationTime](#).]

Note: `StbM_TimerCallback` is called in interrupt context. The operation `NotifyTime` may however only be called from task context. Therefore, the StbM has to decouple the interrupt context from the task context (e.g. by triggering an `ExternalTriggerOccurredEvent`). The details are considered to be implementation specific.

Note: The `StbM_TimerCallback` notification function, which is implemented by the StbM and called by the Gpt, conforms to the `Gpt_Notification_<channel>` prototype. The configured notification function `StbM_TimerCallback` is declared via Gpt header.

[SWS_StbM_00432]

Upstream requirements: [RS_TS_00017](#)

[When the StbM detects for an active customer, while monitoring the difference to `CustomerTimerExpireTime` (see [\[SWS_StbM_00335\]](#)), that

- the `CustomerTimerExpireTime` has already been passed and
- the GPT timer has not yet been started,

the StbM shall call `<Customer>_TimeNotificationCallback<TimeBase>` immediately.]

Note: The scenario that `CustomerTimerExpireTime` expires while GPT timer is not yet started may occur, when the Time Base jumps over the expiration time (i.e., `CustomerTimerExpireTime`) due to an invocation of `StbM_BusSetGlobalTime` but the GPT timer was not yet started.

Note: If GPT timer is already running, StbM will call `<Customer>_TimeNotificationCallback<TimeBase>` only when GPT timer expires.

Note: "immediately" means, that the StbM calls `<Customer>_TimeNotificationCallback<TimeBase>` within the same main function call in which it detects, that `CustomerTimerExpireTime` has already been passed. So, in this case `<Customer>_TimeNotificationCallback<TimeBase>` is called from task context.

[SWS_StbM_00337]

Upstream requirements: [RS_TS_00017](#)

[If multiple `Customer Timers` run and expire within the same interval `StbMTimerStartThreshold`, the StbM shall calculate all expiry points within the `StbMTimerStartThreshold` interval and re-start the same GPT timer for next expiry point after the previous expiry point has been reached.]

Caveat: If a `StbM_BusSetGlobalTime` function call occurs and updates the Time Base, for which a GPT timer is running, the newly received Global Time value could be in the future relative to the Local Time of the Time Base. Depending on how far, that

value is in the future, it could mean, that the timer expires too late (based on the new Global Time).

7.13.2 Status Notifications

The StbM allows Notification Customers to register to be notified of status change events for a Time Base. The StbM tracks for each registered Notification Customer the occurrence of various Time Base related events. Notification Customers can configure the StbM such, that they will be informed by a notification callback, if one or more events occur.

[SWS_StbM_00277]

Upstream requirements: [RS_TS_00037](#), [RS_TS_00016](#)

[For Synchronized and Pure Local Time Bases:

- If parameter `StbMNotificationInterface` is set to either `SR_INTERFACE` or `CALLBACK_AND_SR_INTERFACE`, the StbM shall notify the application via the `StatusNotification` service interface.
- If parameter `StbMNotificationInterface` is set to either `CALLBACK` or `CALLBACK_AND_SR_INTERFACE`, the StbM shall use the callback `StatusNotificationCallback<TimeBase>` to notify a CDD about status related events.
- If parameter `StbMNotificationInterface` is set to `NO_NOTIFICATION` the notification mechanism shall be disabled for the given Time Base.

The callback `StatusNotificationCallback<TimeBase>` shall be set via configuration parameter `StbMNotificationInterface`.]

[SWS_StbM_00526]

Upstream requirements: [RS_TS_00037](#), [RS_TS_00016](#)

[The StbM shall call the Status Notification callback from `StbM_MainFunction`.]

Note: Since a Status Notification is triggered inside `StbM_MainFunction`, the other functions like e.g `StbM_GetTimeBaseStatus` might detect a timeout condition sooner than the corresponding Status Notification is actually triggered. Such a delayed Status Notification is considered acceptable.

[SWS_StbM_00279]

Upstream requirements: [RS_TS_00037](#), [RS_TS_00016](#)

[For each Time Base the StbM has a configurable mask `StbMStatusNotificationMask`, which allows to mask individually status event notifications.]

[SWS_StbM_00284] Status Events detected by the StbM

Upstream requirements: [RS_TS_00016](#)

[

Status Event Name	Status Event Set Condition
EV_GLOBAL_TIME	1: GLOBAL_TIME_BASE bit in <code>timeBaseStatus</code> has changed from 0 to 1 0: otherwise
EV_TIMEOUT_OCCURRED	1: TIMEOUT bit in <code>timeBaseStatus</code> has changed from 0 to 1 0: otherwise
EV_TIMEOUT_REMOVED	1: TIMEOUT bit in <code>timeBaseStatus</code> has changed from 1 to 0 0: otherwise
EV_TIMELEAP_FUTURE	1: TIMELEAP_FUTURE bit in <code>timeBaseStatus</code> has changed from 0 to 1 0: otherwise
EV_TIMELEAP_FUTURE_REMOVED	1: TIMELEAP_FUTURE bit in <code>timeBaseStatus</code> has changed from 1 to 0 0: otherwise
EV_TIMELEAP_PAST	1: TIMELEAP_PAST bit in <code>timeBaseStatus</code> has changed from 0 to 1 0: otherwise
EV_TIMELEAP_PAST_REMOVED	1: TIMELEAP_PAST bit in <code>timeBaseStatus</code> has changed from 1 to 0 0: otherwise
EV_SYNC_TO_SUBDOMAIN	1: SYNC_TO_GATEWAY bit in <code>timeBaseStatus</code> has changed from 0 to 1 0: otherwise
EV_SYNC_TO_GLOBAL_MASTER	1: SYNC_TO_GATEWAY bit in <code>timeBaseStatus</code> has changed from 1 to 0 0: otherwise
EV_RESYNC	1: resynchronization has occurred and a new time value has been applied 0: otherwise
EV_RATECORRECTION	1: a valid rate correction (not beyond limits) has been calculated 0: otherwise
EV_RATE_EXCEEDED	1: an invalid rate correction (beyond limits) has been calculated 0: otherwise
EV_PDELAY_EXCEEDED	1: PDELAY_EXCEEDED bit in <code>timeBaseStatus</code> has changed from 0 to 1 0: otherwise
EV_PDELAY_EXCEEDED_REMOVED	1: PDELAY_EXCEEDED bit in <code>timeBaseStatus</code> has changed from 1 to 0 0: otherwise
EV_RATEJITTERWANDER_EXCEEDED	1: RATEJITTERWANDER_EXCEEDED bit in <code>timeBaseStatus</code> has changed from 0 to 1 0: otherwise
EV_RATEJITTERWANDER_EXCEEDED_REMOVED	1: RATEJITTERWANDER_EXCEEDED bit in <code>timeBaseStatus</code> has changed from 1 to 0

	0: otherwise
EV_TIME_PROGRESSION_INCONSISTENCY	1: TIME_PROGRESSION_INCONSISTENCY bit in <code>timeBaseStatus</code> has changed from 0 to 1 0: otherwise
EV_TIME_PROGRESSION_INCONSISTENCY_REMOVED	1: TIME_PROGRESSION_INCONSISTENCY bit in <code>timeBaseStatus</code> has changed from 1 to 0 0: otherwise
EV_FALLBACK_TIME_EXTRAPOLATION	1: FALLBACK_TIME_EXTRAPOLATION bit in <code>timeBaseStatus</code> has changed from 0 to 1 0: otherwise
EV_FALLBACK_TIME_EXTRAPOLATION_REMOVED	1: FALLBACK_TIME_EXTRAPOLATION bit in <code>timeBaseStatus</code> has changed from 1 to 0 0: otherwise

]

[SWS_StbM_00278]

Upstream requirements: [RS_TS_00037](#)

[For Synchronized Time Bases the StbM shall use a variable `NotificationEvents` of type `StbM_TimeBaseNotificationType` to keep track, if any status event (refer to [\[SWS_StbM_00284\]](#)) for the referenced Time Base occurs.

If any status event occurs and the corresponding bit in the `StbMStatusNotificationMask` mask is set, the corresponding bit in the `NotificationEvents` variable is set, i.e., `NotificationEvents` can only contain bits for the events, which are enabled within the `StbMStatusNotificationMask` mask (refer to [\[SWS_StbM_00284\]](#)).]

[SWS_StbM_00282]

Upstream requirements: [RS_TS_00037](#)

[If any status event (refer to [\[SWS_StbM_00284\]](#)) occurs and the corresponding bit in the `StbMStatusNotificationMask` mask is set, the StbM shall report the value of the `NotificationEvents` variable

- via the callback function `StatusNotificationCallback<TimeBase>` (refer to parameter `eventNotification`) and/or
- via `StatusNotification` service interface (refer to data element `eventNotification`)

depending on the setting of parameter `StbMNotificationInterface`.

If multiple status events occur simultaneously for the same Time Base, the StbM shall trigger the callback function `StatusNotificationCallback<TimeBase>` and the `StatusNotification` service interface only once in a main cycle.]

Note: If e.g. a (re)synchronization takes place several of the following events may occur simultaneously:

- `EV_RESYNC`,
- `EV_TIMEOUT_REMOVED`,
- `EV_GLOBAL_TIME`,
- `EV_TIMELEAP_FUTURE`,
- `EV_TIMELEAP_PAST`,
- `EV_TIMELEAP_FUTURE_REMOVED`,
- `EV_TIMELEAP_PAST_REMOVED`,
- `EV_RATECORRECTION`,
- `EV_RATE_EXCEEDED`,
- `EV_SYNC_TO_SUBDOMAIN` and
- `EV_SYNC_TO_GLOBAL_MASTER`
- `EV_PDELAY_EXCEEDED`
- `EV_PDELAY_EXCEEDED_REMOVED`
- `EV_RATEJITTERWANDER_EXCEEDED`
- `EV_RATEJITTERWANDER_EXCEEDED_REMOVED`.

If on the other hand a time is retrieved following events may be observed:

- `EV_TIME_PROGRESSION_INCONSISTENCY`
- `EV_TIME_PROGRESSION_INCONSISTENCY_REMOVED`
- `EV_FALLBACK_TIME_EXTRAPOLATION`
- `EV_FALLBACK_TIME_EXTRAPOLATION_REMOVED`

because of a validation error or switch-over to the Fallback `Virtual Local Time`.

[SWS_StbM_00280]

Upstream requirements: [RS_TS_00016](#)

[After reporting a status event via the `StatusNotificationCallback<Time-Base>` API and the `StatusNotification` service interface the StbM shall reset `NotificationEvents` to 0.]

7.14 Triggering Customers

The OS provides the API `SyncScheduleTable` to synchronize a schedule table to a counter value.

[SWS_StbM_00020]

Upstream requirements: [SRS_BSW_00429](#), [RS_TS_00037](#), [RS_TS_00032](#)

[The Synchronized Time-Base Manager must be able to interact with the OS as Triggered Customer. The module calls the OS API for synchronizing OS ScheduleTables.]

[SWS_StbM_00022]

Upstream requirements: [RS_TS_00037](#), [RS_TS_00032](#)

[The Synchronized Time-Base Manager shall provide means to configure the Time Base to which the OS ScheduleTable should be synchronized (see container [StbMTriggeredCustomer](#)).]

The schedule table to be synchronized is given by [StbMOSScheduleTableRef](#) and the Time Base, which synchronizes the schedule table, is given by [StbMSynchronizedTimeBaseRef](#).

It is configurable at pre-compile time if an OS ScheduleTable shall be synchronized with a Synchronized Time Base.

[SWS_StbM_00084]

Upstream requirements: [RS_TS_00032](#)

[Customers of type Triggered Customer shall be invoked periodically by the Synchronized Time-Base Manager.]

[SWS_StbM_00031]

Upstream requirements: [RS_TS_00025](#)

[If a Triggered Customer is configured (refer to [StbMTriggeredCustomer](#)), the Synchronized Time-Base Manager shall monitor the cyclic execution of the [StbM_Main-Function](#).

This is to guarantee cyclic synchronization of OS schedule tables.]

[SWS_StbM_00093]

Upstream requirements: [RS_TS_00037](#), [RS_TS_00032](#)

[The triggering period [StbMTriggeredCustomerPeriod](#) shall be configurable for each Triggered Customer.]

Based on the configuration, the Synchronized Time-Base Manager synchronizes the OS counter value of the associated OS ScheduleTable.

[SWS_StbM_00302]

Upstream requirements: [RS_TS_00032](#)

[The StbM shall set the synchronization count of the OS ScheduleTable via `SyncScheduleTable`.]

The Synchronized Time-Base Manager is not responsible for starting and stopping the execution of OS ScheduleTables.

[SWS_StbM_00303]

Upstream requirements: [RS_TS_00037](#), [RS_TS_00032](#)

[The StbM shall derive the synchronization count of the OS ScheduleTable in microseconds by calculating the modulus of the current Time Base value (converted to microseconds) and `OsScheduleTableDuration` (refer to `OsScheduleTable` container referenced via `StbMOSScheduleTableRef`).]

Note: This requires, that the ticks of an OS counter, which drives a schedule table, have a duration of 1 us.

[SWS_StbM_00077]

Upstream requirements: [RS_TS_00032](#)

[The Synchronized Time-Base Manager shall synchronize OS ScheduleTables only when the associated Synchronized Time Base is synchronized, i.e., if

- `GLOBAL_TIME_BASE` = 1 and
- `TIMEOUT` = 0 and
- `SYNC_TO_GATEWAY` = 0 and
- `TIMELEAP_FUTURE` = 0 and
- `TIMELEAP_PAST` = 0

]

[SWS_StbM_00092]

Upstream requirements: [SRS_BSW_00429](#), [RS_TS_00032](#)

[The Synchronized Time-Base Manager shall check the OS for the status of the OS ScheduleTable by calling `GetScheduleTableStatus` before performing the synchronization.

The Synchronized Time-Base Manager shall synchronize only OS ScheduleTables that are in one of the states

- SCHEDULETABLE_WAITING,
- SCHEDULETABLE_RUNNING or
- SCHEDULETABLE_RUNNING_AND_SYNCHRONOUS.

]

Note: The Synchronized Time-Base Manager should ignore possible errors caused by the sequential execution of

1. getting OS ScheduleTable status and
2. performing the synchronization

(e.g., someone else might have called a service to stop the OS ScheduleTable in the meantime).

7.15 Time Recording

7.15.1 General

[SWS_StbM_00307]

Upstream requirements: [RS_TS_00034](#)

[The StbM shall support the Global Time precision measurement for a Time Base, if [StbMTimeRecordingSupport](#) is set to TRUE.]

7.15.2 Global Time Precision Measurement Support

To verify the precision of each Local Time Base compared to the Global Time Base a recording mechanism shall be optionally supported for Time Slaves and Time Gateways.

In principle, the StbM takes a snapshot of all required data at the point in time, where a synchronization event takes place. The StbM provides access to those values by an actively pushed API function on each successful assembled data block. An Off-Board Tester collects each block and calculates the precision afterwards and maintains a history of recorded blocks and their elements accordingly.

How and by which protocol the data will be transferred to the Off-Board Tester will be specified by the Application.

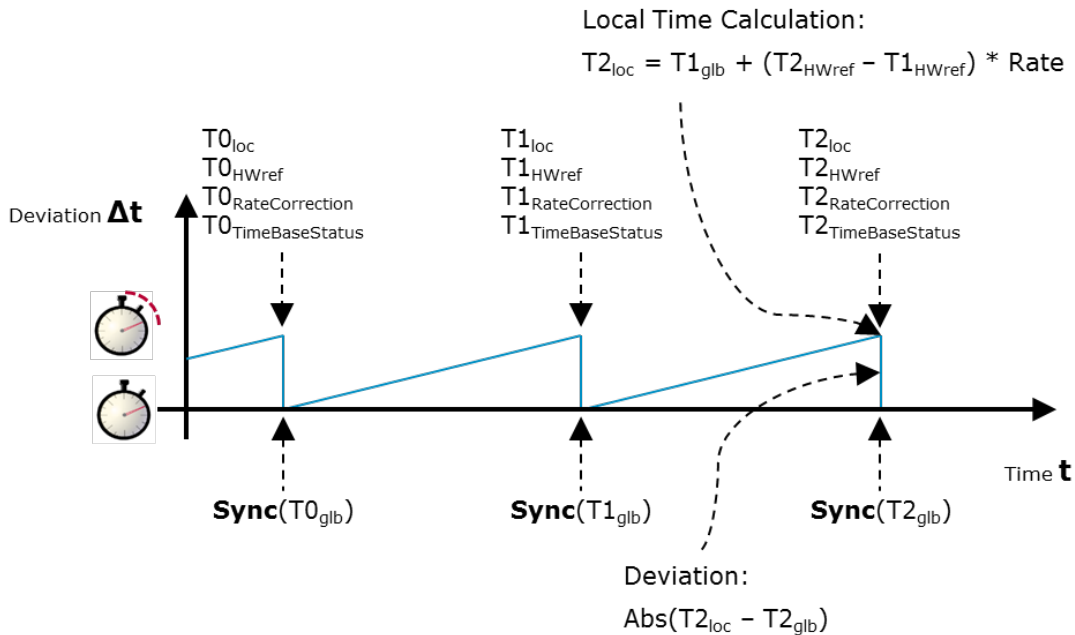


Figure 7.9: Simplified view how the recorded Time Base related snapshot data are taken

[SWS_StbM_00428]

Upstream requirements: [RS_TS_00034](#)

[The StbM shall do Global Time precision measurement only for Synchronized Time Bases for which [stbMIsSystemWideGlobalTimeMaster](#) is set to FALSE.]

7.15.2.1 Synchronized Time Base Record Table

[SWS_StbM_00308]

Upstream requirements: [RS_TS_00034](#)

[If Global Time Precision Measurement is enabled (refer to [\[SWS_StbM_00428\]](#) and [\[SWS_StbM_00307\]](#)) for the Synchronized Time Base, the StbM shall establish a Synchronized Time Base Record Table according to [\[SWS_StbM_00562\]](#) to record values of the Synchronized Time Base.]

[SWS_StbM_00562] Synchronized Time Base Record Table

Upstream requirements: [RS_TS_00034](#)

[

	Record Table Element	Multi- plicity	Range	Bytes	Type	Unit
Header		1		9		
	Synchronized-	1	0 .. 127	1	uint8	

	TimeDomain					
	HWfrequency	1	0 .. 4294967295	4	uint32	Hz
	HWprescaler	1	0 .. 4294967295	4	uint32	
Block 0		1		32		
	GlbSeconds	1	0 .. 4294967295	4	StbM_TimeStampType .seconds	sec
	GlbNanoSeconds	1	0 .. 9999999999	4	StbM_TimeStampType .nanoseconds	ns
	TimeBaseStatus	1	0 .. 65535	2	StbM_TimeBaseStatusType	
	VirtualLocalTimeLow	1	0 .. 4294967295	4	uint32	ns
	RateDeviation	1	0 .. +/-32000	2	StbM_Rate- DeviationType	ppm
	LocSeconds	1	0 .. 4294967295	4	StbM_TimeStampType .seconds	sec
	LocNanoSeconds	1	0 .. 9999999999	4	StbM_TimeStampType .nanoseconds	ns
	PathDelay	1	0 .. 4294967295	4	uint32	ns
	FallbackVirtualTimeLow	1	0 .. 4294967295	4	uint32	ns
Block 1	...	1				
...		1				
Block (Block-Count-1)	...	1				

]

[SWS_StbM_00309]

Upstream requirements: [RS_TS_00034](#)

[If Global Time Precision Measurement is enabled (refer to [\[SWS_StbM_00428\]](#) and [\[SWS_StbM_00307\]](#)) for the Time Base, `StbMClockFrequency` shall be mapped to the Header Element `HWfrequency` of the table belonging to the Synchronized Time Base unless the Virtual Local Time for the Time Base is provided by a Timesync module. In this case, `HWfrequency` shall be set to 1000000000.]

[SWS_StbM_00310]

Upstream requirements: [RS_TS_00034](#)

[If Global Time Precision Measurement is enabled (refer to [\[SWS_StbM_00428\]](#) and [\[SWS_StbM_00307\]](#)) for the Time Base, `StbMClockPrescaler` shall be mapped to the Header Element `HWprescaler` of the table belonging to the Synchronized Time Base unless the Virtual Local Time for the Time Base is provided by a Timesync module. In this case, `HWprescaler` shall be set to 1.]

[SWS_StbM_00382]

Upstream requirements: [RS_TS_00034](#)

[If Global Time Precision Measurement is enabled (refer to [\[SWS_StbM_00428\]](#) and [\[SWS_StbM_00307\]](#)) for the Time Base, the Synchronized Time Base Record Table (refer to [\[SWS_StbM_00562\]](#)) shall contain a history of as many blocks as configured by `StbMSyncTimeRecordTableBlockCount`.]

7.15.2.2 Snapshot Conditions**[SWS_StbM_00312]**

Upstream requirements: [RS_TS_00034](#)

[If Global Time Precision Measurement is enabled (refer to [\[SWS_StbM_00428\]](#) for the Time Base, on an invocation of `StbM_BusSetGlobalTime` the StbM shall update all elements of the block of the recording table.

If

- all blocks have been written
- and no notification via `SyncTimeRecordBlockCallback<TimeBase>` has yet occurred to pass all blocks with their elements to the application,

then the StbM shall again overwrite the block containing the oldest measurement data with the incoming measurement data.]

Note: From the implementation point of view, this mechanism belongs to a ring buffer concept in case data cannot be forwarded to the Application fast enough.

[SWS_StbM_00313]

Upstream requirements: [RS_TS_00034](#)

[For Synchronized Time Bases, if Global Time Precision Measurement is enabled (refer to [\[SWS_StbM_00428\]](#) and [\[SWS_StbM_00307\]](#)) for the Time Base, on an invocation of `StbM_BusSetGlobalTime` the StbM shall write the block elements

- `LocSeconds` and
- `LocNanoSeconds`

to the related Synchronized Time Base Record Table (refer to [\[SWS_StbM_00562\]](#) before updating the Main Time Tuple (i.e., updating the Local Time Base by the Global Time Base).

`LocSeconds` and `LocNanoSeconds` are the elements of the Global Time part of the Synclocal Time Tuple (i.e., `TLSync`, see [\[SWS_StbM_00438\]](#)).

[SWS_StbM_00314]

Upstream requirements: [RS_TS_00034](#)

[For Synchronized Time Bases, if Global Time Precision Measurement is enabled (refer to [\[SWS_StbM_00428\]](#) and [\[SWS_StbM_00307\]](#)) for the Time Base, on an invocation of `StbM_BusSetGlobalTime` the StbM shall write the block elements

- `GlbSeconds`,
- `GlbNanoSeconds`,
- `VirtualLocalTimeLow`,
- `RateDeviation`,
- `timeBaseStatus`
- `PathDelay`
- `FallbackVirtualTimeLow`

to the related Synchronized Time Base Record Table (refer to [\[SWS_StbM_00562\]](#) after updating the Main Time Tuple (i.e., after updating the Local Time Base by the Global Time Base).

`GlbSeconds` and `GlbNanoSeconds` are the elements of the Global Time part of the Updated Rx Time Tuple (i.e., TG_{Rx}); `VirtualLocalTimeLow` is the `nanosecondsLo` element of the Virtual Local Time part of the Updated Rx Time Tuple (i.e., TV_{Rx}).]

Note: `PathDelay` will be retrieved from the `<Bus>TSyn` module as `pathDelay` member of parameter `measureDataPtr` of `StbM_BusSetGlobalTime`.

[SWS_StbM_00315]

Upstream requirements: [RS_TS_00034](#)

[If Global Time Precision Measurement is enabled (refer to [\[SWS_StbM_00428\]](#) and [\[SWS_StbM_00307\]](#)) for the Time Base, the application collects the contents of the header of the Synchronized Time Base Record Table (refer to [\[SWS_StbM_00562\]](#) by calling `StbM_GetSyncTimeRecordHead`.]

[SWS_StbM_00317]

Upstream requirements: [RS_TS_00034](#)

[If Global Time Precision Measurement is enabled for the Time Base (refer to [\[SWS_StbM_00428\]](#) and [\[SWS_StbM_00307\]](#)), the StbM shall notify the Application by calling `SyncTimeRecordBlockCallback<TimeBase>` in the next `StbM_MainFunction` call cycle block by block (i.e., repeatedly) for all unread blocks (i.e., containing data that has yet not been passed to the Application), starting with the block containing the oldest data, and followed by the blocks in ascending order regarding their age (i.e., FIFO order), the block containing the newest data shall be passed last.

The StbM shall ensure data integrity: a block shall not be passed if it currently being overwritten and a block that is passed shall be prevented from being overwritten until processed by the Application.]

7.16 Time Validation

Figure 7.10 outlines the basic concept of the "Time Validation" feature. Time Slaves, Time Masters and Time Gateways collect information on the time synchronization process from the corresponding Timesync modules, to allow for, e.g. predicting the Global Time of Sync ingress events based on their local instance of the Global Time (by using the Synclocal Time Tuple) and make this information available to the application (i.e. to an SWC). In doing so one application can check peer-wise whether a Master and a neighboring Slave agree upon the current Global Time.

The predictions, etc. may be locally analyzed by the application to detect any impairments quickly with the desired safety integrity. Furthermore, information on the time synchronization process between all Time Masters and Slaves that participate in the "Time Validation" is also shared with a Validator SWC which may run anywhere in the network, e.g. on the Global Time Master. The Validator SWC has therefore global system view which allows the Validator to check whether a coherent time base is established among all peers or not. The Validator constitutes simultaneously a single authorization instance that can assess the safety integrity of the overall system with the desired ASIL. The Validator receives the necessary information from all entities via a user defined feedback channel.

The Time Validation feature only provides service interfaces to the application. The feedback channel and the actual validation performed by the respective SWCs is not standardized in AUTOSAR. It is done in a user defined way on application level.

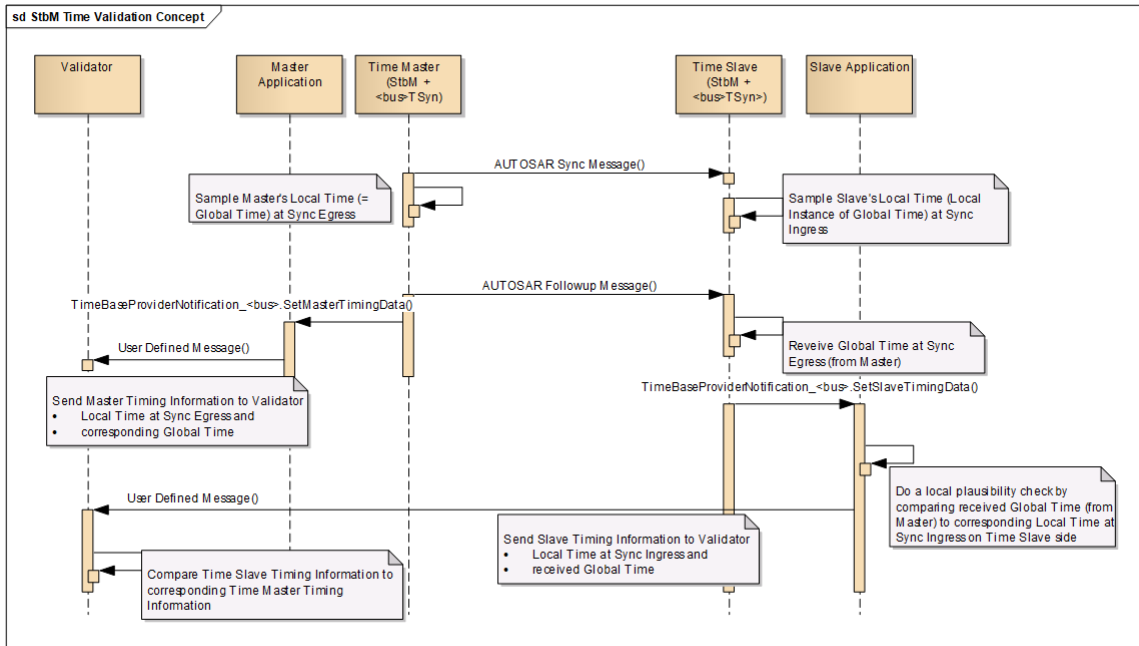


Figure 7.10: : Concept of Time Validation

[SWS_StbM_00465]

Upstream requirements: [RS_TS_00034](#)

[The StbM shall record timing data for Time Validation for Synchronized Time Bases (refer to [StbMSynchronizedTimeBase](#)), which have Time Validation enabled (i.e., [StbMTimeValidation](#) is configured).]

7.16.1 Record Tables

7.16.1.1 Time Slave/Master

[SWS_StbM_00466]

Upstream requirements: [RS_TS_00034](#)

[For each Time Base,

- which has Time Validation enabled (refer to [\[SWS_StbM_00465\]](#)) and
- which is mapped to a Slave Communication Port in a Timesync module

the StbM shall manage a Time Slave Validation Record Table (refer to [\[SWS_StbM_00557\]](#)), which holds

- N blocks (refer to [\[SWS_StbM_00470\]](#))

- of type `StbM_<bus>TimeSlaveMeasurementType` (refer to [\[SWS_StbM_00467\]](#)).

]

[SWS_StbM_00557] Time Slave Validation Record Table

Upstream requirements: [RS_TS_00034](#)

[

	Block Structure Element	Type	Description
Block 0	<code><bus>Slave-TimingData[0]</code>	<code>StbM_<bus>TimeSlave-MeasurementType</code>	Bus specific structure to capture Time Slave Validation recording data
Block 1	<code><bus>Slave-TimingData[1]</code>	<code>StbM_<bus>TimeSlave-MeasurementType</code>	
...			
Block N (= <code>StbMTime-Validation-RecordTable-BlockCount-1</code>)			

]

The type of the blocks in the Time Slave Validation Record Table (refer to [\[SWS_StbM_00557\]](#)) depend on the Timesync Module, which provides the data for the Time Slave Validation Record Table of the corresponding Time Base. The mapping of block type to providing Timesync Module (and respective API) is given in [\[SWS_StbM_00467\]](#).

[SWS_StbM_00467] Type Mapping for Time Slave Validation Record Table

Upstream requirements: [RS_TS_00034](#)

[

Timesync Module (which provides timing record data)	Type (of block in the "Slave Validation Record Table")
CanTSyn (by <code>StbM_CanSetSlaveTimingData</code>)	<code>StbM_CanTimeSlaveMeasurementType</code>
FrTSyn (by <code>StbM_FrSetSlaveTimingData</code>)	<code>StbM_FrTimeSlaveMeasurementType</code>
EthTSyn (by <code>StbM_EthSetSlaveTimingData</code>)	<code>StbM_EthTimeSlaveMeasurementType</code>

]

[SWS_StbM_00468]

Upstream requirements: [RS_TS_00034](#)

[For each Time Base,

- which has Time Validation enabled (refer to [\[SWS_StbM_00465\]](#)) and
- which is mapped to a Master Communication Port in a Timesync module

the StbM shall manage a Time Master Validation Record Table (refer to [\[SWS_StbM_00560\]](#)), which holds

- N blocks (refer to [\[SWS_StbM_00470\]](#))
- of type `StbM_<bus>TimeMasterMeasurementType` (refer to [\[SWS_StbM_00469\]](#)).

]

[SWS_StbM_00560] Time Master Validation Record Table

Upstream requirements: [RS_TS_00034](#)

[

	Block Structure Element	Type	Description
Block 0	<code><bus>Master-TimingData[0]</code>	<code>StbM_<bus>TimeMaster-MeasurementType</code>	Bus specific structure to capture Validation recording data of a Time Master
Block 1	<code><bus>Master-TimingData[1]</code>	<code>StbM_<bus>TimeMaster-MeasurementType</code>	
...			
Block N (= <code>StbMTime-Validation-RecordTable-BlockCount-1</code>)			

]

The type of the blocks in the Time Master Validation Record Table (refer to [\[SWS_StbM_00560\]](#)) shall depend on the Timesync Module, which provides the data for the Time Master Validation Record Table of the corresponding Time Base. The mapping of block type to providing Timesync Module (and respective API) is given in [\[SWS_StbM_00469\]](#).

[SWS_StbM_00469] Type Mapping for Time Master Validation Record Table

Upstream requirements: [RS_TS_00034](#)

[

Timesync Module (which provides timing record data)	Type (of block in the "Master Validation Record Table")
CanTSyn (by StbM_CanSetMasterTimingData)	StbM_CanTimeMasterMeasurementType
FrTSyn (by StbM_FrSetMasterTimingData)	StbM_FrTimeMasterMeasurementType
EthTSyn (by StbM_EthSetMasterTimingData)	StbM_EthTimeMasterMeasurementType

]

Note: The `<bus>TSynSynchronizedTimeBaseRef` parameter in the configuration of the Timesync Modules defines, which Timesync module is linked to which Time Base in the StbM, and hence determines which Timesync Module provides the data for the Time Master Validation Record Table / Time Slave Validation Record Table of the Time Base.

Note: If the StbM is configured to be a Time Gateway for a Time Base with Time Validation enabled, the StbM maintains

- one Time Master Validation Record Table
- and one Time Slave Validation Record Table

for that Time Base.

7.16.1.2 Pdelay Initiator/Responder

[SWS_StbM_00522]

Upstream requirements: [RS_TS_00034](#)

[For each Time Base, which

- has Time Validation enabled (refer to [\[SWS_StbM_00465\]](#)) and
- is mapped to a Slave Communication Port on an Ethernet Time Domain

the StbM shall manage a Pdelay Initiator Validation Record Table (refer to [\[SWS_StbM_00558\]](#)), which holds

- N blocks (refer to [\[SWS_StbM_00470\]](#))
- of type [StbM_PdelayInitiatorMeasurementType](#).

]

[SWS_StbM_00558] Pdelay Initiator Validation Record Table

Upstream requirements: [RS_TS_00034](#)

[

	Block Structure Element	Type	Description
Block 0	pdelayInitiatorData[0]	StbM_PdelayInitiator-MeasurementType	Structure to capture Time Validation recording data of a PdelayInitiator
Block 1	pdelayInitiatorData[1]	StbM_PdelayInitiator-MeasurementType	
...			
Block N (= StbMTime-Validation-RecordTable-BlockCount-1)			

]

[SWS_StbM_00523]

Upstream requirements: [RS_TS_00034](#)

[For each Time Base, which

- has Time Validation enabled (refer to [\[SWS_StbM_00465\]](#)) and
- is mapped to a Master Communication Port on an Ethernet Time Domain

the StbM shall manage a Pdelay Responder Validation Record Table (refer to [\[SWS_StbM_00559\]](#)), which holds

- N blocks (refer to [\[SWS_StbM_00470\]](#))
- of type [StbM_PdelayResponderMeasurementType](#).

]

[SWS_StbM_00559] Pdelay Responder Validation Record Table

Upstream requirements: [RS_TS_00034](#)

[

	Block Structure Element	Type	Description
Block 0	pdelayResponderData[0]	StbM_PdelayResponder-MeasurementType	Structure to capture Time Validation recording data of a PdelayResponder
Block 1	pdelayResponderData[1]	StbM_PdelayResponder-MeasurementType	
...			

Block N (= StbMTime- Validation- RecordTable- BlockCount-1)			
--	--	--	--

]

[SWS_StbM_00624]

Status: DRAFT
 Upstream requirements: [RS_TS_00034](#)

[For each Time Base if

- Time Validation is enabled (refer to [\[SWS_StbM_00465\]](#))
- and `pathDelay` member of parameter `measureDataPtr` of `StbM_BusSet-GlobalTime` greater than `StbMPdelayValidationThreshold`

then the StbM shall set the `PDELAY_EXCEEDED` bit in the `timeBaseStatus` of the corresponding Time Base.

If `pathDelay` is less than or equal to the `StbMPdelayValidationThreshold`, StbM shall reset the `PDELAY_EXCEEDED` bit in the `timeBaseStatus` of the corresponding Time Base.]

7.16.1.3 Common

[SWS_StbM_00470]

Upstream requirements: [RS_TS_00034](#)

[Each

- Slave Validation Record Table (refer to [\[SWS_StbM_00557\]](#))
- Master Validation Record Table (refer to [\[SWS_StbM_00560\]](#))
- Pdelay Initiator Validation Record Table (refer to [\[SWS_StbM_00558\]](#))
- Pdelay Responder Validation Record Table (refer to [\[SWS_StbM_00559\]](#))

shall contain as many blocks as configured by `StbMTimeValidation-RecordTableBlockCount`.]

7.16.2 Time Validation Snapshot Conditions

7.16.2.1 Time Slave/Master

[SWS_StbM_00471]

Upstream requirements: [RS_TS_00034](#)

[If Time Validation is enabled for a Time Base (refer to [\[SWS_StbM_00465\]](#)), upon invocation of `StbM_<bus>SetSlaveTimingData()` the StbM shall copy the content of the structure, which is passed by parameter `measureDataPtr`, to the next free `<bus>SlaveTimingData` block of the "Slave Validation Record Table" of that Time Base (refer to [\[SWS_StbM_00466\]](#)).

The StbM shall then shall set the value of the block element

- `<bus>SlaveTimingData.referenceGlobalTimestamp` as TL_{SYNC} (refer to [\[SWS_StbM_00438\]](#)) and
- `<bus>SlaveTimingData.referenceLocalTimestamp` as TV_{SYNC} (refer to [\[SWS_StbM_00438\]](#)).

(i.e., to the value of the Synclocal Time Tuple as set by the preceding call of `StbM_BusSetGlobalTime`).

[SWS_StbM_00472]

Upstream requirements: [RS_TS_00034](#)

[If no free block is available in the Slave Validation Record Table of a Time Base (refer to [\[SWS_StbM_00466\]](#)) (i.e., all blocks have been written and no notification via operation `SetSlaveTimingData` of port `TimeBaseProviderNotification_{bus}_{TimeBase}` has yet occurred to pass all blocks to the application), the StbM shall overwrite the block containing the oldest measurement data upon invocation of `StbM_<bus>SetSlaveTimingData`].

[SWS_StbM_00473]

Upstream requirements: [RS_TS_00034](#)

[If Time Validation is enabled for a Time Base (refer to [\[SWS_StbM_00465\]](#)), upon invocation of `StbM_<bus>SetMasterTimingData` the StbM shall copy the content of the structure, which is passed by parameter `measureDataPtr`, to the next free block `<bus>MasterTimingData` of the Master Validation Record Table of that Time Base (refer to [\[SWS_StbM_00468\]](#)).

[SWS_StbM_00474]

Upstream requirements: [RS_TS_00034](#)

[If no free block is available in a Master Validation Record Table of a Time Base (refer to [\[SWS_StbM_00468\]](#)) (i.e., all blocks have been written and no

notification via `SetMasterTimingData` of port `TimeBaseProviderNotification_{bus}_{TB_Name}` has yet occurred to pass all blocks to the application), the StbM shall overwrite the block containing the oldest measurement data upon invocation of `StbM_{bus}SetMasterTimingData`.]

Note: From the implementation point of view, this mechanism belongs to a ring buffer concept in case data cannot be forwarded to the application fast enough.

[SWS_StbM_00475]

Upstream requirements: [RS_TS_00034](#)

[For each Time Base,

- which has Time Validation enabled (refer to [\[SWS_StbM_00465\]](#)) and
- for which the StbM is configured as a Time Slave or Time Gateway

the StbM shall check within each `StbM_MainFunction` call, if new blocks (i.e., containing data that has not yet been passed to the application) have been written in the Slave Validation Record Table (refer to [\[SWS_StbM_00466\]](#)).

If so, the StbM shall pass all new blocks to the application by (repeatedly, block by block) calling operation `SetSlaveTimingData` of port `TimeBaseProviderNotification_{bus}_{TimeBase}`.

The StbM shall pass the blocks starting with the block containing the oldest data, and followed by the blocks in ascending order regarding their age (i.e., FIFO order). The block containing the newest data shall be passed last.]

[SWS_StbM_00476]

Upstream requirements: [RS_TS_00034](#)

[For each Time Base,

- which has Time Validation enabled (refer to [\[SWS_StbM_00465\]](#)) and
- for which the StbM is configured as a Time Master or Time Gateway

the StbM shall check within each `StbM_MainFunction`, if new blocks (i.e., containing data that has not yet been passed to the application) have been written to the Master Validation Record Table (refer to [\[SWS_StbM_00468\]](#)).

If so, the StbM shall pass all new blocks to the application by (repeatedly, block by block) calling operation `SetMasterTimingData` of port `TimeBaseProviderNotification_{bus}_{TimeBase}`.

The StbM shall pass the blocks starting with the block containing the oldest data, and followed by the blocks in ascending order regarding their age (i.e., FIFO order). The block containing the newest data shall be passed last.]

7.16.2.2 Pdelay Initiator/Responder

[SWS_StbM_00478]

Upstream requirements: [RS_TS_00034](#)

[If Time Validation is enabled for a Time Base (refer to [\[SWS_StbM_00465\]](#)), upon invocation of [StbM_EthSetPdelayInitiatorData](#) the StbM shall write the content of the structure, which is passed by parameter [measureDataPtr](#), to the next free block [pdelayInitiatorData](#) of the corresponding Pdelay Initiator Validation Record Table of that Time Base (refer to [\[SWS_StbM_00522\]](#).)]

[SWS_StbM_00524]

Upstream requirements: [RS_TS_00034](#)

[If no free block is available in the Pdelay Initiator Validation Record Table of a Time Base (refer to [\[SWS_StbM_00522\]](#)) (i.e., all blocks have been written and no notification via operation [SetPdelayInitiatorData](#) of port [TimeBaseProviderNotification_Eth](#) has yet occurred to pass all blocks to the application), the StbM shall overwrite the block containing the oldest measurement data upon invocation of [StbM_EthSetPdelayInitiatorData](#).]

[SWS_StbM_00479]

Upstream requirements: [RS_TS_00034](#)

[For each Time Base, which

- has Time Validation enabled (refer to [\[SWS_StbM_00465\]](#)) and
- is mapped to a Slave Communication Port on an Ethernet Time Domain,

the StbM shall check within each [StbM_MainFunction](#), if new blocks (i.e., containing data that has not yet been passed to the application) have been written to the Pdelay Initiator Validation Record Table of a Time Base (refer to [\[SWS_StbM_00522\]](#)).

If so, the StbM shall pass all new blocks to the application by (repeatedly, block by block) calling operation [SetPdelayInitiatorData](#) of port [TimeBaseProviderNotification_Eth](#).

The StbM shall pass the blocks starting with the block containing the oldest data and followed by the blocks in ascending order regarding their age (i.e., FIFO order). The block containing the newest data shall be passed last.]

[SWS_StbM_00480]

Upstream requirements: [RS_TS_00034](#)

[If Time Validation is enabled for a Time Base (refer to [\[SWS_StbM_00465\]](#)), upon invocation of [StbM_EthSetPdelayResponderData](#) the StbM shall write the content

of the structure, which is passed by parameter `measureDataPtr`, to the next free block `PdelayResponderData` of the corresponding `Pdelay Responder Validation Record Table` of that Time Base (refer to [SWS_StbM_00523].)

[SWS_StbM_00525]

Upstream requirements: RS_TS_00034

[If no free block is available in the `Pdelay Responder Validation Record Table` of a Time Base (refer to [SWS_StbM_00523] (i.e., all blocks have been written and no notification via operation `SetPdelayResponderData` of port `TimeBaseProviderNotification_Eth` has yet occurred to pass all blocks to the application), the StbM shall overwrite the block containing the oldest measurement data upon invocation of `StbM_EthSetPdelayResponderData`.]

[SWS_StbM_00481]

Upstream requirements: RS_TS_00034

[For each Time Base, which

- has Time Validation enabled (refer to [SWS_StbM_00465]) and
- is mapped to a Master Communication Port on an Ethernet Time Domain,

the StbM shall check within each `StbM_MainFunction`, if new blocks (i.e., containing data that has not yet been passed to the application) have been written to the `Pdelay Responder Validation Record Table` (refer to [SWS_StbM_00523].

If so, the StbM shall pass all new blocks to the application by (repeatedly, block by block) calling operation `SetPdelayResponderData` of port `TimeBaseProviderNotification_Eth`.

The StbM shall pass the blocks starting with the block containing the oldest data and followed by the blocks in ascending order regarding their age (i.e., FIFO order). The block containing the newest data shall be passed last.]

7.16.2.3 Common

[SWS_StbM_00477]

Upstream requirements: RS_TS_00034

[The StbM shall ensure data integrity of the blocks in the

- Slave Validation Record Table (refer to [SWS_StbM_00466])
- Master Validation Record Table (refer to [SWS_StbM_00468])

- Pdelay Initiator Validation Record Table (refer to [SWS_StbM_00522])
- Pdelay Responder Validation Record Table (refer to [SWS_StbM_00523]).

If a block is currently being overwritten, it shall not be passed to the application.

If a block is currently passed to the application, it shall not be overwritten until processed by the application.]

7.16.3 Monitoring the progression of the Local Instance of Global Time

If the Primary and Fallback *Virtual Local Time* both are present, the StbM can use the the Fallback *Virtual Local Time* to monitor the Primary *Local Instance of the Global Time* and with that the underlying Primary *Virtual Local Time*.

If the StbM detects any inconsistency it raises a status flag.

[SWS_StbM_00599]

Status: DRAFT

Upstream requirements: RS_TS_00041

[For Synchronized and a Pure Local Time Base if

- Time Validation is enabled
- and the Fallback *Virtual Local Time* is valid (i.e., *StbM_GetFallbackVirtualLocalTime* returns E_OK),
- and the Primary *Virtual Local Time* is valid (i.e., *StbM_GetCurrentVirtualLocalTime* returns E_OK),

when the StbM is about to update the *Main Time Tuple* for the Time Base (refer to [SWS_StbM_00433]) then the StbM shall in that order

1. save the current *Main Time Triple* of the Time Base as the **previous** *Main Time Triple* [$TL_{Main(prev)}$, $TV_{Main(prev)}$, $TV_{Fallback_Main(prev)}$]
2. update the **current** *Main Time Tuple* of the Time Base (refer to [SWS_StbM_00433])

]

Note: The previous *Main Time Triple* is only updated when the Primary *Virtual Local Time* is available.

[SWS_StbM_00596]*Status:* DRAFT*Upstream requirements:* [RS_TS_00040](#)

[For Synchronized and a Pure Local Time Base if

- Time Validation is enabled
- and the Primary [Virtual Local Time](#) is valid (refer to [\[SWS_StbM_00437\]](#))
- and the Fallback [Virtual Local Time](#) is valid (refer to [\[SWS_StbM_00595\]](#)),

when the StbM updates the [Local Instance of the Global Time](#) (TL) of the Time Basethen the StbM shall calculate a value $TL_{Monitor}$ for the Time Base according to

$$TL_{Monitor} = TL_{Main(prev)} + (TV_{Fallback} - TV_{Fallback_Main(prev)}) \quad (7.4)$$

using the **previous Main Time Tuple** [$TL_{Main(prev)}$, $TV_{Main(prev)}$, $TV_{Fallback_Main(prev)}$] (refer to [\[SWS_StbM_00599\]](#))

]

Rationale: $TL_{Monitor}$ is calculated based on the **previous Main Time Triple** to achieve freedom from interference, i.e., **current Main Time Triple** and **previous Main Time Triple** are separated in memory and if one got corrupted, this could still be detected based on the other one. Note however that there are also other means to achieve freedom from interference.

Note: Extrapolation for monitoring purposes currently does not support rate correction.

[SWS_StbM_00597]*Status:* DRAFT*Upstream requirements:* [RS_TS_00040](#)

[For a Time Base

if Time Validation is enabled (i.e., [StbMTimeValidation](#) is configured)when the monitoring value ($TL_{Monitor}$) has been updated (refer to [\[SWS_StbM_00596\]](#)),then the StbM shall compare $TL_{Monitor}$ to the current value of the [Local Instance of the Global Time](#) (TL).

If the absolute value of the difference between TL and $TL_{Monitor}$ is greater than the threshold referenced by parameter [StbMMaxProgressionMismatchThreshold](#), then the StbM shall set the [TIME_PROGRESSION_INCONSISTENCY](#) bit in `timebaseStatus`, else the StbM shall reset the [TIME_PROGRESSION_INCONSISTENCY](#) bit.]

[SWS_StbM_00603]*Status:* DRAFT*Upstream requirements:* [RS_TS_00043](#)

[For a Time Base if

- Time Validation is enabled (i.e., [StbMTimeValidation](#) is configured)
- and the Primary [Virtual Local Time](#) is available
- and Fallback [Virtual Local Time](#) is available

then [StbM_GetCurrentSafeTime](#) shall return `E_OK` and

- the current [Local Instance of the Global Time](#),
- the current Primary [Virtual Local Time](#),
- the current Fallback [Virtual Local Time](#),
- the current `timeBaseStatus`.

]

[SWS_StbM_00604]*Status:* DRAFT*Upstream requirements:* [RS_TS_00043](#)

[For a Time Base if

- Time Validation is not enabled (i.e., [StbMTimeValidation](#) is not configured)
- or the Primary or Fallback [Virtual Local Time](#) are not available

then [StbM_GetCurrentSafeTime](#) shall return `E_NOT_OK`.]

7.16.4 Rate validation

Rate Validation is to establish a robust framework for clock synchronization, particularly in scenarios involving multiple clocks.

Clocks in a system can experience variations in their rates from effects like rate jitter, or rate wander. Jitter describes a short-term frequency variation while the term wander is typically used to describe rather long-term frequency variations or drift. These variations, if left unchecked, could lead to inaccurate timekeeping and synchronization issues.

- **Master Clock (Global Time) vs. Local Clock (Virtual Local Time):**

By comparing the elapsed time measurements ($\Delta TG_{\text{Validation}}$) from the Global Time Master clock and ($\Delta TV_{\text{Validation}}$) from the local clock of a Time Slave (which

is inherently done upon rate correction), the system can assess if the clock deviation between Master and Slave is exceeding a predefined threshold. This check helps to catch potential discontinuities in the local clock's timekeeping capabilities. A central entity could even figure out whether the Master or a Slave clock is running unstable by evaluating measurements from several Slave against the Global Time Master.

- **Master Clock (Global Time) vs. Synchronized Clock (Local Instance of Global Time):**

The process involves comparing the elapsed time measurements of the master clock ($\Delta TG_{\text{Validation}}$) with the synchronized clock ($\Delta TL_{\text{Validation}}$). This comparison verifies the accuracy of the synchronization process and identifies any discrepancies between the master clock's time and the synchronized time.

- **Local Clock (Virtual Local Time) vs. Synchronized Clock (Local Instance of Global Time):**

Another comparison is made between the elapsed time measurements of the local clock ($\Delta TV_{\text{Validation}}$) and the synchronized clock ($\Delta TL_{\text{Validation}}$). This check ensures that the local clock aligns with the synchronized time, contributing to consistent and reliable synchronization.

The objective is to create a comprehensive validation mechanism that confirms the alignment of different clocks, promoting accurate timekeeping and synchronization across the system. Upon receiving slave timing data via (`StbM_BusSetGlobalTime`) from the Timesync module, StbM will perform rate validation if time validation support is on. Rate validation includes check for

- **Rate Jitter** according to [7.16.4.1 "Calculation of Rate Jitter"](#)
- and **Rate Wander** according to [7.16.4.2 "Calculation of Rate Wander"](#).

7.16.4.1 Calculation of Rate Jitter

Rate Jitter refers to the variation or fluctuation in the rate at which a clock's time drifts or deviates from the ideal time reference over time.

[SWS_StbM_00612]

Status: DRAFT

Upstream requirements: [RS_TS_00043](#)

[For a Time Slave of a Synchronized Time Base, if Time Validation is enabled, then the StbM shall calculate elapsed Global Time $\Delta TG_{\text{Validation}_1}$ by using following formula:

$$\Delta TG_{\text{Validation}_1} = TG_{Rx(i)} - TG_{Rx(i-1)} \quad (7.5)$$

with:

- $TG_{Rx(i)}$: Value of Global Time member of the latest [Rx Time Tuple](#)
- $TG_{Rx(i)}$: Value of Global Time member of the previous [Rx Time Tuple](#)

]

[SWS_StbM_00613]

Status: DRAFT

Upstream requirements: [RS_TS_00043](#)

[For a Time Slave of a Synchronized Time Base if Time Validation is enabled, then the StbM shall calculate elapsed Virtual Local Time $TV_{Validation_1}$ by using following formula:

$$\Delta TV_{Validation_1} = TV_{Rx(i)} - TV_{Rx(i-1)} \quad (7.6)$$

with:

- $TV_{Rx(i)}$: Value of Local Time member of the latest [Rx Time Tuple](#)
- $TV_{Rx(i-1)}$: Value of Local Time member of the previous [Rx Time Tuple](#)

]

[SWS_StbM_00614]

Status: DRAFT

Upstream requirements: [RS_TS_00043](#)

[For a Time Slave of a Synchronized Time Base, if Time Validation is enabled, then the StbM shall calculate elapsed Local Instance of the Global Time $TL_{Validation_1}$ by using following formula:

$$TL_{Validation_1} = TL_{Sync(i)} - TL_{Sync(i-1)} \quad (7.7)$$

with:

- $TL_{Sync(i)}$: Global Time member of the current [SyncLocal Time Tuple](#).
- $TL_{Sync(i-1)}$: Local Time member of the previous [SyncLocal Time Tuple](#)

]

For rate jitter sample calculation StbM will check both,

- the progression of its [Local Instance of the Global Time](#)
- and the Global Time Master's time with its own local clock.

[SWS_StbM_00615]

Status: DRAFT

Upstream requirements: [RS_TS_00043](#)

[For a Time Slave of a Synchronized Time Base, if

- the Time Validation is enabled
- and
 - the absolute value of rate deviation

$$\frac{TG_{Validation_1}}{TL_{Validation_1}} - 1 \quad (7.8)$$

- or the absolute value of the rate deviation

$$\frac{TV_{Validation_1}}{TL_{Validation_1}} - 1 \quad (7.9)$$

exceeds the [StbMRateJitterThreshold](#),

then the StbM shall set the [RATEJITTERWANDER_EXCEEDED](#) bit within `timeBaseStatus`.]

[SWS_StbM_00616]

Status: DRAFT

Upstream requirements: [RS_TS_00040](#), [RS_TS_00041](#), [RS_TS_00043](#)

[If any one of the values

- $TL_{Sync(i-1)}$
- or $TG_{Rx(i-1)}$
- or $TV_{Rx(i-1)}$

is not available, then the StbM shall not check for rate jitter.]

7.16.4.2 Calculation of Rate Wander

The overall Rate Wander is given by aggregating several "Rate Jitter Samples/Measurements". However, for validation purposes it is sufficient to consider only the instantaneous variation by evaluating the current value of the Rate Jitter in between two consecutive synchronizations (see [7.16.4.1 "Calculation of Rate Jitter"](#)) and the current value of the rate wander, i.e., the frequency variation measured via an interval of N re-synchronizations.

For StbM shall check both the progression of its [Local Instance of the Global Time](#) and the Global Time Master's time with its [Virtual Local Time](#) to calculate rate wander.

[SWS_StbM_00617]

Status: DRAFT
Upstream requirements: [RS_TS_00043](#)

[For a Time Slave of a Synchronized Time Base, if Time Validation is enabled, then the StbM shall further calculate elapsed global time at Global Time Master $\Delta TG_{Validation_N}$ as:

$$\Delta TG_{Validation_N} = TG_{Rx(i)} - TG_{Rx(i-N)} \quad (7.10)$$

with

- $TG_{Rx(i)}$: Value of Global Time member of the latest [Rx Time Tuple](#)
- $TG_{Rx(i-N)}$: Value of Global Time member of the N^{th} [Rx Time Tuple](#) before the current one
- N : Index shift represented by parameter [StbMRateWanderIntervalWindow](#) that defines which sample of [Rx Time Tuple](#) to take.

]

[SWS_StbM_00618]

Status: DRAFT
Upstream requirements: [RS_TS_00043](#)

[For a Time Slave of a Synchronized Time Base, if Time Validation is enabled, then the StbM shall calculate elapsed [Virtual Local Time](#) $TV_{Validation_N}$ as:

$$\Delta TV_{Validation_N} = TV_{Rx(i)} - TV_{Rx(i-N)} \quad (7.11)$$

with:

- $TG_{Rx(i)}$: Value of Local Time member of the latest [Rx Time Tuple](#)
- $TG_{Rx(i-N)}$: Value of Local Time member of the N^{th} [Rx Time Tuple](#) before the current one
- NN : Index shift as given by parameter [StbMRateWanderIntervalWindow](#) that defines which sample of [Rx Time Tuple](#) to take.

]

[SWS_StbM_00619]

Status: DRAFT
Upstream requirements: [RS_TS_00043](#)

[For a Time Slave of a Synchronized Time Base, if Time Validation is enabled, then the StbM shall calculate elapsed [Local Instance of the Global Time](#) $TL_{Validation_N}$ as:

$$\Delta TL_{Validation_N} = TL_{Sync(i)} - TL_{Sync(i-N)} \quad (7.12)$$

with:

- $TL_{Sync(i)}$: Value of the Global Time member of the latest [SyncLocal Time Tuple](#).
- $TL_{Sync(i-N)}$: Value of Global Time member of the N^{th} [SyncLocal Time Tuple](#) before the current one
- N : Index shift represented by parameter [StbMRateWanderIntervalWindow](#) that defines which sample of the [SyncLocal Time Tuple](#) to take.

]

Note: The i^{th} and $(i-N)^{\text{th}}$ samples of TG_{Rx} , TV_{Rx} or TL_{Sync} , respectively, are not two consecutive values received by [StbM_BusSetGlobalTime](#), as rate wander monitors rate fluctuations over longer duration of time.

[SWS_StbM_00620]

Status: DRAFT

Upstream requirements: [RS_TS_00043](#)

[If value of N , i.e., [StbMRateWanderIntervalWindow](#) is configured as 0 StbM shall disable the rate wander calculation.]

[SWS_StbM_00621]

Status: DRAFT

Upstream requirements: [RS_TS_00043](#)

[If any one of the values

- $TL_{Sync(i-N)}$
- or $TG_{Rx(i-N)}$
- or $TV_{Rx(i-N)}$

is not available, then the StbM shall not check for rate wander.]

[SWS_StbM_00622]

Status: DRAFT

Upstream requirements: [RS_TS_00040](#), [RS_TS_00041](#), [RS_TS_00043](#)

[For a Time Slave of a Synchronized Time Base, if

- the Time Validation is enabled
- and

- the absolute value of the rate deviation

$$\frac{\Delta TG_{Validation_N}}{\Delta TL_{Validation_N}} - 1 \quad (7.13)$$

- or the absolute value of the rate deviation

$$\frac{\Delta TV_{Validation_N}}{\Delta TL_{Validation_N}} - 1 \quad (7.14)$$

exceeds the `StbMRateWanderThreshold`,

then the StbM shall set the `RATEJITTERWANDER_EXCEEDED` bit within `StbM_TimeBaseStatus`.]

[SWS_StbM_00623]

Status: DRAFT

Upstream requirements: [RS_TS_00043](#)

[If and only if both errors

- Rate Jitter error (refer to [\[SWS_StbM_00615\]](#))
- and Rate Wander error (refer to [\[SWS_StbM_00622\]](#))

are absent, then the StbM shall reset the `RATEJITTERWANDER_EXCEEDED` flag.]

7.17 Freshness Value

The Freshness Value (FV) refers to a monotonic counter that is used to ensure freshness of the authenticated time synchronization messages. Such a monotonic counter could be realized by means of individual message counters, called Freshness Counter, or by a time stamp value called Freshness Timestamp. Freshness Values are to be derived from a Freshness Value Manager (FVM). The role of StbM here is to act as an interface between the <Bus>TSyn modules and the FVM.

More information about handling the FV can be found in [\[8, AUTOSAR Specification of Secure Onboard Communication\]](#)

The FVM can be a SWC with which StbM will communicate through an Rte interface, or it can be a CDD, in which case the Api's need to be explicitly added in the StbM configuration.

[SWS_StbM_00541]

Upstream requirements: [RS_TS_00039](#)

[The StbM shall support the Freshness Values for a Time Domain, if `StbM-QueryFreshnessValue` is set to a value different than NONE.

]

[SWS_StbM_00542]

Upstream requirements: [RS_TS_00039](#)

[If [StbMQueryFreshnessValue](#) is set to CFUNC, the functions:

- [StbM_GetTxFreshness](#)
- [StbM_GetRxFreshness](#)
- [StbM_SPduTxConfirmation](#)

shall call the corresponding configurable interfaces (refer to chapter 8.6.3):

- [GetTxFreshnessFct](#)
- [GetRxFreshnessFct](#)
- [SPduTxConfirmationFct](#)

and return the same return value as those.]

[SWS_StbM_00543]

Upstream requirements: [RS_TS_00039](#)

[If [StbMQueryFreshnessValue](#) is set to SERVICE, the functions:

- [StbM_GetTxFreshness](#)
- [StbM_GetRxFreshness](#)
- [StbM_SPduTxConfirmation](#)

shall call the corresponding service interfaces (refer to chapter 8.7.4):

- [GetTxFreshness](#)
- [GetRxFreshness](#)
- [SPduTxConfirmation](#)

and return the same return value as those.]

Rationale: StbM should not take any action on the return of the configurable/service interfaces, but return it, exactly as received, to the caller.

[SWS_StbM_00564]

Upstream requirements: [RS_TS_00039](#)

[If

- [StbMQueryFreshnessValue](#) is set to CFUNC
- and [StbMProvideTxTruncatedFreshnessValue](#) is set to false,

then StbM shall call `GetTxFreshnessFct` within `StbM_GetTxFreshnessTruncData`, and then do the truncation by itself.]

[SWS_StbM_00565]

Upstream requirements: [RS_TS_00039](#)

[If

- `StbMQueryFreshnessValue` is set to `SERVICE`
- and `StbMProvideTxTruncatedFreshnessValue` is set to `false`,

then StbM shall call `GetTxFreshness` within `StbM_GetTxFreshnessTruncData`, and then do the truncation by itself.]

[SWS_StbM_00566]

Upstream requirements: [RS_TS_00039](#)

[If

- `StbMQueryFreshnessValue` is set to `CFUNC`
- and `StbMProvideTxTruncatedFreshnessValue` is set to `true`,

then StbM shall call `GetTxFreshnessTruncDataFct` within `StbM_GetTxFreshnessTruncData` in order to obtain the truncated FV.]

[SWS_StbM_00567]

Upstream requirements: [RS_TS_00039](#)

[If

- `StbMQueryFreshnessValue` is set to `SERVICE`
- and `StbMProvideTxTruncatedFreshnessValue` is set to `true`,

then StbM shall call `GetTxFreshnessTruncData` within `StbM_GetTxFreshnessTruncData` in order to obtain the truncated FV.]

Note: Details on how the FV is aligned can be found in [SWS_SecOC_00220](#) in the [8, Specification of Secure Onboard Communication].

7.18 Interaction with User Defined Timesync Module (CDD)

User defined Time Base Providers are implemented by a CDD module. Details of the interaction between the StbM and such a CDD module are described in section "Interfacing with StbM module" of [9, "Complex Driver Design and Integration Guideline"].

7.19 Multicore Distribution

The StbM needs to ensure the precision of Synchronized Time Bases (i.e. the Global Time). Therefore, it needs to ensure processing APIs reporting current timestamps without any delay, even so APIs need to support Master/Satellite-approach according to [10, "Guide to BSW Distribution"]. This is only possible in a synchronous processing directly in the caller context. Means all these APIs are executed in different context and StbM needs to protect the access to according data with multi-core capable means.

[SWS_StbM_00513]

Upstream requirements: [SRS_BSW_00459](#)

[The StbM module shall apply appropriate mechanisms to allow calls of its APIs from other partitions than its main function, e.g. by providing a StbM satellite.]

Note: Parameter [StbMEcucPartitionRef](#) references the partition, which the [StbM_MainFunction](#) function is allocated to.

[SWS_StbM_00514]

Upstream requirements: [SRS_BSW_00459](#)

[The StbM module shall ensure to keep the synchronous contract of its APIs, even so they are called in other partitions than StbM module is assigned to.]

7.20 Error Handling

[SWS_StbM_00199]

Upstream requirements: [RS_TS_00025](#)

[For any StbM API service other than [StbM_Init](#) and [StbM_GetVersionInfo](#) all out parameters shall remain untouched, if an error occurs during execution of that API service.]

Note: For further details refer to the chapter 7.2 "Error Handling" in [4, SWS BSW General] and [Chapter 8](#) for API specific error handling.

7.21 Error Classification

Section "Error Handling" of the document [4, SWS BSW General] describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

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

7.21.1 Development Errors

[SWS_StbM_00041] Definition of development errors in module StbM

Upstream requirements: [SRS_BSW_00337](#), [SRS_BSW_00385](#), [SRS_BSW_00386](#), [SRS_BSW_00327](#), [SRS_BSW_00323](#)

[

<i>Type of error</i>	<i>Related error code</i>	<i>Error value</i>
API called with invalid time base ID	STBM_E_PARAM	0x0A
API called while StbM is not initialized	STBM_E_UNINIT	0x0B
API called with invalid pointer in parameter list	STBM_E_PARAM_POINTER	0x10
StbM_Init called with an invalid configuration pointer	STBM_E_INIT_FAILED	0x11
API not supported with current configuration	STBM_E_SERVICE_NOT_SUPPORTED	0x12
API called with invalid timestamp	STBM_E_PARAM_TIMESTAMP	0x25
API called with invalid user data	STBM_E_PARAM_USERDATA	0x26

]

7.21.2 Runtime Errors

There are no runtime errors.

7.21.3 Production Errors

There are no production errors.

7.21.4 Extended Production Errors

There are no extended production errors.

7.22 Version Check

For details refer to the chapter 5.1.8 "Version Check" in [4, SWS BSW General].

8 API specification

8.1 Imported types

In this chapter all types included from the following files are listed.

[SWS_StbM_00051] Definition of imported datatypes of module StbM

Upstream requirements: [SRS_BSW_00301](#)

[

Module	Header File	Imported Type
Can	Can_GeneralTypes.h	Can_TimeStampType (draft)
Eth	Eth.h	Eth_RateDeviationStatusType (draft)
	Eth.h	Eth_RateDeviationType (draft)
	Eth_GeneralTypes.h	Eth_TimeStampQualType (obsolete)
	Eth_GeneralTypes.h	Eth_TimeStampType (obsolete)
Gpt	Gpt.h	Gpt_ChannelType
	Gpt.h	Gpt_ValueType
Os	Os.h	ScheduleTableStatusRefType
	Os.h	ScheduleTableStatusType
	Os.h	ScheduleTableType
	Os.h	StatusType
	Os.h	TickRefType
	Os.h	TickType
	Rte_Os_Type.h	CounterType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

]

8.2 Type definitions

8.2.1 Extension to Std_ReturnType

[SWS_StbM_91017] Definition of Std_ReturnType-extension for module StbM

Upstream requirements: [RS_TS_00039](#)

[

Range	STBM_E_BUSY	0x02	The function call failed because the service is still busy
Description	–		
Available via	StbM.h		

]

8.2.2 StbM_ConfigType

[SWS_StbM_00249] Definition of datatype StbM_ConfigType

Upstream requirements: [SRS_BSW_00414](#)

[

Name	StbM_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	–
Description	Configuration data structure of the StbM module.	
Available via	StbM.h	

]

8.2.3 StbM_MeasurementType

[SWS_StbM_00384] Definition of datatype StbM_MeasurementType

Upstream requirements: [RS_TS_00034](#)

[

Name	StbM_MeasurementType	
Kind	Structure	
Elements	pathDelay	
	Type	uint32
	Comment	Propagation delay in nanoseconds
	rateDeviation	
	Type	Eth_RateDeviationType
	Comment	Rate deviation value as calculated by Timesync module
Description	Structure which contains additional measurement data	
Available via	StbM.h	

]

8.3 Function definitions

This is a list of functions provided for upper layer modules.

8.3.1 StbM_GetVersionInfo

[SWS_StbM_00066] Definition of API function StbM_GetVersionInfo

Upstream requirements: [SRS_BSW_00407](#)

[

Service Name	StbM_GetVersionInfo	
Syntax	<pre>void StbM_GetVersionInfo (Std_VersionInfoType* versioninfo)</pre>	
Service ID [hex]	0x05	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	versioninfo	Pointer to the memory location holding the version information of the module.
Return value	None	
Description	Returns the version information of this module.	
Available via	StbM.h	

]

[SWS_StbM_00094]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00337](#)

[If development error detection for the StbM module is enabled the function `StbM_GetVersionInfo` shall raise the development error `STBM_E_PARAM_POINTER` and return if versioninfo is a NULL pointer (NULL_PTR).]

8.3.2 StbM_Init

[SWS_StbM_00052] Definition of API function StbM_Init

Upstream requirements: [SRS_BSW_00101](#), [SRS_BSW_00358](#), [SRS_BSW_00414](#)

[

Service Name	StbM_Init	
Syntax	<pre>void StbM_Init (const StbM_ConfigType* ConfigPtr)</pre>	
Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to the selected configuration set.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Initializes the Synchronized Time-base Manager	
Available via	StbM.h	

]

The ECU State Manager calls the function [StbM_Init](#) during the startup phase of the ECU in order to initialize the module. The StbM is not functional until this function has been called.

[SWS_StbM_00100]

Upstream requirements: [SRS_BSW_00406](#)

[A static status variable denoting if the StbM is initialized shall be initialized with value 0 before any APIs of the StbM are called.]

[SWS_StbM_00121]

Upstream requirements: [SRS_BSW_00406](#)

[[StbM_Init](#) shall set the static status variable to a value not equal to 0.]

8.3.3 StbM_GetCurrentTime

[SWS_StbM_00195] Definition of API function StbM_GetCurrentTime

Upstream requirements: [RS_TS_00005](#), [RS_TS_00006](#), [RS_TS_00029](#), [RS_TS_00030](#), [RS_TS_00031](#), [RS_TS_00014](#)

[

Service Name	StbM_GetCurrentTime	
Syntax	<pre>Std_ReturnType StbM_GetCurrentTime (StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeTupleType* timeTuple, StbM_UserDataType* userData)</pre>	
Service ID [hex]	0x07	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	time base reference
Parameters (inout)	None	
Parameters (out)	timeTuple	Current time tuple that is valid at this time
	userData	User data of the Time Base
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Returns a time tuple (Local time, Global time and Timebase status) and user data details Note: This API shall be called with locked interrupts / within an Exclusive Area to prevent interruption (i.e., the risk that the time stamp is outdated on return of the function call).	
Available via	StbM.h	

]

[SWS_StbM_00196]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetCurrentTime` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which is

- not configured or
- within the reserved value range.

]

[SWS_StbM_00197]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetCurrentTime` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a `NULL` pointer for parameter `timeTuple` or `userData`.]

8.3.4 StbM_GetCurrentSafeTime

[SWS_StbM_91027] Definition of API function StbM_GetCurrentSafeTime

Status: DRAFT

Upstream requirements: [RS_TS_00005](#), [RS_TS_00006](#), [RS_TS_00029](#), [RS_TS_00030](#), [RS_TS_00031](#), [RS_TS_00014](#)

[

Service Name	StbM_GetCurrentSafeTime (draft)	
Syntax	<pre>Std_ReturnType StbM_GetCurrentSafeTime (StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeTripleType* timeTriple, StbM_UserDataType* userData)</pre>	
Service ID [hex]	0x30	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	time base reference
Parameters (inout)	None	
Parameters (out)	timeTriple	Current time triple values (Current time , Current fallback time, and global time)
	userData	User data of the Time Base
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	<p>Returns a time triple (Local Time, Fallback Local Time, Global time and timebase status) and user data details. Note: This API shall be called with locked interrupts / within an Exclusive Area to prevent interruption (i.e., the risk that the time stamp is outdated on return of the function call).</p> <p>Tags: atp.Status=draft</p>	
Available via	StbM.h	

]

[SWS_StbM_00598]

Status: DRAFT

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetCurrentSafeTime` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which is

- not configured or
- within the reserved value range.

]

[SWS_StbM_00605]

Status: DRAFT

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetCurrentSafeTime` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a `NULL` pointer for parameter `timeTriple` or `userData`]

8.3.5 StbM_GetCurrentVirtualLocalTime

[SWS_StbM_91006] Definition of API function StbM_GetCurrentVirtualLocalTime

Upstream requirements: [RS_TS_00006](#), [RS_TS_00008](#)

[

Service Name	StbM_GetCurrentVirtualLocalTime	
Syntax	Std_ReturnType StbM_GetCurrentVirtualLocalTime (StbM_SynchronizedTimeBaseType timeBaseId, StbM_VirtualLocalTimeType* localTimePtr)	
Service ID [hex]	0x1e	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
Parameters (inout)	None	
Parameters (out)	localTimePtr	Current Virtual Local Time value
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Returns the Virtual Local Time of the referenced Time Base.	
Available via	StbM.h	

]

[SWS_StbM_00444]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetCurrentVirtualLocalTime` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a `NULL` pointer for parameter `localTimePtr`.]

[SWS_StbM_00445]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetCurrentVirtualLocalTime` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which

- is not configured or

- is within the reserved value range.

]

8.3.6 StbM_GetFallbackVirtualLocalTime

[SWS_StbM_91029] Definition of API function StbM_GetFallbackVirtualLocalTime

Status: DRAFT

Upstream requirements: [RS_TS_00006](#), [RS_TS_00008](#)

[

Service Name	StbM_GetFallbackVirtualLocalTime (draft)	
Syntax	<pre>Std_ReturnType StbM_GetFallbackVirtualLocalTime (StbM_SynchronizedTimeBaseType timeBaseId, StbM_VirtualLocalTimeType* localTimePtr)</pre>	
Service ID [hex]	0x31	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
Parameters (inout)	None	
Parameters (out)	localTimePtr	Current Fallback Virtual Local Time value
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Returns the Fallback Virtual Local Time of the referenced Time Base. Tags: atp.Status=draft	
Available via	StbM.h	

]

[SWS_StbM_00625]

Status: DRAFT

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_GetFallbackVirtualLocalTime` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a NULL pointer for parameter `localTimePtr`]

[SWS_StbM_00626]

Status: DRAFT

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_GetFallbackVirtualLocalTime` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which

- is not configured or
- is within the reserved value range

]

8.3.7 StbM_SetGlobalTime

[SWS_StbM_00213] Definition of API function StbM_SetGlobalTime

Upstream requirements: [RS_TS_00029](#), [RS_TS_00010](#)

[

Service Name	StbM_SetGlobalTime	
Syntax	<pre>Std_ReturnType StbM_SetGlobalTime (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_TimeStampType* timeStamp, const StbM_UserDataType* userData)</pre>	
Service ID [hex]	0x0b	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	time base reference
	timeStamp	New time stamp
	userData	New user data (if not NULL)
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Allows the Customers to set the new global time that has to be valid for the system, which will be sent to the busses. This function will be used if a Time Master is present in this ECU.	
Available via	StbM.h	

]

[SWS_StbM_00214]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_SetGlobalTime` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which is

- not configured or
- within the reserved value range.

]

[SWS_StbM_00215]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_SetGlobalTime` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a `NULL` pointer for parameter `timeStamp`.]

[SWS_StbM_00448]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_SetGlobalTime` shall report to DET the development error `STBM_E_PARAM_TIMESTAMP`, if called with a parameter `timeStamp` that contains invalid elements (e.g., nanoseconds part > 999999999 ns).]

[SWS_StbM_00449]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_SetGlobalTime` shall report to DET the development error `STBM_E_PARAM_USERDATA`, if called with an invalid value of parameter `userData`, i.e., `userData->userDataLength > 3`.]

8.3.8 StbM_UpdateGlobalTime

[SWS_StbM_00385] Definition of API function StbM_UpdateGlobalTime

Upstream requirements: [RS_TS_00010](#)

[

Service Name	StbM_UpdateGlobalTime	
Syntax	<pre>Std_ReturnType StbM_UpdateGlobalTime (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_TimeStampType* timeStamp, const StbM_UserDataType* userData)</pre>	
Service ID [hex]	0x10	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	time base reference
	timeStamp	New time stamp
	userData	New user data (if not NULL)
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed





Description	Allows the Customers to set the Global Time that will be sent to the buses. This function will be used if a Time Master is present in this ECU. Using UpdateGlobalTime will not lead to an immediate transmission of the Global Time.
Available via	StbM.h

]

[SWS_StbM_00340]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_UpdateGlobalTime` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which is

- not configured or
- within the reserved value range.

]

[SWS_StbM_00341]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_UpdateGlobalTime` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a `NULL` pointer for parameter `timeStamp`.]

[SWS_StbM_00451]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_UpdateGlobalTime` shall report to DET the development error `STBM_E_PARAM_TIMESTAMP`, if called with a parameter `timeStamp` that contains invalid elements (e.g., nanoseconds part > 999999999 ns).]

[SWS_StbM_00452]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_UpdateGlobalTime` shall report to DET the development error `STBM_E_PARAM_USERDATA`, if called with an invalid value of parameter `userData`, i.e., `userData->userDataLength > 3`.]

8.3.9 StbM_SetUserData

[SWS_StbM_00218] Definition of API function StbM_SetUserData

Upstream requirements: [RS_TS_00015](#)

[

Service Name	StbM_SetUserData	
Syntax	<pre>Std_ReturnType StbM_SetUserData (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_UserDataType* userData)</pre>	
Service ID [hex]	0x0c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
	userData	New User Data
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Allows the Customers to set the new User Data that has to be valid for the system, which will be sent to the busses.	
Available via	StbM.h	

]

[SWS_StbM_00219]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_SetUserData` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which

- is not configured or
- is within the reserved value range.

]

[SWS_StbM_00630] "Service Not Supported" Development Error for StbM_SetUserData

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, when `StbM_SetUserData` is called for a Time Base for which the StbM is not configured as a Global Time Master, then `StbM_SetUserData` shall report to DET the development error `STBM_E_SERVICE_NOT_SUPPORTED`.]

[SWS_StbM_00220]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_SetUserData` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a `NULL` pointer for parameter `userData`.]

[SWS_StbM_00457]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_SetUserData` shall report to DET the development error `STBM_E_PARAM_USERDATA`, if called with an invalid value of parameter `userData`, i.e., `userData->userDataLength > 3`.]

8.3.10 StbM_BusSetGlobalTime

[SWS_StbM_00233] Definition of API function StbM_BusSetGlobalTime

Upstream requirements: [RS_TS_00007](#), [RS_TS_00030](#), [RS_TS_00031](#), [RS_TS_00034](#)

[

Service Name	StbM_BusSetGlobalTime	
Syntax	<pre>Std_ReturnType StbM_BusSetGlobalTime (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_TimeTupleType* timeTuplePtr, const StbM_UserDataType* userDataPtr, const StbM_MeasurementType* measureDataPtr)</pre>	
Service ID [hex]	0x0f	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
	timeTuplePtr	Rx Time Tuple
	userDataPtr	New User Data (if not NULL)
	measureDataPtr	New measurement data
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
	Description	
Available via	StbM.h	

]

[SWS_StbM_00234]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_BusSetGlobalTime` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which

- refers to a Pure Local Time Base or
- is not configured or
- is within the reserved value range

]

Note: A parameter `timeBaseId` within the reserved value range indicates legacy use.

[SWS_StbM_00235]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_BusSetGlobalTime` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a `NULL` pointer for parameter

- `timeTuplePtr`
- `measureDataPtr`

]

[SWS_StbM_00455]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_BusSetGlobalTime` shall report to DET the development error `STBM_E_PARAM_TIMESTAMP`, if called with a parameter `timeTuplePtr` that references invalid timestamps (e.g., nanoseconds part > 999999999 ns).]

[SWS_StbM_00456]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_BusSetGlobalTime` shall report to DET the development error `STBM_E_PARAM_USERDATA`, if called with a parameter `userDataPtr` that references an invalid length, i.e., `userDataPtr->userDataLength > 3`.]

8.3.11 StbM_GetRateDeviation

[SWS_StbM_00378] Definition of API function StbM_GetRateDeviation

Upstream requirements: [RS_TS_00018](#)

[

Service Name	StbM_GetRateDeviation	
Syntax	Std_ReturnType StbM_GetRateDeviation (StbM_SynchronizedTimeBaseType timeBaseId, StbM_RateDeviationType* rateDeviation)	
Service ID [hex]	0x11	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	timeBaseId	Time Base reference
Parameters (inout)	None	
Parameters (out)	rateDeviation	Value of the current rate deviation of a Time Base
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Returns value of the current rate deviation of a Time Base	
Available via	StbM.h	

]

[SWS_StbM_00379]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetRateDeviation` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which

- is not configured or
- is within the reserved value range.

]

[SWS_StbM_00380]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetRateDeviation` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a `NULL` pointer for parameter `rateDeviation`.]

8.3.12 StbM_SetRateCorrection

[SWS_StbM_00390] Definition of API function StbM_SetRateCorrection

Upstream requirements: [RS_TS_00018](#)

[

Service Name	StbM_SetRateCorrection	
Syntax	Std_ReturnType StbM_SetRateCorrection (StbM_SynchronizedTimeBaseType timeBaseId, StbM_RateDeviationType rateDeviation)	
Service ID [hex]	0x12	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
	rateDeviation	Value of the applied rate deviation
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
	Description	
Description		Allows to set the rate of a Synchronized Time Base (being either a Pure Local Time Base or not).
Available via	StbM.h	

]

[SWS_StbM_00391]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch [StbMDevErrorDetect](#) is set to TRUE, [StbM_SetRateCorrection](#) shall report to DET the development error [STBM_E_PARAM](#), if called with a parameter [timeBaseId](#), which

- is not configured or
- is within the reserved value range.

]

[SWS_StbM_00392]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch [StbMDevErrorDetect](#) is set to TRUE, [StbM_SetRateCorrection](#) shall report to DET the development error [STBM_E_SERVICE_NOT_SUPPORTED](#), if [StbMAllowMasterRateCorrection](#) is set to FALSE for the corresponding Time Base, i.e., it is not allowed to call [StbM_SetRateCorrection](#).]

8.3.13 StbM_GetTimeLeap

[SWS_StbM_00267] Definition of API function StbM_GetTimeLeap

Upstream requirements: [RS_TS_00005](#)

[

Service Name	StbM_GetTimeLeap	
Syntax	<pre>Std_ReturnType StbM_GetTimeLeap (StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeDiffType* timeJump)</pre>	
Service ID [hex]	0x13	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	timeBaseId	Time Base reference
Parameters (inout)	None	
Parameters (out)	timeJump	Time leap value
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Returns value of Time Leap.	
Available via	StbM.h	

]

[SWS_StbM_00268]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetTimeLeap` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which

- is not configured or
- refers to a Pure Local Time Base or
- is within the reserved value range.

]

[SWS_StbM_00269]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetTimeLeap` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a `NULL` pointer for parameter `timeJump`.]

8.3.14 StbM_GetTimeBaseStatus

[SWS_StbM_00263] Definition of API function StbM_GetTimeBaseStatus

Upstream requirements: [RS_TS_00005](#)

[

Service Name	StbM_GetTimeBaseStatus	
Syntax	Std_ReturnType StbM_GetTimeBaseStatus (StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeBaseStatusType* syncTimeBaseStatus)	
Service ID [hex]	0x14	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	timeBaseId	Time Base reference
Parameters (inout)	None	
Parameters (out)	syncTimeBaseStatus	Status of the Synchronized (or Pure Local) Time Base
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Returns detailed status information for a Synchronized (or Pure Local) Time Base.	
Available via	StbM.h	

]

[SWS_StbM_00264]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch [StbMDevErrorDetect](#) is set to TRUE, [StbM_GetTimeBaseStatus](#) shall report to DET the development error [STBM_E_PARAM](#), if called with a parameter [timeBaseId](#), which

- is not configured or
- is within the reserved value range.

]

[SWS_StbM_00386]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch [StbMDevErrorDetect](#) is set to TRUE, [StbM_GetTimeBaseStatus](#) shall report to DET the development error [STBM_E_PARAM_POINTER](#), if called with a NULL pointer for parameter [syncTimeBaseStatus](#).]

8.3.15 StbM_CloneTimeBase

[SWS_StbM_91012] Definition of API function StbM_CloneTimeBase

Upstream requirements: [RS_TS_00038](#)

[

Service Name	StbM_CloneTimeBase	
Syntax	<pre>Std_ReturnType StbM_CloneTimeBase (StbM_SynchronizedTimeBaseType timeBaseId, StbM_CloneConfigType cloneCfg, StbM_TimeBaseStatusType statusMask, StbM_TimeBaseStatusType statusValue)</pre>	
Service ID [hex]	0x2b	
Sync/Async	Depends on configuration	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Destination Time Base for cloning
	cloneCfg	Refines how source Time Base is cloned to destination
	statusMask	Status flags mask for definition of relevant status flags
	statusValue	Status flags value define whether cloning shall take place
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	<p>Copies Time Base data (current time, user data, rate correction) from Source Time Base to Destination Time Base. The Source Time Base is identified by the parameter StbMSourceTime Base (ECUC_StbM_00074).</p> <p>StbM_?CloneTimeBase behaves synchronously (immediate copy of Time Base) if DEFERRED_COPY flag of parameter cloneCfg is set to true, otherwise it behaves asynchronously (deferred copy of Time Base).</p> <p>Note: Even, if configured to behave synchronously (immediate copy of Time Base), actual transmission of cloned Time Base value on the bus occurs asynchronously.</p>	
Available via	StbM.h	

]

[SWS_StbM_00537]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch [StbMDevErrorDetect](#) is set to TRUE, [StbM_CloneTimeBase](#) shall report to DET the development error [STBM_E_PARAM](#), if called with a parameter [time-BaseId](#), which

- is not configured or
- is within the reserved value range.

]

8.3.16 StbM_SetCloningOffset

[SWS_StbM_91030] Definition of API function StbM_SetCloningOffset

Upstream requirements: [RS_TS_00038](#)

[

Service Name	StbM_SetCloningOffset	
Syntax	Std_ReturnType StbM_SetCloningOffset (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_TimeStampType* timeOffset)	
Service ID [hex]	0x32	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference of the Destination Time Base
	timeOffset	Offset value for cloning
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful. E_NOT_OK: failed
Description	Sets an offset time that will be added to the synchronized time when cloning a Time Base.	
Available via	StbM.h	

]

[SWS_StbM_00627] "Invalid Time Base Parameter" Development Error for StbM_SetCloningOffset

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch [StbMDevErrorDetect](#) is set to TRUE, [StbM_SetCloningOffset](#) shall report to DET the development error [STBM_E_PARAM](#), if called with a parameter [timeBaseId](#), which

- is not configured or
- is within the reserved value range.

]

[SWS_StbM_00628] "Service Not Supported" Development Error for StbM_SetCloningOffset

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch [StbMDevErrorDetect](#) is set to TRUE, [StbM_SetCloningOffset](#) shall report to DET the development error [STBM_E_SERVICE_NOT_SUPPORTED](#), if called for a Time Base, which has no Source Time Base (refer to parameter [StbM-SourceTimeBase](#)) configured.]

8.3.17 StbM_StartTimer

[SWS_StbM_00272] Definition of API function StbM_StartTimer

Upstream requirements: [RS_TS_00017](#)

[

Service Name	StbM_StartTimer	
Syntax	<pre>Std_ReturnType StbM_StartTimer (StbM_SynchronizedTimeBaseType timeBaseId, StbM_CustomerIdType customerId, const StbM_TimeStampType* expireTime)</pre>	
Service ID [hex]	0x15	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
	customerId	unique identifier of a notification customer
	expireTime	Time value relative to current Time Base value of the Notification Customer, when the Timer shall expire
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Sets a time value, which the Time Base value is compared against	
Available via	StbM.h	

]

[SWS_StbM_00296]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_StartTimer` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which

- is not configured or
- is within the reserved value range.

]

[SWS_StbM_00406]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_StartTimer` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `customerId`, which is not configured.]

[SWS_StbM_00298]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_StartTimer` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with an invalid pointer of parameter `expireTime`.]

[SWS_StbM_00556]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_StartTimer` shall report to DET the development error `STBM_E_PARAM_TIMESTAMP`, if called with a parameter `expireTime` that contains invalid elements (e.g., nanoseconds part > 999999999 ns).]

8.3.18 StbM_GetSyncTimeRecordHead

[SWS_StbM_00319] Definition of API function `StbM_GetSyncTimeRecordHead`

Upstream requirements: [RS_TS_00034](#)

[

Service Name	StbM_GetSyncTimeRecordHead	
Syntax	<pre>Std_ReturnType StbM_GetSyncTimeRecordHead (StbM_SynchronizedTimeBaseType timeBaseId, StbM_SyncRecordTableHeadType* syncRecordTableHead)</pre>	
Service ID [hex]	0x16	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	timeBaseId	Time Base reference
Parameters (inout)	None	
Parameters (out)	syncRecordTableHead	Header of the table
Return value	Std_ReturnType	E_OK: Table access done E_NOT_OK: Table contains no data or access invalid
Description	Accesses to the recorded snapshot data Header of the table belonging to the Synchronized Time Base.	
Available via	StbM.h	

]

[SWS_StbM_00320]

Upstream requirements: [RS_TS_00034](#)

[The function `StbM_GetSyncTimeRecordHead` shall be pre compile time configurable ON/OFF by the configuration parameter `StbMTimeRecordingSupport`.]

[SWS_StbM_00394]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetSyncTimeRecord-Head` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which

- is not configured or
- refers to a Pure Local Time Base or
- is within the reserved value range.

]

[SWS_StbM_00405]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetSyncTimeRecord-Head` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with an invalid pointer of parameter `syncRecordTableHead`.]

8.3.19 StbM_TriggerTimeTransmission

[SWS_StbM_00346] Definition of API function StbM_TriggerTimeTransmission

Upstream requirements: [RS_TS_00011](#)

[

Service Name	StbM_TriggerTimeTransmission	
Syntax	Std_ReturnType StbM_TriggerTimeTransmission (StbM_SynchronizedTimeBaseType timeBaseId)	
Service ID [hex]	0x1c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Operation successful E_NOT_OK: Operation not successful
Description	Called by the <Upper Layer> to force the Timesync Modules to transmit the current Time Base again due to an incremented timeBaseUpdateCounter[timeBaseId]	
Available via	StbM.h	

]

[SWS_StbM_00349]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_TriggerTimeTransmission` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which

- is not configured or
- refers to a Pure Local Time Base or
- is within the reserved value range.

]

8.3.20 StbM_GetTimeBaseUpdateCounter

[SWS_StbM_00347] Definition of API function StbM_GetTimeBaseUpdateCounter

Upstream requirements: [RS_TS_00011](#)

[

Service Name	StbM_GetTimeBaseUpdateCounter	
Syntax	<pre>uint8 StbM_GetTimeBaseUpdateCounter (StbM_SynchronizedTimeBaseType timeBaseId)</pre>	
Service ID [hex]	0x1b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	timeBaseId	Time Base reference
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint8	Counter value belonging to the Time Base, that indicates a Time Base update to the Timesync Modules
Description	Allows the Timesync Modules to detect, whether a Time Base should be transmitted immediately in the subsequent <Bus>TSyn_MainFunction() cycle.	
Available via	StbM.h	

]

[SWS_StbM_00348]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetTimeBaseUpdateCounter` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which

- is not configured or

- refers to a Pure Local Time Base or
- is within the reserved value range.

]

8.3.21 StbM_GetMasterConfig

[SWS_StbM_91002] Definition of API function StbM_GetMasterConfig

Upstream requirements: [RS_TS_00029](#)

[

Service Name	StbM_GetMasterConfig	
Syntax	<pre>Std_ReturnType StbM_GetMasterConfig (StbM_SynchronizedTimeBaseType timeBaseId, StbM_MasterConfigType* masterConfig)</pre>	
Service ID [hex]	0x1d	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	timeBaseId	Time Base reference
Parameters (inout)	None	
Parameters (out)	masterConfig	Indicates, if system wide master functionality is supported
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Indicates if the functionality for a system wide master (e.g. StbM_SetGlobalTime) for a given Time Base is available or not.	
Available via	StbM.h	

]

[SWS_StbM_00415]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_GetMasterConfig` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which

- is not configured or
- is within the reserved value range.

]

[SWS_StbM_00416]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetMasterConfig` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a `NULL` pointer for parameter `masterConfig`.]

8.3.22 StbM_CanSetSlaveTimingData

[SWS_StbM_00484] Definition of API function StbM_CanSetSlaveTimingData

Status: DRAFT

Upstream requirements: [RS_TS_00030](#), [RS_TS_00031](#), [RS_TS_00034](#)

[

Service Name	StbM_CanSetSlaveTimingData (draft)	
Syntax	<pre>Std_ReturnType StbM_CanSetSlaveTimingData (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_CanTimeSlaveMeasurementType* measureDataPtr)</pre>	
Service ID [hex]	0x26	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
	measureDataPtr	New measurement data
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
	Description Allows the CanTSyn Module to forward CAN specific details to the StbM. Tags: atp.Status=draft	
Available via	StbM_CanTSyn.h	

]

8.3.23 StbM_FrSetSlaveTimingData

[SWS_StbM_00485] Definition of API function StbM_FrSetSlaveTimingData

Status: DRAFT

Upstream requirements: [RS_TS_00030](#), [RS_TS_00031](#), [RS_TS_00034](#)

[

Service Name	StbM_FrSetSlaveTimingData (draft)	
Syntax	Std_ReturnType StbM_FrSetSlaveTimingData (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_FrTimeSlaveMeasurementType* measureDataPtr)	
Service ID [hex]	0x27	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
	measureDataPtr	New measurement data
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Allows the FrTSyn Module to forward Flexray specific details to the StbM. Tags: atp.Status=draft	
Available via	StbM_FrTSyn.h	

]

8.3.24 StbM_EthSetSlaveTimingData

[SWS_StbM_00486] Definition of API function StbM_EthSetSlaveTimingData

Status: DRAFT

Upstream requirements: [RS_TS_00030](#), [RS_TS_00031](#), [RS_TS_00034](#)

[

Service Name	StbM_EthSetSlaveTimingData (draft)	
Syntax	Std_ReturnType StbM_EthSetSlaveTimingData (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_EthTimeSlaveMeasurementType* measureDataPtr)	
Service ID [hex]	0x28	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
	measureDataPtr	New measurement data

▽



Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Allows the EthTSyn Module to forward Ethernet specific details to the StbM. Tags: atp.Status=draft	
Available via	StbM_EthTSyn.h	

]

[SWS_StbM_00487]

Upstream requirements: [RS_TS_00034](#)

[The function `StbM_<bus>SetSlaveTimingData` shall be pre compile time configurable ON/OFF. If the corresponding `<bus>TSyn` module is configured with Time Validation Support enabled (refer to parameter `<bus>TSynTimeValidationSupport` in `<bus>TSyn` module) `StbM_<bus>SetSlaveTimingData` shall be ON, otherwise OFF.]

[SWS_StbM_00488]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_<bus>SetSlaveTimingData` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which does not refer to a Synchronized Time Base.]

Note: A parameter `timeBaseId` within the reserved value range indicates legacy use.

[SWS_StbM_00489]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_<bus>SetSlaveTimingData` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a NULL pointer for parameter `measureDataPtr`.]

8.3.25 StbM_CanSetMasterTimingData

[SWS_StbM_00490] Definition of API function StbM_CanSetMasterTimingData

Status: DRAFT

Upstream requirements: [RS_TS_00029](#), [RS_TS_00031](#), [RS_TS_00034](#)

[

Service Name	StbM_CanSetMasterTimingData (draft)	
Syntax	Std_ReturnType StbM_CanSetMasterTimingData (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_CanTimeMasterMeasurementType* measureDataPtr)	
Service ID [hex]	0x20	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
	measureDataPtr	Measurement data
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Provides CAN Timesyn module specific data for a Time Master to the StbM. Tags: atp.Status=draft	
Available via	StbM_CanTSyn.h	

]

8.3.26 StbM_FrSetMasterTimingData

[SWS_StbM_00491] Definition of API function StbM_FrSetMasterTimingData

Status: DRAFT

Upstream requirements: [RS_TS_00029](#), [RS_TS_00031](#), [RS_TS_00034](#)

[

Service Name	StbM_FrSetMasterTimingData (draft)	
Syntax	Std_ReturnType StbM_FrSetMasterTimingData (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_FrTimeMasterMeasurementType* measureDataPtr)	
Service ID [hex]	0x21	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
	measureDataPtr	Measurement data

▽



Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Provides Flexray Timesyn module specific data for a Time Master to the StbM. Tags: atp.Status=draft	
Available via	StbM_FrTSyn.h	

]

8.3.27 StbM_EthSetMasterTimingData

[SWS_StbM_00492] Definition of API function StbM_EthSetMasterTimingData

Status: DRAFT

Upstream requirements: [RS_TS_00029](#), [RS_TS_00031](#), [RS_TS_00034](#)

[

Service Name	StbM_EthSetMasterTimingData (draft)	
Syntax	Std_ReturnType StbM_EthSetMasterTimingData (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_EthTimeMasterMeasurementType* measureDataPtr)	
Service ID [hex]	0x22	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseld	Time Base reference
	measureDataPtr	Measurement data
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Provides Ethernet Timesyn module specific data for a Time Master to the StbM. Tags: atp.Status=draft	
Available via	StbM_EthTSyn.h	

]

[SWS_StbM_00493]

Upstream requirements: [RS_TS_00034](#)

[The function StbM_<bus>SetMasterTimingData shall be pre compile time configurable ON/OFF. If the corresponding <bus>TSyn module is configured with Time Validation Support enabled (refer to parameter <bus>TSynTimeValidationSupport in <bus>TSyn module), StbM_<bus>SetMasterTimingData shall be ON, otherwise .]

[SWS_StbM_00494]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_<bus>SetMasterTimingData` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which does not refer to a Synchronized Time Base]

[SWS_StbM_00495]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_<bus>SetMasterTimingData` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a `NULL` pointer for parameter `measureDataPtr`.]

8.3.28 StbM_EthSetPdelayInitiatorData

[SWS_StbM_00496] Definition of API function StbM_EthSetPdelayInitiatorData

Status: DRAFT

Upstream requirements: [RS_TS_00034](#)

[

Service Name	StbM_EthSetPdelayInitiatorData (draft)	
Syntax	<pre>Std_ReturnType StbM_EthSetPdelayInitiatorData (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_PdelayInitiatorMeasurementType* measureDataPtr)</pre>	
Service ID [hex]	0x23	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
	measureDataPtr	Measurement data
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	<p>–</p> <p>Tags: atp.Status=draft</p>	
Available via	StbM_EthTSyn.h	

]

[SWS_StbM_00497]

Upstream requirements: [RS_TS_00034](#)

[The function [StbM_EthSetPdelayInitiatorData](#) shall be pre compile time configurable ON/OFF. If the EthTSyn module is configured with Time Validation Support enabled (refer to parameter [EthTSynTimeValidationSupport](#) in EthTSyn module), [StbM_EthSetPdelayInitiatorData](#) shall be ON, otherwise OFF.]

[SWS_StbM_00498]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch [StbMDevErrorDetect](#) is set to TRUE, [StbM_EthSetPdelayInitiatorData](#) shall report to DET the development error [STBM_E_PARAM](#), if called with a parameter [timeBaseId](#), which does not refer to a Synchronized Time Base]

[SWS_StbM_00499]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch [StbMDevErrorDetect](#) is set to TRUE, [StbM_EthSetPdelayInitiatorData](#) shall report to DET the development error [STBM_E_PARAM_POINTER](#), if called with a NULL pointer for parameter [measureDataPtr](#).]

8.3.29 StbM_EthSetPdelayResponderData

[SWS_StbM_00500] Definition of API function StbM_EthSetPdelayResponderData

Status: DRAFT

Upstream requirements: [RS_TS_00034](#)

[

Service Name	StbM_EthSetPdelayResponderData (draft)	
Syntax	<pre>Std_ReturnType StbM_EthSetPdelayResponderData (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_PdelayResponderMeasurementType* measureDataPtr)</pre>	
Service ID [hex]	0x24	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
	measureDataPtr	Measurement data
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed





Description	- Tags: atp.Status=draft
Available via	StbM_EthTSyn.h

]

[SWS_StbM_00501]

Upstream requirements: [RS_TS_00034](#)

[The function `StbM_EthSetPdelayResponderData` shall be pre compile time configurable ON/OFF. If the EthTSyn module is configured with Time Validation Support enabled (refer to parameter `EthTSynTimeValidationSupport` in EthTSyn module), `StbM_EthSetPdelayResponderData` shall be ON, otherwise OFF.]

[SWS_StbM_00502]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_EthSetPdelayResponderData` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which does not refer to a Synchronized Time Base]

[SWS_StbM_00503]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#), [RS_TS_00034](#)

[If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_EthSetPdelayResponderData` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a NULL pointer for parameter `measureDataPtr`]

8.3.30 StbM_GetBusProtocolParam

[SWS_StbM_91007] Definition of API function StbM_GetBusProtocolParam

Upstream requirements: [RS_TS_20069](#)

[

Service Name	StbM_GetBusProtocolParam
Syntax	Std_ReturnType StbM_GetBusProtocolParam (StbM_SynchronizedTimeBaseType timeBaseId, StbM_ProtocolParamType* protocolParam)
Service ID [hex]	0x29
Sync/Async	Synchronous
Reentrancy	Reentrant





Parameters (in)	timeBaseId	Id of referenced Time Base
Parameters (inout)	None	
Parameters (out)	protocolParam	structure to store received Follow_Up information TLV parameters
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	This API is used to get bus specific parameters from received Follow_Up message	
Available via	StbM.h	

]

[SWS_StbM_00518]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_GetBusProtocolParam` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which is not referring to a Synchronized Time Base.]

[SWS_StbM_00519]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_GetBusProtocolParam` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a NULL pointer for parameter `protocolParam`.]

8.3.31 StbM_SetBusProtocolParam

[SWS_StbM_91008] Definition of API function `StbM_SetBusProtocolParam`

Upstream requirements: [RS_TS_20069](#)

[

Service Name	StbM_SetBusProtocolParam	
Syntax	<pre>Std_ReturnType StbM_SetBusProtocolParam (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_ProtocolParamType* protocolParam)</pre>	
Service ID [hex]	0x2a	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Id of referenced Time Base
	protocolParam	structure with Follow_Up information TLV parameters
Parameters (inout)	None	
Parameters (out)	None	





Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	This API is used to set bus specific parameters of a Time Master	
Available via	StbM.h	

]

[SWS_StbM_00520]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch [StbMDevErrorDetect](#) is set to TRUE, [StbM_SetBusProtocolParam](#) shall report to DET the development error [STBM_E_PARAM](#), if called with a parameter [timeBaseId](#), which is not referring to a Synchronized Time Base.]

[SWS_StbM_00521]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch [StbMDevErrorDetect](#) is set to TRUE, [StbM_SetBusProtocolParam](#) shall report to DET the development error [STBM_E_PARAM_POINTER](#), if called with a NULL pointer for parameter [protocolParam](#).]

8.3.32 StbM_GetTxFreshness

[SWS_StbM_91018] Definition of API function StbM_GetTxFreshness

Upstream requirements: [RS_TS_00039](#)

[

Service Name	StbM_GetTxFreshness	
Syntax	Std_ReturnType StbM_GetTxFreshness (uint16 StbMFreshnessValueId, uint8* StbMFreshnessValue, uint32* StbMFreshnessValueLength)	
Service ID [hex]	0x2c	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	StbMFreshnessValueId	Holds the identifier of the freshness value
Parameters (inout)	StbMFreshnessValueLength	Holds the length of the provided freshness in bits
Parameters (out)	StbMFreshnessValue	Holds the current freshness value





Return value	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed, a Freshness Value cannot be provided due to general issues for Freshness for this FreshnessValueId STBM_E_BUSY: The Freshness information can temporarily not be provided.
Description	This API returns the freshness value from the Most Significant Bits in the first byte, of the Freshness array, in big endian format.	
Available via	StbM.h	

]

[SWS_StbM_00544]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch [StbMDevErrorDetect](#) is set to TRUE, [StbM_GetTxFreshness](#) shall report to DET the development error [STBM_E_PARAM](#), if called with a parameter [StbMFreshnessValueId](#), which is not configured.]

[SWS_StbM_00545]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch [StbMDevErrorDetect](#) is set to TRUE, [StbM_GetTxFreshness](#) shall report to DET the development error [STBM_E_PARAM_POINTER](#), if called with a NULL pointer for parameter

- [StbMFreshnessValueLength](#)
- [StbMFreshnessValue](#)

]

8.3.33 StbM_GetTxFreshnessTruncData

[SWS_StbM_91014] Definition of API function StbM_GetTxFreshnessTruncData

Upstream requirements: [RS_TS_00039](#)

[

Service Name	StbM_GetTxFreshnessTruncData
Syntax	Std_ReturnType StbM_GetTxFreshnessTruncData (uint16 StbMFreshnessValueId, uint8* StbMFreshnessValue, uint32* StbMFreshnessValueLength, uint8* StbMTruncatedFreshnessValue, uint32* StbMTruncatedFreshnessValueLength)
Service ID [hex]	0x2d





Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	StbMFreshnessValueId	Holds the identifier of the freshness value
Parameters (inout)	StbMFreshnessValueLength	Holds the length of the provided freshness in bits
	StbMTruncatedFreshnessValueLength	Provides the truncated freshness length configured for this freshness. The caller may adapt the value if needed or can leave it unchanged if the configured length and provided length is the same.
Parameters (out)	StbMFreshnessValue	Holds the current freshness value
	StbMTruncatedFreshnessValue	Holds the truncated freshness to be included into the Secured time sync message. The parameter is optional.
Return value	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed, a Freshness Value cannot be provided due to general issues for Freshness for this FreshnessValueId STBM_E_BUSY: The Freshness information can temporarily not be provided.
Description	This interface is used by the StbM to obtain the current freshness value. The interface function provides also the truncated freshness transmitted in the secured time sync message.	
Available via	StbM.h	

]

[SWS_StbM_00546]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_GetTxFreshnessTruncData` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `StbMFreshnessValueId`, which is not configured.]

[SWS_StbM_00547]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_GetTxFreshnessTruncData` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a NULL pointer for parameter

- `StbMFreshnessValueLength`
- `StbMFreshnessValue`
- `StbMTruncatedFreshnessValueLength`
- `StbMTruncatedFreshnessValue`

]

8.3.34 StbM_SPduTxConfirmation

[SWS_StbM_91015] Definition of API function StbM_SPduTxConfirmation

Upstream requirements: [RS_TS_00039](#)

[

Service Name	StbM_SPduTxConfirmation	
Syntax	<pre>void StbM_SPduTxConfirmation (uint16 StbMFreshnessValueId)</pre>	
Service ID [hex]	0x2e	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	StbMFreshnessValueId	Holds the identifier of the freshness value
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This interface is used by the StbM to indicate that the Secured Time Synchronization Message has been initiated for transmission.	
Available via	StbM.h	

]

[SWS_StbM_00548]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch [StbMDevErrorDetect](#) is set to `TRUE`, [StbM_SPduTxConfirmation](#) shall report to DET the development error [STBM_E_PARAM](#), if called with a parameter [StbMFreshnessValueId](#), which is not configured.]

8.3.35 StbM_GetRxFreshness

[SWS_StbM_91016] Definition of API function StbM_GetRxFreshness

Upstream requirements: [RS_TS_00039](#)

[

Service Name	StbM_GetRxFreshness	
Syntax	<pre>Std_ReturnType StbM_GetRxFreshness (uint16 StbMFreshnessValueId, const uint8* StbMTruncatedFreshnessValue, uint32 StbMTruncatedFreshnessValueLength, uint16 StbMAuthVerifyAttempts, uint8* StbMFreshnessValue, uint32* StbMFreshnessValueLength)</pre>	
Service ID [hex]	0x2f	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	StbMFreshnessValueId	Holds the identifier of the freshness value.
	StbMTruncatedFreshnessValue	Holds the truncated freshness value that was contained in the Secured time sync message.
	StbMTruncatedFreshnessValueLength	Holds the length in bits of the truncated freshness value.
	StbMAuthVerifyAttempts	Holds the number of authentication verify attempts of this time sync message since the last reception. The value is 0 for the first attempt and incremented on every unsuccessful verification attempt.
Parameters (inout)	StbMFreshnessValueLength	Holds the length in bits of the freshness value
Parameters (out)	StbMFreshnessValue	Holds the current freshness value
Return value	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed, a Freshness Value cannot be provided due to general issues for Freshness for this FreshnessValueId STBM_E_BUSY: The Freshness information can temporarily not be provided.
Description	This interface is used by the StbM to query the current freshness value.	
Available via	StbM.h	

]

[SWS_StbM_00549]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetRxFreshness` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `StbMFreshnessValueId`, which is not configured.]

[SWS_StbM_00550]

Upstream requirements: [SRS_BSW_00386](#), [SRS_BSW_00323](#)

[If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetRxFreshness` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a `NULL` pointer for parameter

- `StbMFreshnessValueLength`
- `StbMFreshnessValue`

]

8.4 Callback notifications

No callback notifications defined.

8.5 Scheduled functions

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

8.5.1 StbM_MainFunction

[SWS_StbM_00057] Definition of scheduled function StbM_MainFunction

Upstream requirements: [SRS_BSW_00172](#), [SRS_BSW_00373](#)

[

Service Name	StbM_MainFunction
Syntax	<code>void StbM_MainFunction (void)</code>
Service ID [hex]	0x04
Description	This function will be called cyclically by a task body provided by the BSW Schedule. It will invoke the triggered customers and synchronize the referenced OS ScheduleTables.
Available via	SchM_StbM.h

]

[SWS_StbM_00407]

Upstream requirements: [SRS_BSW_00172](#)

[The frequency of invocations of `StbM_MainFunction` is determined by the configuration parameter `StbMMainFunctionPeriod`.]

[SWS_StbM_00107]

Upstream requirements: [RS_TS_00032](#), [SRS_BSW_00333](#)

[If OS is configured as triggered customer, the function `StbM_MainFunction` shall synchronize the referenced OS ScheduleTable.]

8.6 Expected interfaces

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

8.6.1 Mandatory interfaces

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

[SWS_StbM_00058] Definition of mandatory interfaces required by module StbM

Upstream requirements: [SRS_BSW_00301](#), [SRS_BSW_00339](#)

[

<i>API Function</i>	<i>Header File</i>	<i>Description</i>
Det_ReportError	Det.h	Service to report development errors.
EthTSyn_GetProtocolParam	EthTSyn.h	This API is used to read FollowUp information TLV parameters from received Follow_Up message.
EthTSyn_SetProtocolParam	EthTSyn.h	This API is used to set FollowUp information TLV parameters of a Follow_Up message prior transmission. The API is called within StbM_SetBus ProtocolParam which provides the content of the structure protocolParam.

]

8.6.2 Optional interfaces

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

[SWS_StbM_00059] Definition of optional interfaces requested by module StbM

Upstream requirements: [SRS_BSW_00301](#), [SRS_BSW_00339](#)

[

API Function	Header File	Description
Canlf_GetCurrentTime (draft)	Canlf.h	This service calls the corresponding CAN Driver service to retrieve the current time value out of the HW registers. Tags: atp.Status=draft
Ethlf_GetCurrentTime (obsolete)	Ethlf.h	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. Important Note: Ethlf_GetCurrentTime may be called within an exclusive area. Tags: atp.Status=obsolete
GetCounterValue	Os.h	This service reads the current count value of a counter (returning either the hardware timer ticks if counter is driven by hardware or the software ticks when user drives counter).
GetElapsedValue	Os.h	This service gets the number of ticks between the current tick value and a previously read tick value.
GetScheduleTableStatus	Os.h	This service queries the state of a schedule table (also with respect to synchronization).
Gpt_GetTimeElapsed	Gpt.h	Returns the time already elapsed.
Gpt_StartTimer	Gpt.h	Starts a timer channel.
SyncScheduleTable	Os.h	This service provides the schedule table with a synchronization count and start synchronization.

]

8.6.3 Configurable interfaces

In this section, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of this kind of interfaces are not fixed because they are configurable.

Note: The return value of the callback C-APIs is defined as `Std_ReturnType` to follow the signature of the corresponding service APIs. According to chapter 8.4 of [4, SWS BSW General] the caller, i.e. the StbM, can assume, that the callback will always return `E_OK`.

8.6.3.1 SyncTimeRecordBlockCallback

[SWS_StbM_00322] Definition of configurable interface SyncTimeRecordBlockCallback<TimeBase>

Upstream requirements: [RS_TS_00034](#)

[

Service Name	SyncTimeRecordBlockCallback<TimeBase>	
Syntax	Std_ReturnType SyncTimeRecordBlockCallback<TimeBase> (const StbM_SyncRecordTableBlockType* syncRecordTableBlock)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	syncRecordTableBlock	Block of the table
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Table access done E_NOT_OK: Table contains no data or access invalid
Description	Provides a recorded snapshot data block of the measurement data table belonging to the Synchronized Time Base.	
Available via	StbM_Externals.h	

]

[SWS_StbM_00323]

Upstream requirements: [RS_TS_00034](#)

[The function [SyncTimeRecordBlockCallback<TimeBase>](#) shall be set by the parameter [StbMSyncTimeRecordBlockCallback](#).]

8.6.3.2 StatusNotificationCallback

[SWS_StbM_00285] Definition of configurable interface StatusNotificationCallback<TimeBase>

Upstream requirements: [RS_TS_00037](#), [RS_TS_00016](#), [SRS_BSW_00457](#), [SRS_BSW_00360](#),
[SRS_BSW_00333](#)

[

Service Name	StatusNotificationCallback<TimeBase>	
Syntax	Std_ReturnType StatusNotificationCallback<TimeBase> (StbM_TimeBaseNotificationType eventNotification)	
Sync/Async	Synchronous	





Reentrancy	Non Reentrant	
Parameters (in)	eventNotification	Holds the notification bits for the different Time Base related events
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	The callback notifies the customers, when a <TimeBase> related event occurs, which is enabled by the notification mask	
Available via	StbM_Externals.h	

]

[SWS_StbM_00299]

Upstream requirements: [RS_TS_00016](#)

[The status notification callback function shall be set by the parameter [StbMTimeNotificationCallback](#).]

Note: The event notification callback might be called in interrupt context only, if there is no callback configured in StbM which belongs to a SW-C.

8.6.3.3 <Customer>_TimeNotificationCallback

[SWS_StbM_00273] Definition of configurable interface <Customer>_TimeNotificationCallback<TimeBase>

Upstream requirements: [RS_TS_00017](#), [SRS_BSW_00457](#), [SRS_BSW_00360](#), [SRS_BSW_00333](#)

[

Service Name	<Customer>_TimeNotificationCallback<TimeBase>	
Syntax	Std_ReturnType <Customer>_TimeNotificationCallback<TimeBase> (StbM_TimeDiffType deviationTime)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	deviationTime	Difference time value when callback is called by StbM.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	This callback notifies the <Customer>, when a Time Base reaches the time value set by StbM_StartTimer for the <TimeBase>	
Available via	StbM_Externals.h	

]

[SWS_StbM_00274]

Upstream requirements: [RS_TS_00017](#)

[The event notification callback function shall be set by the parameter [StbMTimeNotificationCallback](#)]

8.6.3.4 SPduTxConfirmationFct

[SWS_StbM_91022] Definition of configurable interface SPduTxConfirmationFct

Upstream requirements: [RS_TS_00039](#)

[

Service Name	SPduTxConfirmationFct	
Syntax	void SPduTxConfirmationFct (uint16 StbMFreshnessValueId)	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	StbMFreshnessValueId	Holds the identifier of the freshness value
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This interface is used by the StbM to indicate that the Secured Time Synchronization Message has been initiated for transmission.	
Available via	StbM_Externals.h	

]

[SWS_StbM_00551]

Upstream requirements: [RS_TS_00039](#)

[The [SPduTxConfirmationFct](#) function shall be set by the parameter [StbMGetTxConfFreshnessValueFuncName](#).]

8.6.3.5 GetTxFreshnessTruncDataFct

[SWS_StbM_91023] Definition of configurable interface GetTxFreshnessTruncDataFct

Upstream requirements: [RS_TS_00039](#)

[

Service Name	GetTxFreshnessTruncDataFct	
Syntax	<pre>Std_ReturnType GetTxFreshnessTruncDataFct (uint16 StbMFreshnessValueId, uint8* StbMFreshnessValue, uint32* StbMFreshnessValueLength, uint8* StbMTruncatedFreshnessValue, uint32* StbMTruncatedFreshnessValueLength)</pre>	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	StbMFreshnessValueId	Holds the identifier of the freshness value
Parameters (inout)	StbMFreshnessValueLength	Holds the length of the provided freshness in bits
	StbMTruncatedFreshnessValueLength	Provides the truncated freshness length configured for this freshness. The caller may adapt the value if needed or can leave it unchanged if the configured length and provided length is the same.
Parameters (out)	StbMFreshnessValue	Holds the current freshness value
	StbMTruncatedFreshnessValue	Holds the truncated freshness to be included into the Secured time sync message. The parameter is optional.
Return value	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed, a Freshness Value cannot be provided due to general issues for Freshness for this FreshnessValueId STBM_E_BUSY: The Freshness information can temporarily not be provided.
Description	This interface is used by the StbM to obtain the current freshness value. The interface function provides also the truncated freshness transmitted in the secured time sync message.	
Available via	StbM_Externals.h	

]

[SWS_StbM_00552]

Upstream requirements: [RS_TS_00039](#)

[The [GetTxFreshnessTruncDataFct](#) function shall be set by the parameter [StbMGetTxTruncFreshnessValueFuncName](#).]

8.6.3.6 GetTxFreshnessFct

[SWS_StbM_91024] Definition of configurable interface GetTxFreshnessFct

Upstream requirements: [RS_TS_00039](#)

[

Service Name	GetTxFreshnessFct	
Syntax	<pre>Std_ReturnType GetTxFreshnessFct (uint16 StbMFreshnessValueId, uint8* StbMFreshnessValue, uint32* StbMFreshnessValueLength)</pre>	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	StbMFreshnessValueId	Holds the identifier of the freshness value
Parameters (inout)	StbMFreshnessValue Length	Holds the length of the provided freshness in bits
Parameters (out)	StbMFreshnessValue	Holds the current freshness value
Return value	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed, a Freshness Value cannot be provided due to general issues for Freshness for this FreshnessValueId STBM_E_BUSY: The Freshness information can temporarily not be provided.
Description	This API returns the freshness value from the Most Significant Bits in the first byte, of the Freshness array, in big endian format	
Available via	StbM_Externals.h	

]

[SWS_StbM_00553]

Upstream requirements: [RS_TS_00039](#)

[The `GetTxFreshnessFct` function shall be set by the parameter `StbMGetTxFreshnessValueFuncName`.]

8.6.3.7 GetRxFreshnessFct

[SWS_StbM_91025] Definition of configurable interface GetRxFreshnessFct

Upstream requirements: [RS_TS_00039](#)

Service Name	GetRxFreshnessFct	
Syntax	<pre>Std_ReturnType GetRxFreshnessFct (uint16 StbMFreshnessValueId, const uint8* StbMTruncatedFreshnessValue, uint32 StbMTruncatedFreshnessValueLength, uint16 StbMAuthVerifyAttempts, uint8* StbMFreshnessValue, uint32* StbMFreshnessValueLength)</pre>	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	StbMFreshnessValueId	Holds the identifier of the freshness value.
	StbMTruncatedFreshnessValue	Holds the truncated freshness value that was contained in the Secured time sync message.
	StbMTruncatedFreshnessValueLength	Holds the length in bits of the truncated freshness value.
	StbMAuthVerifyAttempts	Holds the number of authentication verify attempts of this time sync message since the last reception. The value is 0 for the first attempt and incremented on every unsuccessful verification attempt.
Parameters (inout)	StbMFreshnessValueLength	Holds the length in bits of the freshness value
Parameters (out)	StbMFreshnessValue	Holds the current freshness value
Return value	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed, a Freshness Value cannot be provided due to general issues for Freshness for this FreshnessValueId STBM_E_BUSY: The Freshness information can temporarily not be provided
Description	This interface is used by the StbM to query the current freshness value.	
Available via	StbM_Externals.h	

[SWS_StbM_00554]

Upstream requirements: [RS_TS_00039](#)

[The `GetRxFreshnessFct` function shall be set by the parameter `Stb-MGetRxFreshnessValueFuncName`.]

8.7 Service Interfaces

This chapter defines the AUTOSAR Interfaces and Ports of the AUTOSAR Service "Synchronized Time-base Manager" (StbM).

The interfaces and ports described here will be visible on the VFB and are used to generate the RTE between application software components and the Synchronized Time-Base Manager.

8.7.1 Provided Ports

8.7.1.1 GlobalTime_Master

[SWS_StbM_00244] Definition of Port GlobalTime_Master_{Name} provided by module StbM

Upstream requirements: [RS_TS_00005](#), [RS_TS_00035](#), [RS_TS_00029](#), [RS_TS_00010](#), [RS_TS_00015](#)

[

Name	GlobalTime_Master_{Name}		
Kind	ProvidedPort	Interface	GlobalTime_Master_{Name}
Description	-		
Port Defined Argument Value(s)	Type	StbM_SynchronizedTimeBaseType	
	Value	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier.value)}	
Variation	(({ecuc(StbM/StbMSynchronizedTimeBase/StbMIsSystemWideGlobalTimeMaster)} == TRUE) ({ecuc(StbM/StbMSynchronizedTimeBase/StbMAllowSystemWideGlobalTimeMaster)} == TRUE))&& ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 128) Name = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		

]

8.7.1.2 GlobalTime_Slave

[SWS_StbM_00248] Definition of Port GlobalTime_Slave_{Name} provided by module StbM

Upstream requirements: [RS_TS_00005](#), [RS_TS_00030](#), [RS_TS_00031](#), [RS_TS_00035](#), [RS_TS_00014](#)

[

Name	GlobalTime_Slave_{Name}		
Kind	ProvidedPort	Interface	GlobalTime_Slave_{Name}
Description	-		
Port Defined Argument Value(s)	Type	StbM_SynchronizedTimeBaseType	
	Value	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier.value)}	

▽



Variation	Name = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}
------------------	---

]

8.7.1.3 GlobalTime_StatusEvent

[SWS_StbM_00290] Definition of Port GlobalTime_StatusEvent_{TBName} provided by module StbM

Upstream requirements: [RS_TS_00035](#), [RS_TS_00016](#)

[

Name	GlobalTime_StatusEvent_{TBName}		
Kind	ProvidedPort	Interface	StatusNotification
Description	-		
Variation	(({ecuc(StbM/StbMSynchronizedTimeBase/StbMNotificationInterface)} == SR_INTERFACE {ecuc(StbM/StbMSynchronizedTimeBase/StbMNotificationInterface)} == CALLBACK_AND_SR_ INTERFACE)) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBase Identifier)} < 128) TBName = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		

]

8.7.1.4 StartTimer

[SWS_StbM_91004] Definition of Port StartTimer_{TimeBase}_{Customer} provided by module StbM

Upstream requirements: [RS_TS_00017](#)

[

Name	StartTimer_{TimeBase}_{Customer}		
Kind	ProvidedPort	Interface	StartTimer
Description	-		
Port Defined Argument Value(s)	Type	StbM_SynchronizedTimeBaseType	
	Value	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBase Identifier.value)}	
	Type	StbM_CustomerIdType	
	Value	{ecuc(StbM/StbMSynchronizedTimeBase/StbMNotificationCustomer/StbMNotificationCustomerId.value)}	





Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 128 TimeBase = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)} Customer = {ecuc(StbM/StbMSynchronizedTimeBase/StbMNotificationCustomer.SHORT-NAME)}
------------------	---

]

8.7.2 Required Ports

8.7.2.1 GlobalTime_TimeEvent

[SWS_StbM_00276] Definition of Port GlobalTime_TimeEvent_{TBName}_{CName} required by module StbM

Upstream requirements: [RS_TS_00035](#), [RS_TS_00017](#)

[

Name	GlobalTime_TimeEvent_{TBName}_{CName}		
Kind	RequiredPort	Interface	TimeNotification
Description	-		
Variation	({ecuc(StbM/StbMSynchronizedTimeBase/StbMNotificationCustomer/StbMTimeNotificationCallback)}==NULL) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 128) TBName={ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)} CName={ecuc(StbM/StbMSynchronizedTimeBase/StbMNotificationCustomer.SHORT-NAME)}		

]

8.7.2.2 GlobalTime_Measurement

[SWS_StbM_00387] Definition of Port MeasurementNotification_{TBName} required by module StbM

Upstream requirements: [RS_TS_00034](#)

[

Name	MeasurementNotification_{TBName}		
Kind	RequiredPort	Interface	MeasurementNotification_{TB_Name}
Description	-		





Variation	<pre>(({ecuc(StbM/StbMSynchronizedTimeBase/StbMIsSystemWideGlobalTimeMaster)} == FALSE) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMAllowSystemWideGlobalTimeMaster)} == FALSE)) && ({ecuc(StbM/StbMGeneral/StbMTimeRecordingSupport)} == True) && ({ecuc(StbM/Stb MSynchronizedTimeBase/StbMSynchronizedTimeBaseType)} == TBTYP_ SYNCHRONIZED) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMTimeRecording/StbMSyncTimeRecordBlock Callback)}==NULL) TBName={ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}</pre>
------------------	---

]

8.7.2.3 TimeBaseProviderNotification_Eth

[SWS_StbM_00458] Definition of Port TimeBaseProviderNotification_Eth_{TB_Name} required by module StbM

Status: DRAFT

Upstream requirements: [RS_TS_00034](#)

[

Name	TimeBaseProviderNotification_Eth_{TB_Name} (draft)		
Kind	RequiredPort	Interface	TimeBaseProviderNotification_Eth_{TB_Name}
Description	<p>–</p> <p>Tags: atp.Status=draft</p>		
Variation	<pre>(({ecuc(StbM/StbMSynchronizedTimeBase/StbMTimeValidation)} != NULL) && ({ecuc(StbM/Stb MSynchronizedTimeBase/StbMSynchronizedTimeBaseType)} == TBTYP_ SYNCHRONIZED) && ({ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(EthTSyn/EthTSynGlobalTimeDomain/Eth TSynSynchronizedTimeBaseRef->StbMSynchronizedTimeBase)})) TBName={ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}</pre>		

]

8.7.2.4 TimeBaseProviderNotification_Fr

[SWS_StbM_00459] Definition of Port TimeBaseProviderNotification_Fr_{TB_Name} required by module StbM

Status: DRAFT

Upstream requirements: [RS_TS_00034](#)

[

Name	TimeBaseProviderNotification_Fr_{TB_Name} (draft)		
Kind	RequiredPort	Interface	TimeBaseProviderNotification_Fr_{TB_Name}
Description	<p>–</p> <p>Tags: atp.Status=draft</p>		
Variation	<p>{ecuc(StbM/StbMSynchronizedTimeBase/StbMTimeValidation)} != NULL && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseType)} == TBTYP_ SYNCHRONIZED) && ({ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(FrTSyn/FrTSynGlobalTimeDomain/FrTSynSynchronizedTimeBaseRef->StbMSynchronizedTimeBase)})</p> <p>TBName={ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}</p>		

]

8.7.2.5 TimeBaseProviderNotification_Can

[SWS_StbM_00460] Definition of Port TimeBaseProviderNotification_Can_{TB_Name} required by module StbM

Status: DRAFT

Upstream requirements: [RS_TS_00034](#)

[

Name	TimeBaseProviderNotification_Can_{TB_Name} (draft)		
Kind	RequiredPort	Interface	TimeBaseProviderNotification_Can_{TB_Name}
Description	<p>–</p> <p>Tags: atp.Status=draft</p>		
Variation	<p>{ecuc(StbM/StbMSynchronizedTimeBase/StbMTimeValidation)} != NULL && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseType)} == TBTYP_ SYNCHRONIZED) && ({ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(CanTSyn/CanTSynGlobalTimeDomain/CanTSynSynchronizedTimeBaseRef->StbMSynchronizedTimeBase)})</p> <p>TBName={ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}</p>		

]

8.7.2.6 FreshnessManagement

[SWS_StbM_91019] Definition of Port FreshnessManagement required by module StbM

Upstream requirements: [RS_TS_00039](#)

[

Name	FreshnessManagement		
Kind	RequiredPort	Interface	FreshnessManagement
Description	Port for the provision of freshness for StbM.		
Variation	((ecuc(StbM/StbMFreshnessValueInformation/StbMQueryFreshnessValue)) == SERVICE)		

]

8.7.3 Sender-Receiver Interfaces

8.7.3.1 StatusNotification

[SWS_StbM_00286] Definition of SenderReceiverInterface StatusNotification

Upstream requirements: [RS_TS_00035](#), [RS_TS_00016](#)

[

Name	StatusNotification		
Comment	Notification about a Time Base related status change		
IsService	true		
Variation	--		
Data Elements	eventNotification		
	Type	StbM_TimeBaseNotificationType	
	Variation	-	

]

8.7.4 Client-Server-Interfaces

8.7.4.1 GlobalTime_Master

[SWS_StbM_00240] Definition of ClientServerInterface GlobalTime_Master_{Name}

Upstream requirements: [RS_TS_00005](#), [RS_TS_00035](#), [RS_TS_00010](#), [RS_TS_00015](#), [RS_TS_00011](#), [RS_TS_20069](#), [RS_TS_00038](#)

[

Name	GlobalTime_Master_{Name}		
Comment	–		
IsService	true		
Variation	(({ecuc(StbM/StbMSynchronizedTimeBase/StbMIsSystemWideGlobalTimeMaster)} == TRUE) ({ecuc(StbM/StbMSynchronizedTimeBase/StbMAllowSystemWideGlobalTimeMaster)} == TRUE)) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 128) Name = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	Clone		
Comment	Copies Time Base data (current time, user data, rate correction) from Source Time Base to Destination Time Base. The Source Time Base is identified by the parameter StbMSourceTime Base (ECUC_StbM_00074)		
Mapped to API	StbM_CloneTimeBase		
Variation	({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseType)} == TBTYPE_ SYNCHRONIZED) ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBase Type)} == TBTYPE_PURELOCAL)		
Parameters	cloneCfg		
	Type	StbM_CloneConfigType	
	Direction	IN	
	Comment	Refines how source Time Base is cloned to destination	
	Variation	–	
	statusMask		
	Type	StbM_TimeBaseStatusType	
	Direction	IN	
	Comment	Status flags mask for definition of relevant status flags	
	Variation	–	
	statusValue		
	Type	StbM_TimeBaseStatusType	
	Direction	IN	
Comment	Status flags value define whether cloning shall take place		
Variation	–		
Possible Errors	E_OK E_NOT_OK		

Operation	GetMasterConfig	
Comment	Indicates in postbuild use case, if the StbM is actually configured as system wide master	
Mapped to API	StbM_GetMasterConfig	
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMAllowSystemWideGlobalTimeMaster)} != NULL	
Parameters	masterConfig	
	Type	StbM_MasterConfigType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	SetBusProtocolParam	
Comment	Operation is used to set bus specific parameters for a Time Master	
Mapped to API	StbM_SetBusProtocolParam	
Variation	((ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseType)) == TBTYPE_SYNCHRONIZED) &&((ecuc(StbM/StbMSynchronizedTimeBase)) == {ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynSynchronizedTimeBaseRef->StbMSynchronizedTimeBase)}))	
Parameters	protocolParams	
	Type	StbM_ProtocolParamType
	Direction	IN
	Comment	Structure with Follow_Up information TLV parameters
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	SetCloningOffset	
Comment	Sets an offset time that will be added to the synchronized time when cloning from a Source Time Base to a Destination Time Base.	
Mapped to API	StbM_SetCloningOffset	
Variation	–	
Parameters	timeOffset	
	Type	StbM_TimeStampType
	Direction	IN
	Comment	Offset value for cloning
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	SetGlobalTime	
Comment	Allows the Customers to set the Global Time that will be sent to the buses and modify HW registers behind the providers, if supported. This function will be used if a Time Master is present in this ECU. Using SetGlobalTime can lead to an immediate transmission of the Global Time.	
Mapped to API	StbM_SetGlobalTime	
Variation	–	
Parameters	timeStamp	
	Type	StbM_TimeStampType
	Direction	IN





	Comment	–
	Variation	–
	userData	
	Type	StbM_UserDataType
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	SetRateCorrection	
Comment	Allows to set the rate of a Synchronized Time Base (being either a Pure Local Time Base or not).	
Mapped to API	StbM_SetRateCorrection	
Variation	–	
Parameters	rateDeviation	
	Type	StbM_RateDeviationType
	Direction	IN
	Comment	Value of the applied rate deviation
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	SetUserData	
Comment	Allows the Customers to set the User Data that will be sent to the buses.	
Mapped to API	StbM_SetUserData	
Variation	–	
Parameters	userData	
	Type	StbM_UserDataType
	Direction	IN
	Comment	New user data
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	TriggerTimeTransmission	
Comment	Allows the Customers to force the Timesync Modules to transmit the current Time Base due to an incremented timeBaseUpdateCounter	
Mapped to API	StbM_TriggerTimeTransmission	
Variation	({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseType)} == TBTYPE_SYNCHRONIZED)	
Possible Errors	E_OK E_NOT_OK	

Operation	UpdateGlobalTime	
Comment	Allows the Customers to set the Global Time that will be sent to the buses and modify HW registers behind the providers, if supported. This function will be used if a Time Master is present in this ECU. Using UpdateGlobalTime will not lead to an immediate transmission of the Global Time.	





Mapped to API	StbM_UpdateGlobalTime	
Variation	–	
Parameters	timeStamp	
	Type	StbM_TimeStampType
	Direction	IN
	Comment	–
	Variation	–
	userData	
	Type	StbM_UserDataType
	Direction	IN
Possible Errors	E_OK	
	E_NOT_OK	

]

8.7.4.2 GlobalTime_Slave

[SWS_StbM_00247] Definition of ClientServerInterface GlobalTime_Slave_{Name}

Upstream requirements: [RS_TS_00005](#), [RS_TS_00035](#), [RS_TS_00014](#), [RS_TS_00017](#), [RS_TS_00034](#), [RS_TS_20069](#)

[

Name	GlobalTime_Slave_{Name}		
Comment	–		
IsService	true		
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 128 Name = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	GetBusProtocolParam		
Comment	Operation is used to get bus specific parameters for a Time Master or Time Slave		
Mapped to API	StbM_GetBusProtocolParam		
Variation	({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseType)} == TBTYPE_SYNCHRONIZED) &&({ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynSynchronizedTimeBaseRef->StbMSynchronizedTimeBase)})		
Parameters	protocolParams		
	Type	StbM_ProtocolParamType	
	Direction	OUT	
	Comment	Structure with Follow_Up information TLV parameters	





	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	GetCurrentSafeTime	
Comment	Returns a time triple (Local Time, Fallback Local Time, Global time and timebase status) and user data details.	
Mapped to API	StbM_GetCurrentSafeTime	
Variation	–	
Parameters	timeTuple	
	Type	StbM_TimeTripleType
	Direction	OUT
	Comment	–
	Variation	–
	userData	
	Type	StbM_UserDataType
	Direction	OUT
Possible Errors	–	

Operation	GetCurrentTime	
Comment	Returns the current Time Tuple [Local Time Base derived from Global Time Base; Virtual Local Time] together with the user data of the Time Base	
Mapped to API	StbM_GetCurrentTime	
Variation	–	
Parameters	timeTuple	
	Type	StbM_TimeTupleType
	Direction	OUT
	Comment	–
	Variation	–
	userData	
	Type	StbM_UserDataType
	Direction	OUT
Possible Errors	E_OK E_NOT_OK	

Operation	GetRateDeviation	
Comment	Returns value of the current rate deviation of a Time Base	
Mapped to API	StbM_GetRateDeviation	
Variation	–	
Parameters	rateDeviation	
	Type	StbM_RateDeviationType
	Direction	OUT
	Comment	Value of the current rate deviation of a Time Base
Variation	–	





Possible Errors	E_OK E_NOT_OK
------------------------	------------------

Operation	GetSyncTimeRecordHead
Comment	Reads the header of the table with recorded measurement data belonging to the Synchronized Time Base
Mapped to API	StbM_GetSyncTimeRecordHead
Variation	(({ecuc(StbM/StbMSynchronizedTimeBase/StbMIsSystemWideGlobalTimeMaster)} == FALSE) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMAllowSystemWideGlobalTimeMaster)} == FALSE)) && ({ecuc(StbM/StbMGeneral/StbMTimeRecordingSupport)} == True) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseType)} == TBTYPE_SYNCHRONIZED) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMTimeRecording/StbMSyncTimeRecordBlockCallback)}==NULL)
Parameters	syncRecordTableHead
	Type StbM_SyncRecordTableHeadType
	Direction OUT
	Comment Header of the table
	Variation –
Possible Errors	E_OK E_NOT_OK

Operation	GetTimeBaseStatus
Comment	Returns detailed status information for a Synchronized (or Pure Local) Time Base.
Mapped to API	StbM_GetTimeBaseStatus
Variation	–
Parameters	syncTimeBaseStatus
	Type StbM_TimeBaseStatusType
	Direction OUT
	Comment Status of the Synchronized (or Pure Local) Time Base
	Variation –
Possible Errors	E_OK E_NOT_OK

Operation	GetTimeLeap
Comment	Returns value of time leap.
Mapped to API	StbM_GetTimeLeap
Variation	(({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseType)} == TBTYPE_SYNCHRONIZED)
Parameters	timeJump
	Type StbM_TimeDiffType
	Direction OUT
	Comment Time leap value
	Variation –
Possible Errors	E_OK E_NOT_OK

]

8.7.4.3 StartTimer

[SWS_StbM_00409] Definition of ClientServerInterface StartTimer

Upstream requirements: [RS_TS_00017](#)

[

Name	StartTimer		
Comment	Interface, which starts a timer for a Time Base		
IsService	true		
Variation	--		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	StartTimer		
Comment	Starts a StbM internal timer, which expires at the given expireTime and which triggers a time notification callback.		
Mapped to API	StbM_StartTimer		
Variation	-		
Parameters	expireTime		
	Type	StbM_TimeStampType	
	Direction	IN	
	Comment	-	
	Variation	-	
Possible Errors	E_OK E_NOT_OK		

]

8.7.4.4 TimeNotification

[SWS_StbM_00275] Definition of ClientServerInterface TimeNotification

Upstream requirements: [RS_TS_00035](#), [RS_TS_00017](#)

[

Name	TimeNotification		
Comment	Notification, which indicates, that the timer has expired, which has been set by StartTimer		
IsService	true		
Variation	--		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	NotifyTime	
Comment	Notification, which indicates, that the timer has expired, which has been set by StbM_StartTimer	
Mapped to API	<Customer>_TimeNotificationCallback<TimeBase>	
Variation	–	
Parameters	deviationTime	
	Type	StbM_TimeDiffType
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

]

8.7.4.5 MeasurementNotification

[SWS_StbM_00339] Definition of ClientServerInterface MeasurementNotification_{TB_Name}

Upstream requirements: [RS_TS_00034](#)

[

Name	MeasurementNotification_{TB_Name}		
Comment	Notifies about the availability of a new recorded measurement data block belonging to the Time Base.		
IsService	true		
Variation	({ecuc(StbM/StbMGeneral/StbMTimeRecordingSupport)} == True) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseType)} == TBTYPE_SYNCHRONIZED)) TBName={ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	SetSyncTimeRecordTable	
Comment	Provides the recorded snapshot data Block of the table belonging to the Synchronized Time Base.	
Mapped to API	SyncTimeRecordBlockCallback<TimeBase>	
Variation	({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseType)} == TBTYPE_SYNCHRONIZED)	
Parameters	syncRecordTableBlock	
	Type	StbM_SyncRecordTableBlockType
	Direction	IN
	Comment	Block of the table
	Variation	–
Possible Errors	E_OK E_NOT_OK	

]

8.7.4.6 TimeBaseProviderNotification_Eth

[SWS_StbM_00461] Definition of ClientServerInterface TimeBaseProviderNotification_Eth_{TB_Name}

Status: DRAFT

Upstream requirements: [RS_TS_00034](#)

[

Name	TimeBaseProviderNotification_Eth_{TB_Name} (draft)		
Comment	Notifies about the availability of a new Ethernet specific data block recorded for the Time Base. Tags: atp.Status=draft		
IsService	true		
Variation	<pre> {{ecuc(StbM/StbMSynchronizedTimeBase/StbMTimeValidation)} != NULL} && {{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseType)} == TBTYPPE_SYNCHRONIZED) && {{ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(EthTSyn/EthTSynGlobalTime Domain/EthTSynSynchronizedTimeBaseRef->StbMSynchronizedTimeBase)}} TB_Name ={{ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}} </pre>		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	SetMasterTimingData		
Comment	Provides the recorded data block for the Time Master of the Time Base.		
Mapped to API	-		
Variation	<pre> {{ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynPortRole/EthTSynGlobalTime Master)}!=NULL) </pre>		
Parameters	measurementData		
	Type	StbM_EthTimeMasterMeasurementType	
	Direction	IN	
	Comment	Block of the table	
	Variation	-	
Possible Errors	E_OK E_NOT_OK		

Operation	SetPdelayInitiatorData		
Comment	Provides the recorded data block for the pDelay Initiator of the Time Base.		
Mapped to API	-		
Variation	<pre> ({{ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynPortRole/EthTSynGlobalTime Slave)}!=NULL) </pre>		
Parameters	measurementData		
	Type	StbM_PdelayInitiatorMeasurementType	
	Direction	IN	
	Comment	Block of the table	
	Variation	-	
Possible Errors	E_OK E_NOT_OK		

Operation	SetPdelayResponderData	
Comment	Provides the recorded data block for the pDelay Responder of the Time Base.	
Mapped to API	–	
Variation	(({ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynPortRole/EthTSynGlobalTime Master)}!=NULL)	
Parameters	measurementData	
	Type	StbM_PdelayResponderMeasurementType
	Direction	IN
	Comment	Block of the table
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	SetSlaveTimingData	
Comment	Provides the recorded data block for the Time Slave of the Time Base.	
Mapped to API	–	
Variation	(({ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynPortRole/EthTSynGlobalTime Slave)}!=NULL)	
Parameters	measurementData	
	Type	StbM_EthTimeSlaveMeasurementType
	Direction	IN
	Comment	Block of the table
	Variation	–
Possible Errors	E_OK E_NOT_OK	

]

8.7.4.7 TimeBaseProviderNotification_Fr

[SWS_StbM_00462] Definition of ClientServerInterface TimeBaseProviderNotification_Fr_{TB_Name}

Status: DRAFT

Upstream requirements: [RS_TS_00034](#)

[

Name	TimeBaseProviderNotification_Fr_{TB_Name} (draft)
Comment	Notifies about the availability of a new Flexray specific data block recorded for the Time Base. Tags: atp.Status=draft
IsService	true





Variation	(({ecuc(StbM/StbMSynchronizedTimeBase/StbMTimeValidation)} != NULL) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseType)} == TBTYP_ SYNCHRONIZED) && ({ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(FrTSyn/FrTSynGlobalTimeDomain/FrTSynSynchronizedTimeBaseRef->StbMSynchronizedTimeBase)}) TB_Name =({ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)})		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	SetMasterTimingData		
Comment	Provides the recorded data block for the Time Master of the Time Base.		
Mapped to API	-		
Variation	({ecuc(FrTSyn/FrTSynGlobalTimeDomain/FrTSynGlobalTimeMaster)}!=NULL)		
Parameters	measurementData		
	Type	StbM_FrTimeMasterMeasurementType	
	Direction	IN	
	Comment	Block of the table	
	Variation	-	
Possible Errors	E_OK E_NOT_OK		

Operation	SetSlaveTimingData		
Comment	Provides the recorded data block for the Time Slave of the Time Base.		
Mapped to API	-		
Variation	({ecuc(FrTSyn/FrTSynGlobalTimeDomain/FrTSynGlobalTimeSlave)}!=NULL)		
Parameters	measurementData		
	Type	StbM_FrTimeSlaveMeasurementType	
	Direction	IN	
	Comment	Block of the table	
	Variation	-	
Possible Errors	E_OK E_NOT_OK		

]

8.7.4.8 TimeBaseProviderNotification_Can

[SWS_StbM_00463] Definition of ClientServerInterface TimeBaseProviderNotification_Can_{TB_Name}

Status: DRAFT

Upstream requirements: [RS_TS_00034](#)

[

Name	TimeBaseProviderNotification_Can_{TB_Name} (draft)		
Comment	Notifies about the availability of a new CAN specific data block recorded for the Time Base. Tags: atp.Status=draft		
IsService	true		
Variation	({ecuc(StbM/StbMSynchronizedTimeBase/StbMTimeValidation)} != NULL) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseType)} == TBTYPETYPE_SYNCHRONIZED) && ({ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(CanTSyn/CanTSynGlobalTimeDomain/CanTSynSynchronizedTimeBaseRef->StbMSynchronizedTimeBase)}) TB_Name = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	SetMasterTimingData		
Comment	Provides the recorded data block for the Time Master of the Time Base.		
Mapped to API	-		
Variation	({ecuc(CanTSyn/CanTSynGlobalTimeDomain/CanTSynGlobalTimeMaster)} != NULL)		
Parameters	measurementData		
	Type	StbM_CanTimeMasterMeasurementType	
	Direction	IN	
	Comment	Block of the table	
	Variation	-	
Possible Errors	E_OK E_NOT_OK		

Operation	SetSlaveTimingData		
Comment	Provides the recorded data block for the Time Slave of the Time Base.		
Mapped to API	-		
Variation	({ecuc(CanTSyn/CanTSynGlobalTimeDomain/CanTSynGlobalTimeSlave)} != NULL)		
Parameters	measurementData		
	Type	StbM_CanTimeSlaveMeasurementType	
	Direction	IN	
	Comment	Block of the table	
	Variation	-	
Possible Errors	E_OK E_NOT_OK		

]

8.7.4.9 FreshnessManagement

[SWS_StbM_91026] Definition of ClientServerInterface FreshnessManagement

Upstream requirements: [RS_TS_00039](#)

[

Name	FreshnessManagement		
Comment	Freshness Management for StbM		
IsService	true		
Variation	({ecuc(StbM/StbMFreshnessValueInformation/StbMQueryFreshnessValue)} == SERVICE)		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	STBM_E_BUSY	Operation temporary failed, a freshness cannot be provided at the moment.

Operation	GetRxFreshness		
Comment	This interface is used by the StbM to obtain the current freshness value.		
Mapped to API	GetRxFreshnessFct		
Variation	({ecuc(CanTSyn/CanTSynGlobalTimeDomain/CanTSynGlobalTimeSlave)}!=NULL) ({ecuc(FrTSyn/FrTSynGlobalTimeDomain/FrTSynGlobalTimeSlave)}!=NULL) ({ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynPortRole/EthTSynGlobalTimeSlave)}!=NULL)		
Parameters	freshnessValueId		
	Type	uint16	
	Direction	IN	
	Comment	Identifier of the freshness	
	Variation	–	
	truncatedFreshnessValue		
	Type	StbM_FreshnessArrayType	
	Direction	IN	
	Comment	The truncated freshness value from the received Secured time sync message	
	Variation	–	
	truncatedFreshnessValueLength		
	Type	uint32	
	Direction	IN	
	Comment	Length in bits of the truncated freshness value	
	Variation	–	
	authVerifyAttempts		
	Type	uint16	
	Direction	IN	
	Comment	The number of authentication verify attempts for the current time sync message	
	Variation	–	
	freshnessValue		
	Type	StbM_FreshnessArrayType	
	Direction	OUT	
	Comment	The freshness value for this time sync message	
Variation	–		
freshnessValueLength			
Type	uint32		
Direction	INOUT		
Comment	The freshness value length in bits		
Variation	–		





Possible Errors	E_OK E_NOT_OK STBM_E_BUSY
------------------------	---

Operation	GetTxFreshness	
Comment	Returns the freshness value in big endian format.	
Mapped to API	GetTxFreshnessFct	
Variation	({{ecuc(CanTSyn/CanTSynGlobalTimeDomain/CanTSynGlobalTimeMaster)}!=NULL) ({{ecuc(FrTSyn/FrTSynGlobalTimeDomain/FrTSynGlobalTimeMaster)}!=NULL) ({{ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynPortRole/EthTSynGlobalTimeMaster)}!=NULL)	
Parameters	freshnessValueId	
	Type	uint16
	Direction	IN
	Comment	Identifier of the freshness
	Variation	–
	freshnessValue	
	Type	Stbm_FreshnessArrayType
	Direction	OUT
	Comment	Freshness value
	Variation	–
	freshnessValueLength	
	Type	uint32
Direction	INOUT	
Comment	Length in bits of the freshness value	
Variation	–	
Possible Errors	E_OK E_NOT_OK STBM_E_BUSY	

Operation	GetTxFreshnessTruncData	
Comment	This operation is used by the StbM to obtain the freshness that corresponds to the freshness ValueId. The operation provides the freshness and also the truncated freshness that shall be placed into the Secured time sync message.	
Mapped to API	GetTxFreshnessTruncDataFct	
Variation	({{ecuc(CanTSyn/CanTSynGlobalTimeDomain/CanTSynGlobalTimeMaster)}!=NULL) ({{ecuc(FrTSyn/FrTSynGlobalTimeDomain/FrTSynGlobalTimeMaster)}!=NULL) ({{ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynPortRole/EthTSynGlobalTimeMaster)}!=NULL)	
Parameters	freshnessValueId	
	Type	uint16
	Direction	IN
	Comment	Identifier of the freshness
	Variation	–
	freshnessValue	
	Type	Stbm_FreshnessArrayType
	Direction	OUT
	Comment	Freshness value
	Variation	–
	freshnessValueLength	
	Type	uint32





	Direction	INOUT
	Comment	Length in bits of the freshness value
	Variation	–
	truncatedFreshnessValue	
	Type	StbM_FreshnessArrayType
	Direction	OUT
	Comment	The truncated freshness value that has to be placed into the Secured time sync message
	Variation	–
	truncatedfreshnessValueLength	
	Type	uint32
	Direction	INOUT
	Comment	The length in bits for the truncated freshness.
Variation	–	
Possible Errors	E_OK E_NOT_OK STBM_E_BUSY	

Operation	SPduTxConfirmation	
Comment	This operation is used by the StbM to indicate that the Secured Time Synchronization Message has been initiated for transmission.	
Mapped to API	SPduTxConfirmationFct	
Variation	({{ecuc(CanTSyn/CanTSynGlobalTimeDomain/CanTSynGlobalTimeMaster)}}!=NULL) ({{ecuc(FrTSyn/FrTSynGlobalTimeDomain/FrTSynGlobalTimeMaster)}}!=NULL) ({{ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynPortRole/EthTSynGlobalTimeMaster)}}!=NULL)	
Parameters	freshnessValueId	
	Type	uint16
	Direction	IN
	Comment	Identifier of the freshness
	Variation	–
Possible Errors	E_OK	

└

8.7.5 Implementation Data Types

This chapter specifies the data types which will be used for the service port interfaces for accessing the Synchronized Time-Base Manager service.

The implementation header defines additionally those data types, which are listed in chapter 8.2 “[Type definitions](#)”, if not included by the application types header.

8.7.5.1 StbM_PortIdType

[SWS_StbM_00483] Definition of ImplementationDataType StbM_PortIdType

Upstream requirements: [RS_TS_00034](#)

[

Name	StbM_PortIdType		
Kind	Structure		
Elements	clockIdentity		
	Type	uint64	
	Comment	ClockIdentity of the clock	
	portNumber		
	Type	uint16	
	Comment	Number of Ethernet port	
Description	Structure which contains port identity data		
Variation	({ecuc(EthTSyn/EthTSynGeneral/EthTSynTimeValidationSupport)} == True)		
Available via	Rte_StbM_Type.h		

]

8.7.5.2 StbM_SynchronizedTimeBaseType

[SWS_StbM_00142] Definition of ImplementationDataType StbM_SynchronizedTimeBaseType

Upstream requirements: [SRS_BSW_00305](#), [RS_TS_00005](#), [RS_TS_00032](#), [RS_TS_00035](#)

[

Name	StbM_SynchronizedTimeBaseType		
Kind	Type		
Derived from	uint16		
Range	0..2 ¹⁶ -1	-	--
Description	Variables of this type are used to represent the kind of synchronized time-base.		
Variation	-		
Available via	Rte_StbM_Type.h		

]

8.7.5.3 StbM_TimeBaseStatusType

[SWS_StbM_00239] Definition of ImplementationDataType StbM_TimeBaseStatusType

Upstream requirements: [RS_TS_00009](#)

[

Name	StbM_TimeBaseStatusType			
Kind	Bitfield			
Derived from	uint16			
Elements	Kind	Name	Mask	Description
	bit	TIMEOUT	0x01	Bit 0 (LSB): 0x00: No Timeout on receiving Synchronisation Messages 0x01: Timeout on receiving Synchronisation Messages
	bit	reserved	0x02	Bit 1: always 0x00
	bit	SYNC_TO_GATEWAY	0x04	Bit 2 0x00: Local Time Base is synchronous to Global Time Master 0x04: Local Time Base updates are based on a Time Gateway below the Global Time Master
	bit	GLOBAL_TIME_BASE	0x08	Bit 3 0x00: Local Time Base is based on Local Time Base reference clock only (never synchronized with Global Time Base) 0x08: Local Time Base was at least synchronized with Global Time Base one time
	bit	TIMELEAP_FUTURE	0x10	Bit 4 0x00: No leap into the future within the received time for Time Base 0x10: Leap into the future within the received time for Time Base exceeds a configured threshold
	bit	TIMELEAP_PAST	0x20	Bit 5 0x00: No leap into the past within the received time for Time Base 0x20: Leap into the past within the received time for Time Base exceeds a configured threshold
	bit	RATE_CORRECTED	0x40	Bit 6: 0x00: Valid rate correction not calculated for the Time Base 0x40: Valid rate correction calculated for the Time Base
	bit	RATE_EXCEEDED	0x80	Bit 7 0x00: Calculated rate for the Time Base does not exceed limits 0x80: Calculated rate for the Time Base exceeds limits
	bit	PDELAY_EXCEEDED	0x100	Bit 8 0x00: Pdelay within the threshold for Time Base 0x100: Pdelay is exceeding threshold for time base

▽



	bit	RATEJITTERWANDER_EXCEEDED	0x200	Bit 9 0x00: Calculated rate jitter/wander for the Time Base does not exceed limits 0x200: Calculated rate jitter/wander for the Time Base exceeds limits
	bit	TIME_PROGRESSION_INCONSISTENCY	0x400	Bit 10 0x00: Time progression discrepancy within the threshold. 0x400: Time progression discrepancy exceeds the threshold
	bit	FALLBACK_TIME_EXTRAPOLATION	0x800	Bit 11 0x00: Both Virtual Local Times are available for the extrapolation 0x800: only Fallback Virtual Local Time is available for extrapolation
Description	Bits 1 and 12 .. 15 are always 0 (reserved for future usage) Variables of this type are used to express if and how a Local Time Base is synchronized to the Global Time Master. The type is a bitfield of individual status bits, although not every combination is possible, i.e. any bit other than GLOBAL_TIME_BASE can only be set if the GLOBAL_TIME_BASE bit itself is set. PDELAY_EXCEEDED, RATEJITTERWANDER_EXCEEDED and TIME_PROGRESSION_INCONSISTENCY bits can only be set if the Time Validation feature is enabled. FALLBACK_TIME_EXTRAPOLATION bit can only be set if a Fallback Virtual Local Time is configured.			
Variation	-			
Available via	Rte_StbM_Type.h			

]

8.7.5.4 StbM_TimeBaseNotificationType

[SWS_StbM_00287] Definition of ImplementationDataType StbM_TimeBaseNotificationType

Upstream requirements: [RS_TS_00035](#), [RS_TS_00016](#)

[

Name	StbM_TimeBaseNotificationType			
Kind	Bitfield			
Derived from	uint32			
Elements	Kind	Name	Mask	Description
	bit	EV_GLOBAL_TIME	0x01	Bit 0 (LSB): 0: synchronization to global time master not changed 1: GLOBAL_TIME_BASE in StbM_TimeBaseStatusType has changed from 0 to 1
	bit	EV_TIMEOUT_OCCURRED	0x02	Bit 1: 1: TIMEOUT bit in time BaseStatus has changed from 0 to 1 0: otherwise
	bit	EV_TIMEOUT_REMOVED	0x04	Bit 2 1: TIMEOUT bit in time BaseStatus has changed from 1 to 0 0: otherwise





bit	EV_TIMELEAP_FUTURE	0x08	Bit 3 1: TIMELEAP_FUTURE bit in timeBaseStatus has changed from 0 to 1 0: otherwise
bit	EV_TIMELEAP_FUTURE_REMOVED	0x10	Bit 4 1: TIMELEAP_FUTURE bit in timeBaseStatus has changed from 1 to 0 0: otherwise
bit	EV_TIMELEAP_PAST	0x20	Bit 5 1: TIMELEAP_PAST bit in timeBaseStatus has changed from 0 to 1 0: otherwise
bit	EV_TIMELEAP_PAST_REMOVED	0x40	Bit 6 1: TIMELEAP_PAST bit in timeBaseStatus has changed from 1 to 0 0: otherwise
bit	EV_SYNC_TO_SUBDOMAIN	0x80	Bit 7 1: SYNC_TO_GATEWAY bit in timeBaseStatus has changed from 0 to 1 0: otherwise
bit	EV_SYNC_TO_GLOBAL_MASTER	0x100	Bit 8 1: SYNC_TO_GATEWAY bit of Time Domain changes from 1 to 0 0: otherwise
bit	EV_RESYNC	0x0200	Bit 9: 1: A synchronization of the local time to the valid Global Time value has occurred 0: No resynchronization event occurred
bit	EV_RATECORRECTION	0x0400	Bit 10 1: a valid rate correction has been calculated (not beyond limits) 0: No rate correction calculated
bit	EV_RATE_EXCEEDED	0x0800	Bit 11: 1: An invalid rate correction has been calculated (i.e., beyond limits) 0: No invalid rate correction calculated
bit	EV_TIME_PROGRESSION_INCONSISTENCY	0x1000	Bit 12: 1: TIME_PROGRESSION_INCONSISTENCY bit in time BaseStatus has changed from 0 to 1 0: otherwise
bit	EV_TIME_PROGRESSION_INCONSISTENCY_REMOVED	0x2000	Bit 13: 1: TIME_PROGRESSION_INCONSISTENCY bit in time BaseStatus has changed from 1 to 0 0: otherwise
bit	EV_RATEJITTERWANDER_EXCEEDED	0x4000	Bit 14: 1: RATEJITTERWANDER_EXCEEDED bit in timeBase Status has changed from 0 to 1 0: otherwise
bit	EV_RATEJITTERWANDER_EXCEEDED_REMOVED	0x8000	Bit 15: 1: RATEJITTERWANDER_EXCEEDED bit in timeBase Status has changed from 1 to 0 0: otherwise
bit	EV_PDELAY_EXCEEDED	0x10000	Bit 16: 1: PDELAY_EXCEEDED bit in timeBaseStatus has changed from 0 to 1 0: otherwise
bit	EV_PDELAY_EXCEEDED_REMOVED	0x20000	Bit 17: 1: PDELAY_EXCEEDED bit in timeBaseStatus has changed from 1 to 0 0: otherwise
bit	EV_FALLBACK_TIME_EXTRAPOLATION	0x40000	Bit 18: 1: FALLBACK_TIME_EXTRAPOLATION bit in time BaseStatus has changed from 0 to 1 0: otherwise



△

	bit	EV_FALLBACK_TIME_EXTRAPOLATION_REMOVED	0x80000	Bit 19: 1: FALLBACK_TIME_EXTRAPOLATION bit in time BaseStatus has changed from 1 to 0 0: otherwise
Description	The StbM_TimeBaseNotificationType type defines a number of global time related events. The type definition is used for storing the events in the status variable NotificationEvents and for setting the mask variable NotificationMask which defines a subset of events for which an interrupt request shall be raised.			
Variation	–			
Available via	Rte_StbM_Type.h			

]

8.7.5.5 StbM_VirtualLocalTimeType

[SWS_StbM_91003] Definition of ImplementationDataType StbM_VirtualLocalTimeType

Upstream requirements: [RS_TS_00009](#)

[

Name	StbM_VirtualLocalTimeType		
Kind	Structure		
Elements	nanosecondsLo		
	Type	uint32	
	Comment	Least significant 32 bits of the 64 bit Virtual Local Time	
	nanosecondsHi		
	Type	uint32	
	Comment	Most significant 32 bits of the 64 bit Virtual Local Time	
Description	Variables of this type store time stamps of the Virtual Local Time. The unit is nanoseconds.		
Variation	–		
Available via	StbM.h		

]

8.7.5.6 StbM_TimeStampShortType

[SWS_StbM_00482] Definition of ImplementationDataType StbM_TimeStamp ShortType

Upstream requirements: [RS_TS_00034](#)

[

Name	StbM_TimeStampShortType	
Kind	Structure	
Elements	nanoseconds	
	Type	uint32
	Comment	Nanoseconds part of the time
	seconds	
	Type	uint32
	Comment	32 bit LSB of the 48 bits Seconds part of the time
Description	<p>Variables of this type are used for expressing time stamps with a limited range including relative time and absolute calendar time. The absolute time starts from 1970-01-01. 0 to 4.294.967.295 s ~ 136 years</p> <p>0 to 999999999ns [0x3B9A C9FF] invalid value in nanoseconds: [0x3B9A CA00] to [0x3FFF FFFF] Bit 30 and 31 reserved, default: 0</p>	
Variation	-	
Available via	Rte_StbM_Type.h	

]

8.7.5.7 StbM_TimeStampType

[SWS_StbM_00241] Definition of ImplementationDataType StbM_TimeStamp Type

Upstream requirements: [RS_TS_00036](#)

[

Name	StbM_TimeStampType	
Kind	Structure	
Elements	nanoseconds	
	Type	uint32
	Comment	Nanoseconds part of the time
	seconds	
	Type	uint32
	Comment	32 bit LSB of the 48 bits Seconds part of the time
	secondsHi	
	Type	uint16

▽



	Comment	16 bit MSB of the 48 bits Seconds part of the time
Description	Variables of this type are used for expressing time stamps including relative time and absolute calendar time. The absolute time starts from 1970-01-01. 0 to 281474976710655s == 3257812230d [0xFFFF FFFF FFFF] 0 to 999999999ns [0x3B9A C9FF] invalid value in nanoseconds: [0x3B9A CA00] to [0x3FFF FFFF] Bit 30 and 31 reserved, default: 0	
Variation	–	
Available via	Rte_StbM_Type.h	

]

Note: Start of absolute time (1970-01-01) is according to [11, IEEE 802.1 AS], Annex C/C1 (refer to parameter "approximate epoch" for PTP)

8.7.5.8 StbM_TimeTupleType

[SWS_StbM_91013] Definition of ImplementationDataType StbM_TimeTupleType

Upstream requirements: [RS_TS_00005](#)

[

Name	StbM_TimeTupleType	
Kind	Structure	
Elements	virtualLocalTime	
	Type	StbM_VirtualLocalTimeType
	Comment	Virtual Local Time value of the Time Tuple
	globalTime	
	Type	StbM_TimeStampType
	Comment	Global Time part of the Time Tuple
	timeBaseStatus	
	Type	StbM_TimeBaseStatusType
Description	Variables of this type are used for expressing time tuples, which include the global time (as received from the Global Time Master or interpolated locally) and the virtual local time	
Variation	–	
Available via	Rte_StbM_Type.h	

]

8.7.5.9 StbM_TimeTripleType

[SWS_StbM_91028] Definition of ImplementationDataType StbM_TimeTripleType

Upstream requirements: [RS_TS_00005](#)

[

Name	StbM_TimeTripleType		
Kind	Structure		
Elements	virtualLocalTime		
	Type	StbM_VirtualLocalTimeType	
	Comment	Virtual Local Time value of the Time Triple	
	fallbackVirtualLocalTime		
	Type	StbM_VirtualLocalTimeType	
	Comment	Fallback Virtual Local Time value of the Time Triple	
	globalTime		
	Type	StbM_TimeStampType	
	Comment	Global Time part of the Time Triple	
	timeBaseStatus		
Type	StbM_TimeBaseStatusType		
Comment	Status of the Time Base Tags: atp.Status=draft		
Description	Variables of this type are used for expressing time triples, which include the global time (as received from the Global Time Master or interpolated locally), the virtual local time and fallback virtual local time		
Variation	-		
Available via	Rte_StbM_Type.h		

]

8.7.5.10 StbM_TimeDiffType

[SWS_StbM_00300] Definition of ImplementationDataType StbM_TimeDiffType

Upstream requirements: [RS_TS_00010](#)

[

Name	StbM_TimeDiffType		
Kind	Type		
Derived from	sint32		
Range	-2147483647..2147483647	-	nanoseconds (-2147483647 .. 2147483647)
Description	Variables of this type are used to express time differences / offsets as signed values in in nanoseconds		
Variation	-		

▽



Available via	Rte_StbM_Type.h
----------------------	-----------------

]

8.7.5.11 StbM_RateDeviationType

[SWS_StbM_00301] Definition of ImplementationDataType StbM_RateDeviation Type

Upstream requirements: [RS_TS_00017](#)

[

Name	StbM_RateDeviationType		
Kind	Type		
Derived from	sint16		
Range	-32000..32000	–	parts per million (-32000..32000)
Description	Variables of this type are used to express a rate deviation in ppm.		
Variation	–		
Available via	Rte_StbM_Type.h		

]

8.7.5.12 StbM_CloneConfigType

[SWS_StbM_91011] Definition of ImplementationDataType StbM_CloneConfig Type

Upstream requirements: [RS_TS_00038](#)

[

Name	StbM_CloneConfigType			
Kind	Bitfield			
Derived from	uint8			
Elements	Kind	Name	Mask	Description
	bit	DEFERRED_COPY	0x01	True: copy of time information to destination is deferred until Source Time base is updated next time by <bus>TSyn module False: time information copied immediately to Destination Time Base



△

	bit	IMMEDIATE_TX	0x02	True: time information is transmitted on destination bus immediately after cloning False: time information is transmitted on destination bus only on next cyclic transmission after cloning
	bit	APPLY_RATE	0x04	True: Rate correction value of Source Time Base shall be applied to Destination Time Base
	bit	APPLY_OFFSET	0x08	True: Offset value (if set) shall be added
	bit	RETRIGGER_DEFERRED_COPY	0x10	True: Deferred clone requests are made pending again after being consumed
Description	Bitfield to configure the cloning process. Bit 5 .. 7 are always 0 (reserved for future usage).			
Variation	-			
Available via	Rte_StbM_Type.h			

┌

8.7.5.13 StbM_UserDataType

[SWS_StbM_00243] Definition of ImplementationDataType StbM_UserDataType

Upstream requirements: [RS_TS_00014](#), [RS_TS_00015](#)

┌

Name	StbM_UserDataType	
Kind	Structure	
Elements	userDataLength	
	Type	uint8
	Comment	User Data Length in bytes, value range: 0..3
	userByte0	
	Type	uint8
	Comment	User Byte 0
	userByte1	
	Type	uint8
	Comment	User Byte 1
	userByte2	
	Type	uint8
	Comment	User Byte 2
Description	Current user data of the Time Base	
Variation	-	
Available via	Rte_StbM_Type.h	

┌

8.7.5.14 StbM_CustomerIdType

[SWS_StbM_00288] Definition of ImplementationDataType StbM_CustomerIdType

Upstream requirements: [RS_TS_00035](#), [RS_TS_00016](#), [RS_TS_00017](#)

[

Name	StbM_CustomerIdType		
Kind	Type		
Derived from	uint16		
Range	0..65535	–	(0x00..0xFFFF)
Description	unique identifier of a notification customer		
Variation	–		
Available via	Rte_StbM_Type.h		

]

8.7.5.15 StbM_SyncRecordTableHeadType

[SWS_StbM_00331] Definition of ImplementationDataType StbM_SyncRecordTableHeadType

Upstream requirements: [RS_TS_00034](#)

[

Name	StbM_SyncRecordTableHeadType		
Kind	Structure		
Elements	SynchronizedTimeDomain		
	Type	uint8	
	Comment	Time Domain 0..15	
	HWfrequency		
	Type	uint32	
	Comment	HW Frequency in Hz	
	HWprescaler		
	Type	uint32	
Comment	Prescaler value		
Description	Synchronized Time Base Record Table Header		
Variation	–		
Available via	Rte_StbM_Type.h		

]

8.7.5.16 StbM_SyncRecordTableBlockType

[SWS_StbM_00332] Definition of ImplementationDataType StbM_SyncRecordTableBlockType

Upstream requirements: [RS_TS_00034](#)

[

Name	StbM_SyncRecordTableBlockType	
Kind	Structure	
Elements	GlbSeconds	
	Type	uint32
	Comment	Seconds of the Local Time Base directly after synchronization with the Global Time Base
	GlbNanoSeconds	
	Type	uint32
	Comment	Nanoseconds of the Local Time Base directly after synchronization with the Global Time Base
	TimeBaseStatus	
	Type	StbM_TimeBaseStatusType
	Comment	Time Base Status of the Local Time Base directly after synchronization with the Global Time Base
	VirtualLocalTimeLow	
	Type	uint32
	Comment	Least significant 32 bit of the Virtual Local Time directly after synchronization with the Global Time Base
	RateDeviation	
	Type	StbM_RateDeviationType
	Comment	Calculated Rate Deviation directly after rate deviation measurement
	LocSeconds	
	Type	uint32
	Comment	Seconds of the Local Time Base directly before synchronization with the Global Time Base
	LocNanoSeconds	
	Type	uint32
Comment	Nanoseconds of the Local Time Base directly before synchronization with the Global Time Base	
PathDelay		
Type	uint32	
Comment	Current propagation delay in nanoseconds	
FallbackVirtualTimeLow		
Type	uint32	
Comment	Least significant 32 bit of the Fallback Virtual time in nanoseconds	
Description	Synchronized Time Base Record Table Block	
Variation	-	
Available via	Rte_StbM_Type.h	

]

8.7.5.17 StbM_MasterConfigType

[SWS_StbM_91001] Definition of ImplementationDataType StbM_MasterConfigType

Upstream requirements: [RS_TS_00029](#)

[

Name	StbM_MasterConfigType		
Kind	Type		
Derived from	uint8		
Range	STBM_SYSTEM_WIDE_MASTER_DISABLED	0x00	not configured as System Wide Master
	STBM_SYSTEM_WIDE_MASTER_ENABLED	0x01	configured as System Wide Master
Description	This type indicates if an ECU is configured for a system wide master for a given Time Base is available or not.		
Variation	-		
Available via	Rte_StbM_Type.h		

]

8.7.5.18 StbM_EthTimeMasterMeasurementType

[SWS_StbM_00504] Definition of ImplementationDataType StbM_EthTimeMasterMeasurementType

Upstream requirements: [RS_TS_00034](#)

[

Name	StbM_EthTimeMasterMeasurementType		
Kind	Structure		
Elements	sequenceId		
	Type	uint16	
	Comment	sequenceId of sent Ethernet frame	
	sourcePortId		
	Type	StbM_PortIdType	
	Comment	sourcePortId of sending Ethernet port	
	syncEgressTimestamp		
	Type	StbM_VirtualLocalTimeType	
	Comment	Egress timestamp of Sync frame	
	preciseOriginTimestamp		
	Type	StbM_TimeStampShortType	
	Comment	the preciseOriginTime as copied to the Follow_Up frame	





	correctionField
	Type sint64
	Comment the correctionField as copied to the Follow_Up frame
Description	Structure with detailed data for Time Validation of the Time Master on Ethernet
Variation	{{ecuc(EthTSyn/EthTSynGeneral/EthTSynTimeValidationSupport)} == True}
Available via	Rte_StbM_Type.h

]

8.7.5.19 StbM_FrTimeMasterMeasurementType

[SWS_StbM_00505] Definition of ImplementationDataType StbM_FrTimeMaster MeasurementType

Upstream requirements: [RS_TS_00034](#)

[

Name	StbM_FrTimeMasterMeasurementType
Kind	Structure
Elements	sequenceCounter
	Type uint16
	Comment sequence counter of sent Sync frame
	referenceTimestamp
	Type StbM_VirtualLocalTimeType
	Comment Retrieved reference Virtual Local Time used to calculate (future) time value of the Time Base
	preciseOriginTimestamp
	Type StbM_TimeStampShortType
	Comment (future) time value of the Time Base in Global Time
	segmentId
	Type uint8
	Comment network segment id of the physical channel on which the Sync message has been sent
	currentCycle
	Type uint8
	Comment Value of current?Cycle upon transmission of the Sync message
	currentMacroticks
Type uint16	
Comment Value of Current?Macroticks upon transmission of the Sync message	
macrotickDuration	
Type uint16	
Comment Duration of one Macrotick in ns	
cycleLength	



△

	Type	uint32
	Comment	Flexray cycle length in nanoseconds
Description	Structure with detailed data for Time Validation of the Time Master on Flexray	
Variation	({{ecuc(FrTSyn/FrTSynGeneral/FrTSynTimeValidationSupport)}} == True)	
Available via	Rte_StbM_Type.h	

]

8.7.5.20 StbM_CanTimeMasterMeasurementType

[SWS_StbM_00511] Definition of ImplementationDataType StbM_CanTimeMasterMeasurementType

Upstream requirements: [RS_TS_00034](#)

[

Name	StbM_CanTimeMasterMeasurementType	
Kind	Structure	
Elements	sequenceCounter	
	Type	uint16
	Comment	Sequence counter of sent CAN frame
	syncEgressTimestamp	
	Type	StbM_VirtualLocalTimeType
	Comment	Egress timestamp of Sync frame
	preciseOriginTimestamp	
	Type	StbM_TimeStampShortType
	Comment	preciseOriginTimestamp as sent in the Follow up frame
	segmentId	
Type	uint8	
Comment	network segment id of the physical channel on which the Sync message has been sent	
Description	Structure with detailed data for Time Validation of the Time Master on CAN	
Variation	({{ecuc(CanTSyn/CanTSynGeneral/CanTSynTimeValidationSupport)}} == True)	
Available via	Rte_StbM_Type.h	

]

8.7.5.21 StbM_EthTimeSlaveMeasurementType

[SWS_StbM_00506] Definition of ImplementationDataType StbM_EthTimeSlave MeasurementType

Upstream requirements: [RS_TS_00034](#)

[

Name	StbM_EthTimeSlaveMeasurementType	
Kind	Structure	
Elements	sequenceId	
	Type	uint16
	Comment	Sequence Id of received Sync frame
	sourcePortId	
	Type	StbM_PortIdType
	Comment	sourcePortId from received Sync frame
	syncIngressTimestamp	
	Type	StbM_VirtualLocalTimeType
	Comment	Ingress timestamp of Sync frame converted to Virtual Local Time
	preciseOriginTimestamp	
	Type	StbM_TimeStampShortType
	Comment	preciseOriginTimestamp taken from the received Follow_Up frame
	correctionField	
	Type	sint64
	Comment	correctionField taken from the received Follow_Up frame
	pDelay	
	Type	uint32
	Comment	Currently valid pDelay value
	referenceLocalTimestamp	
	Type	StbM_VirtualLocalTimeType
Comment	SyncLocal Time Tuple (Virtual Local Time part)	
referenceGlobalTimestamp		
Type	StbM_TimeStampShortType	
Comment	SyncLocal Time Tuple (Global Time part)	
Description	Structure with detailed data for Time Validation of the Time Slave on Ethernet	
Variation	({ecuc(EthTSyn/EthTSynGeneral/EthTSynTimeValidationSupport)} == True)	
Available via	Rte_StbM_Type.h	

]

8.7.5.22 StbM_FrTimeSlaveMeasurementType

[SWS_StbM_00507] Definition of ImplementationDataType StbM_FrTimeSlave MeasurementType

Upstream requirements: [RS_TS_00034](#)

[

Name	StbM_FrTimeSlaveMeasurementType	
Kind	Structure	
Elements	sequenceCounter	
	Type	uint16
	Comment	Sequence counter of received Sync frame
	syncIngressTimestamp	
	Type	StbM_VirtualLocalTimeType
	Comment	Retrieved reference Virtual Local Time used to calculate (future) time value of the Time Base
	preciseOriginTimestampSec	
	Type	StbM_TimeStampShortType
	Comment	Timestamp contained in received Sync frame
	currentCycle	
	Type	uint8
	Comment	Value of currentCycle used to update the Time Slave's local instance of the Time Base
	currentMacroticks	
	Type	uint16
	Comment	Value of currentMacroticks used to update the Time Slave's local instance of the Time Base
	FCNT	
	Type	uint8
	Comment	FCNT of received Sync frame
	macrotickDuration	
	Type	uint16
	Comment	Duration of one Macrotick in ns
	cycleLength	
	Type	uint32
	Comment	Flexray cycle length in nanoseconds
referenceLocalTimestamp		
Type	StbM_VirtualLocalTimeType	
Comment	SyncLocal Time Tuple (Virtual Local Time part)	
referenceGlobalTimestampSec		
Type	StbM_TimeStampShortType	
Comment	SyncLocal Time Tuple (Global Time part)	
segmentId		
Type	uint8	

▽



	Comment	network segment id of the physical channel on which the Sync message has been received
Description	Structure with detailed data for Time Validation of the Time Slave on Flexray	
Variation	({ecuc(FrTSyn/FrTSynGeneral/FrTSynTimeValidationSupport)}) == True	
Available via	Rte_StbM_Type.h	

]

8.7.5.23 StbM_CanTimeSlaveMeasurementType

[SWS_StbM_00510] Definition of ImplementationDataType StbM_CanTimeSlave MeasurementType

Upstream requirements: [RS_TS_00034](#)

[

Name	StbM_CanTimeSlaveMeasurementType	
Kind	Structure	
Elements	sequenceCounter	
	Type	uint16
	Comment	sequence counter of received Sync frame
	syncIngressTimestamp	
	Type	StbM_VirtualLocalTimeType
	Comment	Ingress timestamp of Sync frame
	preciseOriginTimestamp	
	Type	StbM_TimeStampShortType
	Comment	preciseOriginTimestamp taken from the received Follow_Up frame
	referenceLocalTimestamp	
	Type	StbM_VirtualLocalTimeType
	Comment	SyncLocal Time Tuple (Virtual Local Time part)
	referenceGlobalTimestamp	
	Type	StbM_TimeStampShortType
	Comment	SyncLocal Time Tuple (Global Time part)
	pathDelay	
Type	uint32	
Comment	Current propagation delay in nanoseconds	
segmentId		
Type	uint8	
Comment	network segment id of the physical channel on which the Sync message has been received	
Description	Structure with detailed timing data for the Time Slave on CAN	
Variation	({ecuc(CanTSyn/CanTSynGeneral/CanTSynTimeValidationSupport)}) == True	





Available via	Rte_StbM_Type.h
----------------------	-----------------

]

8.7.5.24 StbM_PdelayInitiatorMeasurementType

[SWS_StbM_00508] Definition of ImplementationDataType StbM_PdelayInitiator MeasurementType

Upstream requirements: [RS_TS_00034](#)

[

Name	StbM_PdelayInitiatorMeasurementType	
Kind	Structure	
Elements	sequenceId	
	Type	uint16
	Comment	Sequence Id of sent Pdelay_Req frame
	requestPortId	
	Type	StbM_PortIdType
	Comment	sourcePortId of sent Pdelay_Req frame
	responsePortId	
	Type	StbM_PortIdType
	Comment	sourcePortId of received Pdelay_Resp frame
	requestOriginTimestamp	
	Type	StbM_VirtualLocalTimeType
	Comment	Egress timestamp of Pdelay_Req in Virtual Local Time
	responseReceiptTimestamp	
	Type	StbM_VirtualLocalTimeType
	Comment	Ingress timestamp of Pdelay_Resp in Virtual Local Time
	requestReceiptTimestamp	
	Type	StbM_TimeStampShortType
	Comment	Ingress timestamp of Pdelay_Req in Global Time taken from the received Pdelay_Resp
	responseOriginTimestamp	
	Type	StbM_TimeStampShortType
Comment	Egress timestamp of Pdelay_Resp in Global Time taken from the received Pdelay_Resp_Follow_Up	
referenceLocalTimestamp		
Type	StbM_VirtualLocalTimeType	
Comment	Value of the Virtual Local Time of the reference Global Time Tuple	
referenceGlobalTimestamp		
Type	StbM_TimeStampShortType	





	Comment	Value of the local instance of the Global Time of the reference Global Time Tuple
	pdelay	
	Type	uint32
	Comment	Currently valid Pdelay value
Description	Structure with detailed timing data for the pDelay Initiator	
Variation	({{ecuc(EthTSyn/EthTSynGeneral/EthTSynTimeValidationSupport)}} == True)	
Available via	Rte_StbM_Type.h	

]

8.7.5.25 StbM_PdelayResponderMeasurementType

[SWS_StbM_00509] Definition of ImplementationDataType StbM_PdelayResponderMeasurementType

Upstream requirements: [RS_TS_00034](#)

[

Name	StbM_PdelayResponderMeasurementType	
Kind	Structure	
Elements	sequenceId	
	Type	uint16
	Comment	sequenceId of received Pdelay_Req frame
	requestPortId	
	Type	StbM_PortIdType
	Comment	sourcePortId of received Pdelay_Req frame
	responsePortId	
	Type	StbM_PortIdType
	Comment	sourcePortId of sent Pdelay_Resp frame
	requestReceiptTimestamp	
	Type	StbM_VirtualLocalTimeType
	Comment	Ingress timestamp of Pdelay_Req converted to Virtual Local Time
	responseOriginTimestamp	
	Type	StbM_VirtualLocalTimeType
	Comment	Egress timestamp of Pdelay_Resp converted to Virtual Local Time
	referenceLocalTimestamp	
Type	StbM_VirtualLocalTimeType	
Comment	Value of the Virtual Local Time of the reference Global Time Tuple used to convert requestReceiptTimestamp and responseOriginTimestamp into Global Time	
referenceGlobalTimestamp		



△

	Type	StbM_TimeStampShortType
	Comment	Value of the local instance of the Global Time of the reference Global Time Tuple used to convert requestReceiptTimestamp and response OriginTimestamp into Global Time
Description	Structure with detailed timing data for the pDelay Responder	
Variation	({ecuc(EthTSyn/EthTSynGeneral/EthTSynTimeValidationSupport)} == True)	
Available via	Rte_StbM_Type.h	

]

8.7.5.26 StbM_TimeSyncType

[SWS_StbM_91009] Definition of ImplementationDataType StbM_TimeSyncType

Upstream requirements: [RS_TS_20069](#)

[

Name	StbM_TimeSyncType		
Kind	Type		
Derived from	uint8		
Range	STBM_TIMESYNC_ETHERNET	0x01	Indicates Time Synchronization on Ethernet
	STBM_TIMESYNC_CAN	0x02	Indicates Time Synchronization on CAN
	STBM_TIMESYNC_FLEXRAY	0x03	Indicates Time Synchronization on Flexray
Description	Indicates the underlying Time Sync module		
Variation	-		
Available via	Rte_StbM_Type.h		

]

8.7.5.27 StbM_ProtocolParamType

[SWS_StbM_91010] Definition of ImplementationDataType StbM_ProtocolParamType

Upstream requirements: [RS_TS_20069](#)

[

Name	StbM_ProtocolParamType	
Kind	Structure	
Elements	protocolType	
	Type	StbM_TimeSyncType
	Comment	Indicates the underlying Time Sync module.
	cumulativeScaledRateOffset	
	Type	sint32
	Comment	The cumulative rate offset of the Time Master acc. to IEEE 802.1AS
	gmTimeBaseIndicator	
	Type	uint16
	Comment	The time base indicator of the current Global Time Master acc. to IEEE 802.1AS
	lastGmPhaseChange	
	Type	sint32
	Comment	The phase change of the current Global Time Master acc. to IEEE 802.1AS
	scaledLastGmFreqChange	
	Type	uint32
Comment	The scaled last frequency change of the Global Time Master acc. to IEEE 802.1AS	
Description	This structure defines TimeSync protocol specific parameters relevant for the individual TimeSync modules (only EthTSyn specific parameters are known so far)	
Variation	–	
Available via	Rte_StbM_Type.h	

]

8.7.5.28 StbM_FreshnessArrayType

[SWS_StbM_91021] Definition of ImplementationDataType StbM_FreshnessArrayType

Upstream requirements: [RS_TS_00039](#)

[

Name	StbM_FreshnessArrayType		
Kind	Array	Element type	uint8
Size	STBM_MAX_FRESHNESS_VALUE_SIZE Elements Elements		
Description	An Array that has as many elements as the max value of the Freshness Value has in bytes.		
Variation	(ecuc(StbM/StbMFreshnessValueInformation/StbMQueryFreshnessValue) == SERVICE)		
Available via	Rte_StbM_Type.h		

]

9 Sequence diagrams

The sequence diagrams in this chapter show the basic operations of the Synchronized Time-Base Manager.

Please note that the sequence diagrams are an extension for illustrational purposes to ease understanding of the specification.

9.1 StbM Initialization

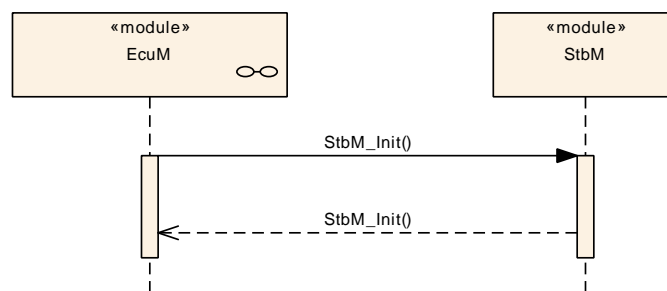


Figure 9.1: StbM Initialization

9.2 Immediate Time Synchronisation

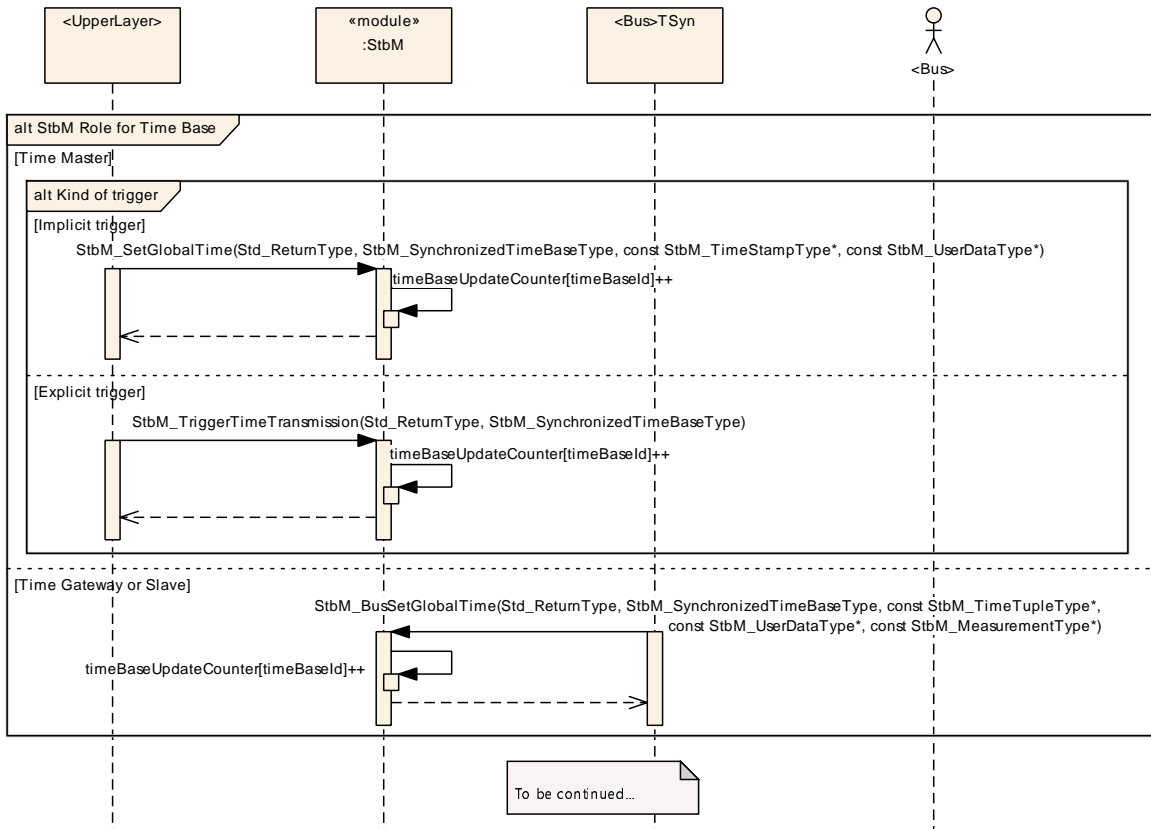


Figure 9.2: Immediate time synchronization sequence - Part 1

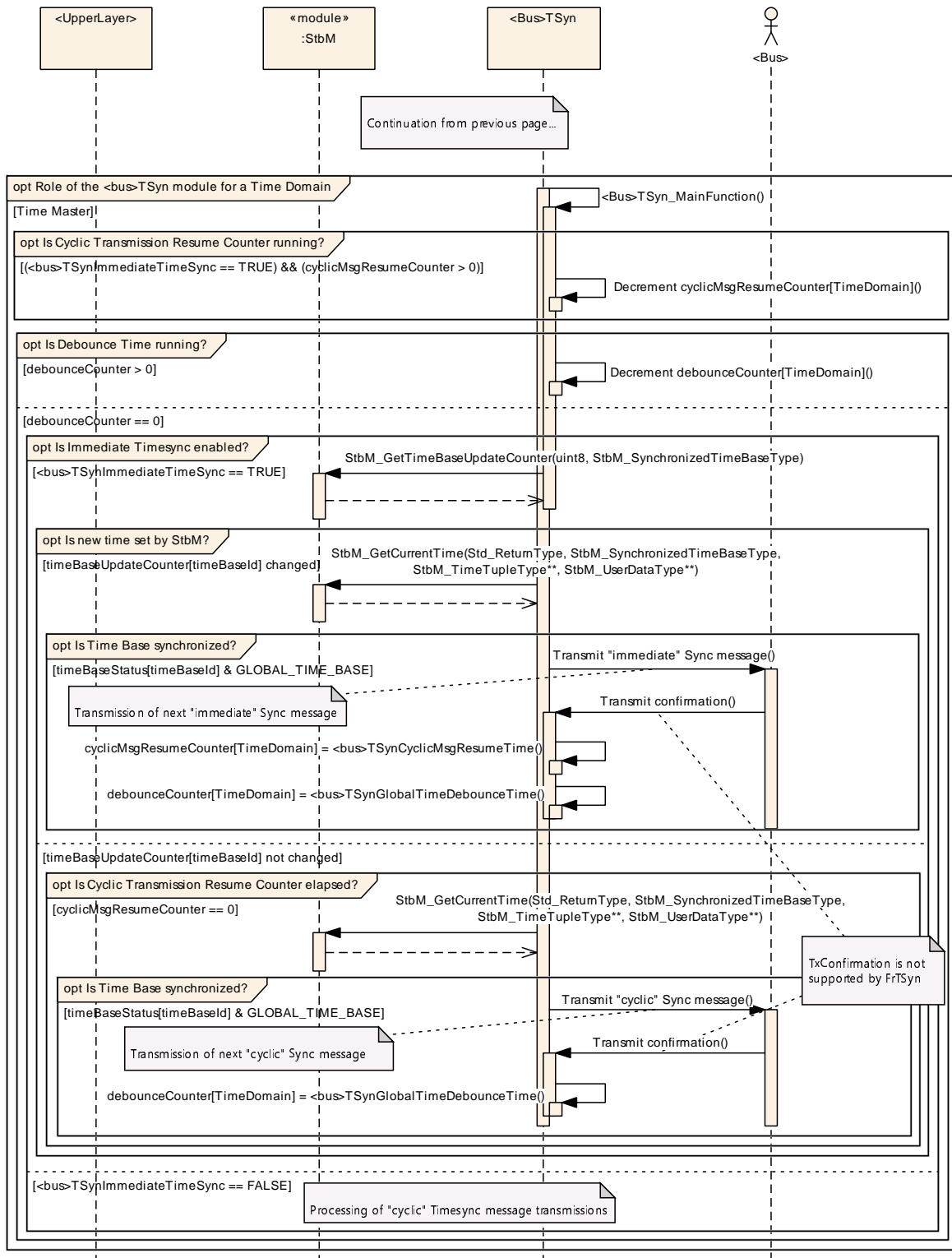


Figure 9.3: Immediate time synchronization sequence - Part 2

9.3 Explicit synchronization of OS ScheduleTable

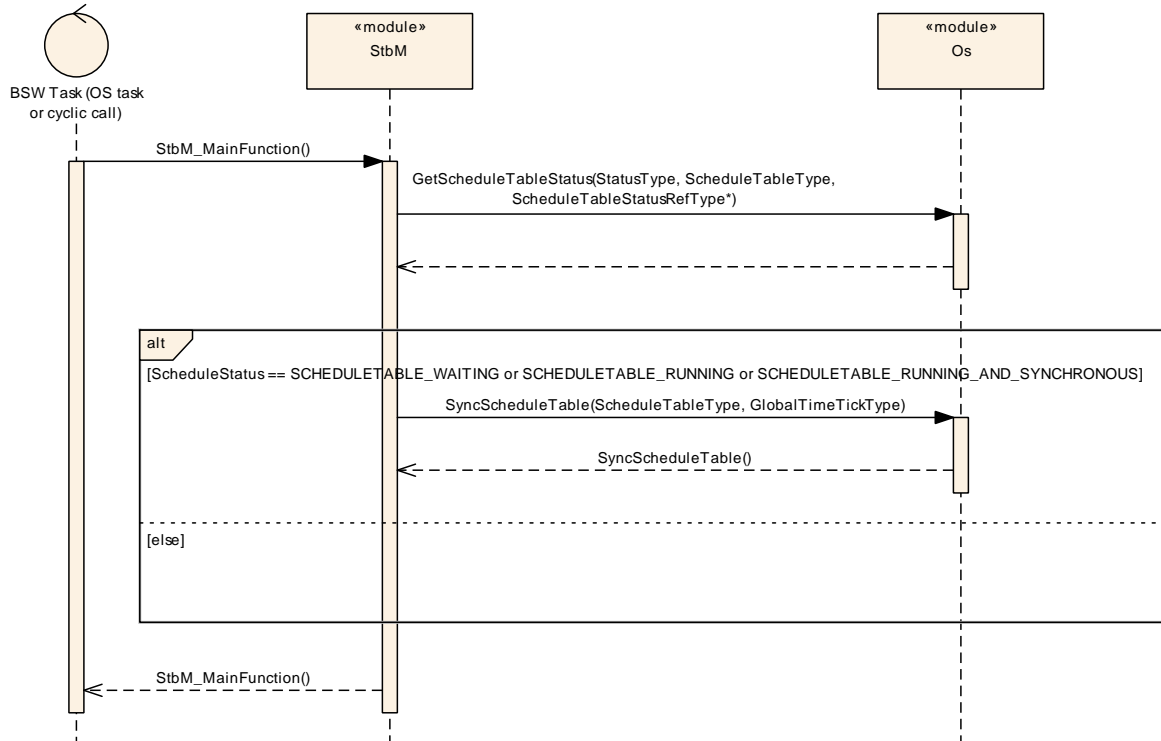


Figure 9.4: Explicit synchronization of OS Schedule Table

9.4 Rx Time Tuple Processing Sequence

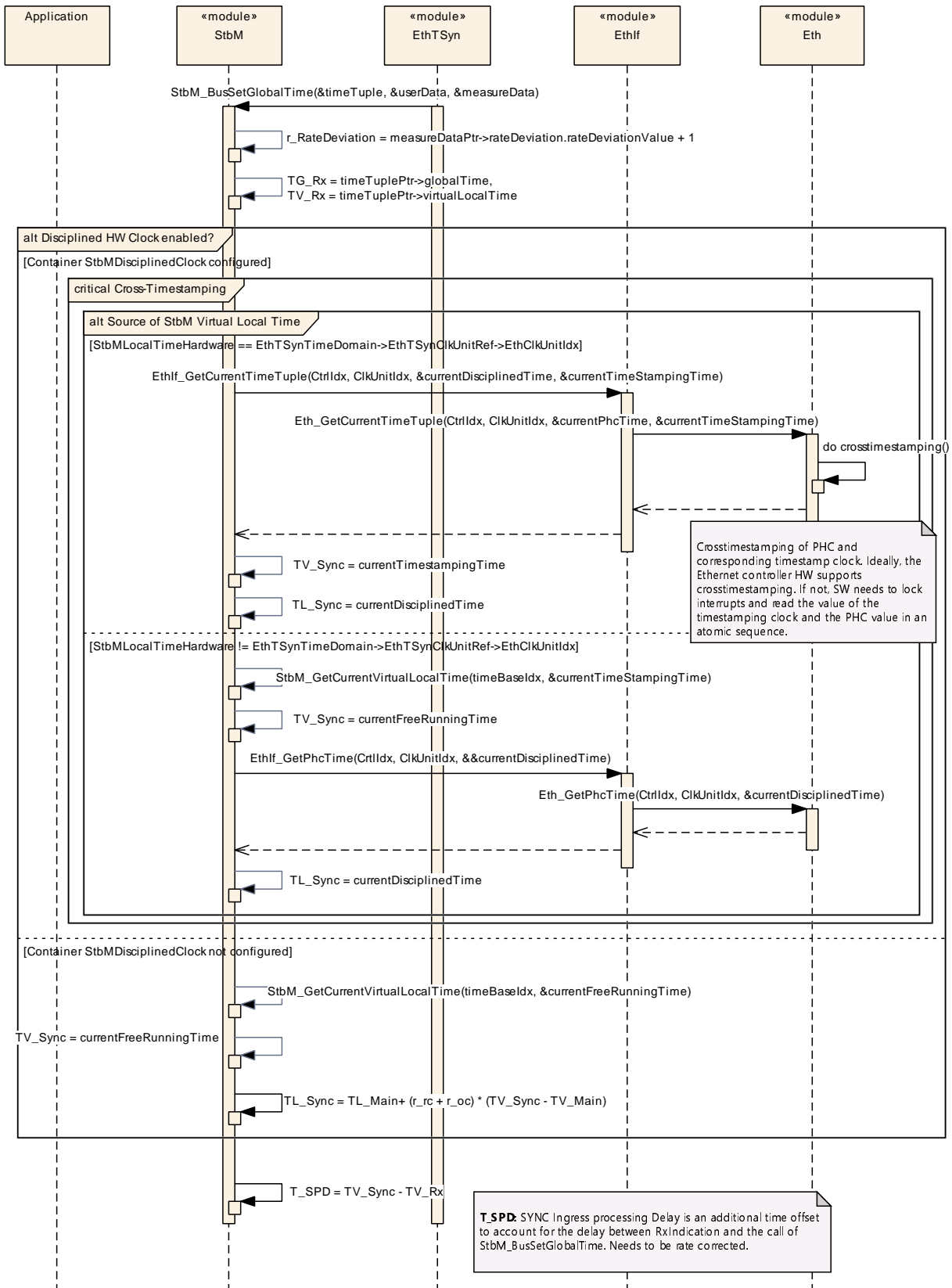


Figure 9.5: Sequence step 1: Derive SyncLocal Time Tuple

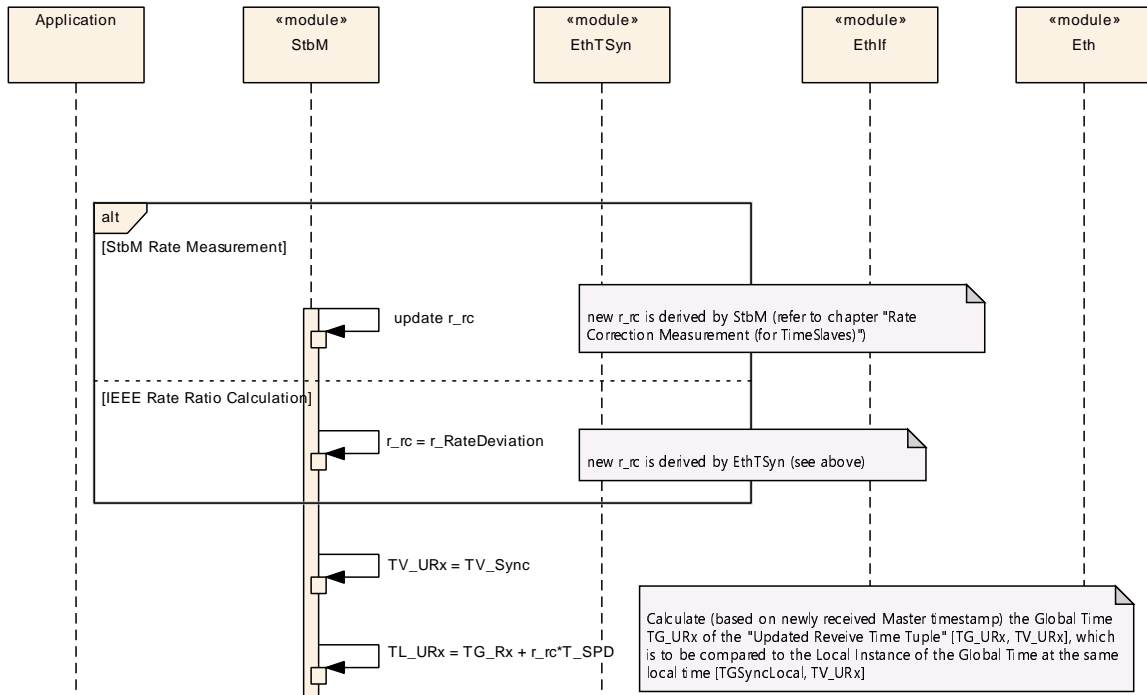


Figure 9.6: Sequence step 2: Calculate Updated Rx Time Tuple

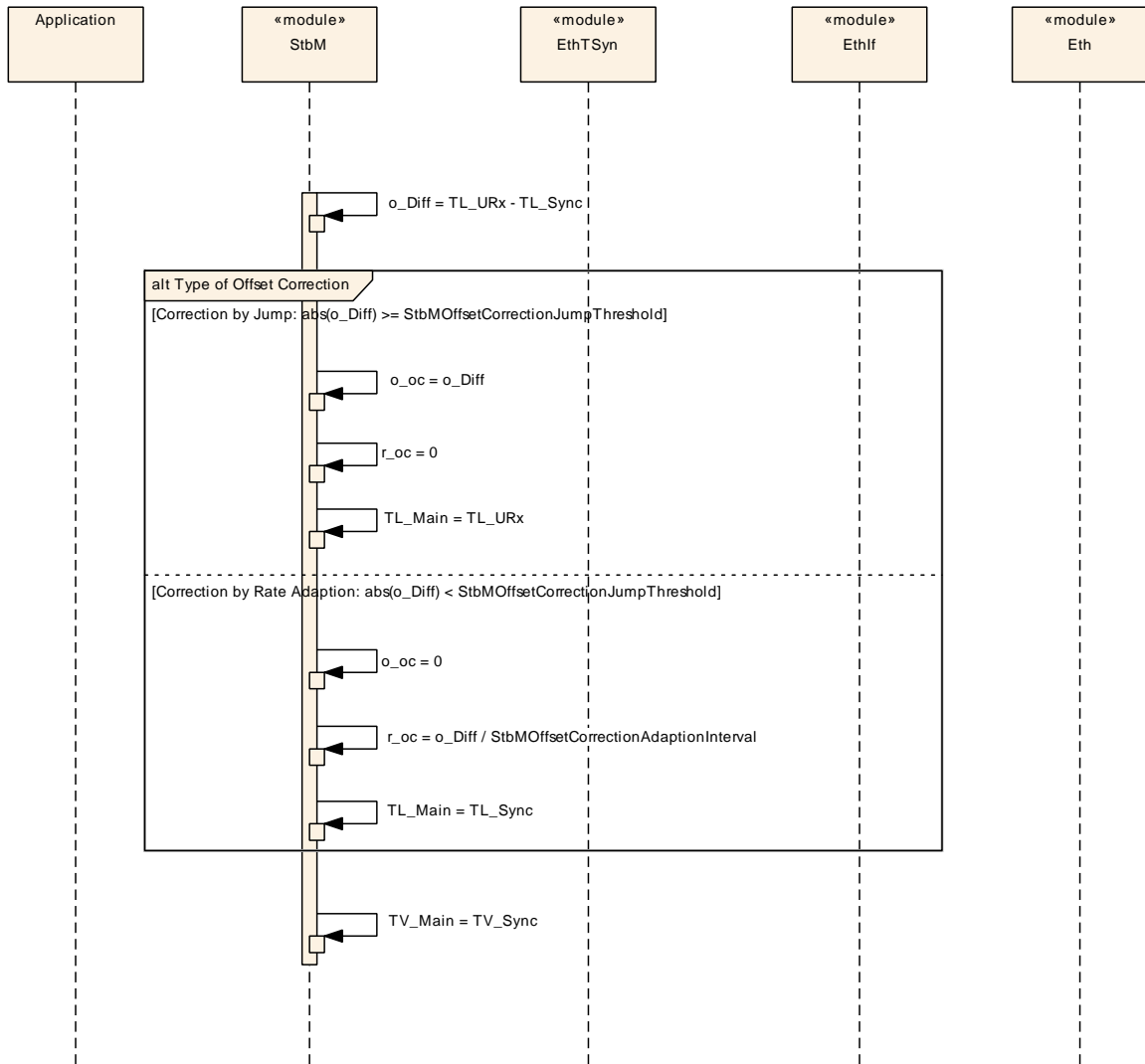


Figure 9.7: Sequence step 3: Calculate Offset Correction und update Main Time Tuple

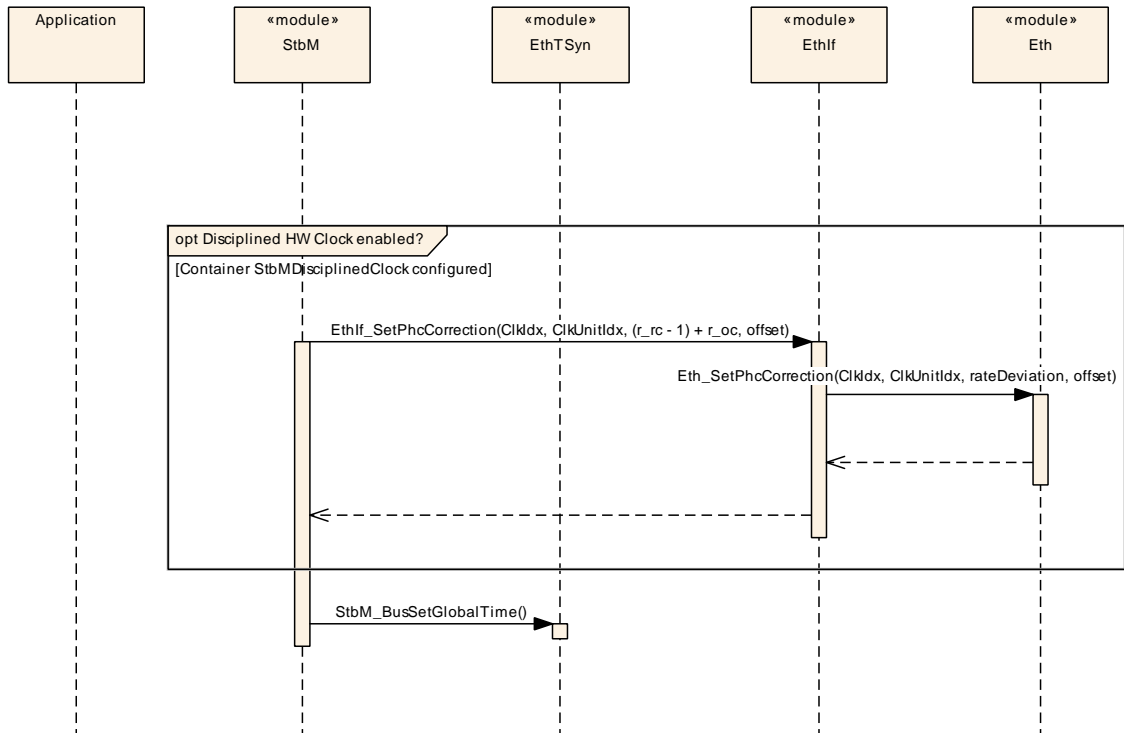


Figure 9.8: Sequence step 4 - Apply rate correction to Disciplined HW Clock

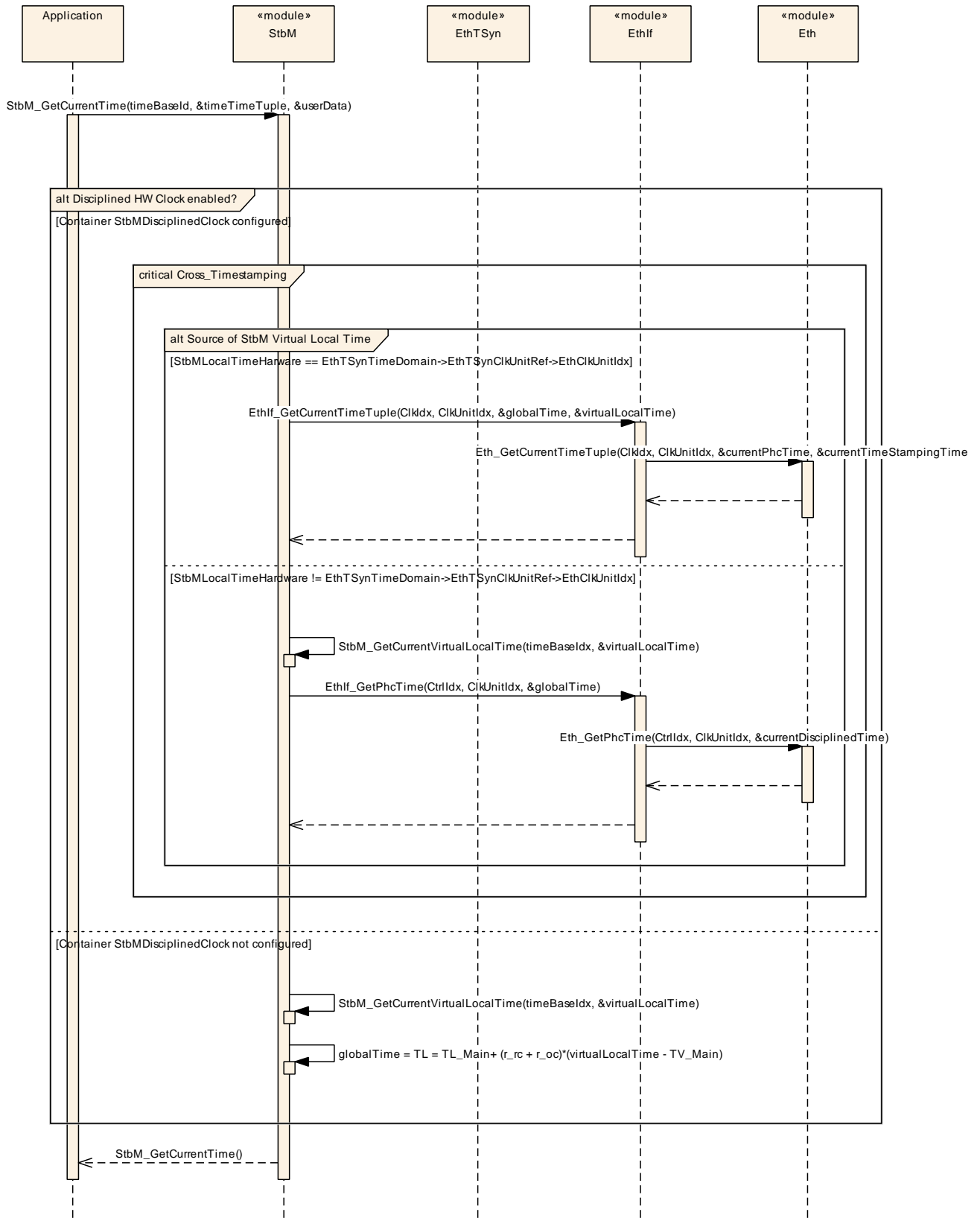


Figure 9.9: Sequence step 5 - Get Current Time Tuple by Application

10 Configuration specification

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

Chapter 10.2 “Containers and configuration parameters” specifies the structure (containers) and the parameters of the module StbM.

Chapter 10.4 “Published Information” specifies published information of the module StbM.

10.1 How to read this chapter

For details refer to the chapter 10.1 "Introduction to configuration specification" in [4, SWS BSW General].

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe chapter 7 “Functional specification” and chapter 8 “API specification”.

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

The configuration tool must check the consistency of the configuration at configuration time.

10.2.1 StbM

[ECUC_StbM_00065] Definition of EcucModuleDef StbM [

Module Name	StbM
Description	Configuration of the Synchronized Time-base Manager (StbM) module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-PRE-COMPILE

Included Containers		
Container Name	Multiplicity	Scope / Dependency
StbMFreshnessValueInformation	1	Container with the Freshness Value configurations
StbMGeneral	1	This container holds the general parameters of the Synchronized Time-base Manager
StbMSynchronizedTimeBase	1..*	Synchronized time.base collects the information about a specific time-base provider within the system.
StbMTriggeredCustomer	0..*	The triggered customer is directly triggered by the Synchronized Time-base Manager by getting synchronized with the current (global) definition of time and passage of time.

]

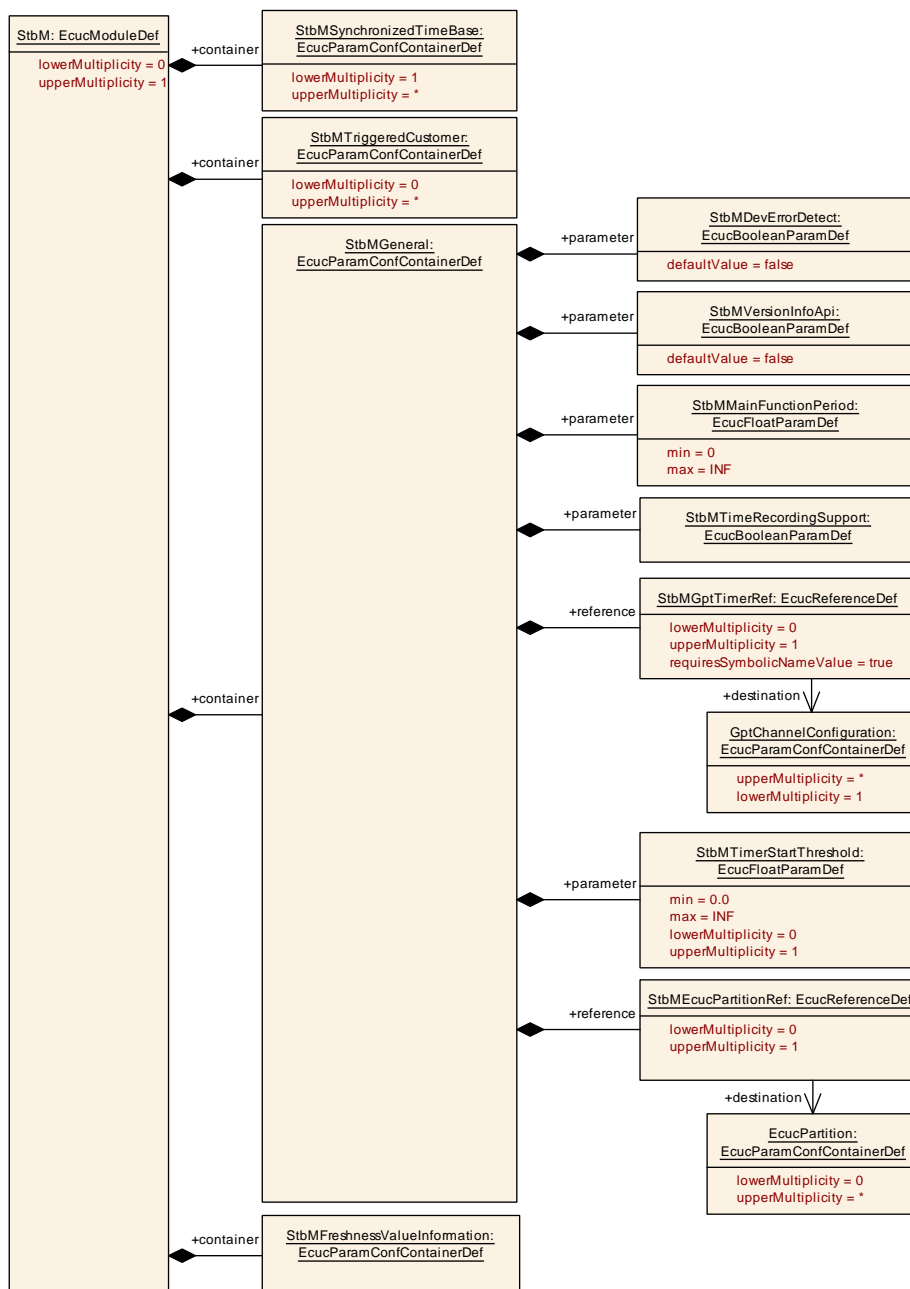


Figure 10.1

10.2.2 StbMGeneral

[ECUC_StbM_00002] Definition of EcucParamConfContainerDef StbMGeneral [

Container Name	StbMGeneral
Parent Container	StbM
Description	This container holds the general parameters of the Synchronized Time-base Manager
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
StbMDevErrorDetect	1	[ECUC_StbM_00012]
StbMMainFunctionPeriod	1	[ECUC_StbM_00027]
StbMTimeRecordingSupport	1	[ECUC_StbM_00038]
StbMTimerStartThreshold	0..1	[ECUC_StbM_00063]
StbMVersionInfoApi	1	[ECUC_StbM_00013]
StbMEcucPartitionRef	0..1	[ECUC_StbM_00069]
StbMGptTimerRef	0..1	[ECUC_StbM_00039]

No Included Containers

]

[ECUC_StbM_00012] Definition of EcucBooleanParamDef StbMDevErrorDetect [

Parameter Name	StbMDevErrorDetect		
Parent Container	StbMGeneral		
Description	Switches the development error detection and notification on or off. <ul style="list-style-type: none"> • 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		

]

[ECUC_StbM_00027] Definition of EcucFloatParamDef StbMMainFunctionPeriod

┌

Parameter Name	StbMMainFunctionPeriod		
Parent Container	StbMGeneral		
Description	Schedule period of the main function StbM_MainFunction. Unit: [s].		
Multiplicity	1		
Type	EcucFloatParamDef		
Range]0 .. INF[
Default value	-		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	-	
	Post-build time	-	
Scope / Dependency	scope: local		

└

[ECUC_StbM_00038] Definition of EcucBooleanParamDef StbMTimeRecording Support

┌

Parameter Name	StbMTimeRecordingSupport		
Parent Container	StbMGeneral		
Description	Enables/Disables the usage of the recording functionality for Synchronized Time Bases for Global Time precision measurement purpose.		
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		

└

[ECUC_StbM_00063] Definition of EcucFloatParamDef StbMTimerStartThreshold

┌

Parameter Name	StbMTimerStartThreshold		
Parent Container	StbMGeneral		
Description	This interval defines, when a GPT Timer shall be started for Time Notification Customers for which the corresponding Customer Timer is running [unit: seconds].		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range]0 .. INF[
Default value	-		



△

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		

]

[ECUC_StbM_00013] Definition of EcucBooleanParamDef StbMVersionInfoApi [

Parameter Name	StbMVersionInfoApi		
Parent Container	StbMGeneral		
Description	Activate/Deactivate the version information API (StbM_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		

]

[ECUC_StbM_00069] Definition of EcucReferenceDef StbMEcucPartitionRef [

Parameter Name	StbMEcucPartitionRef		
Parent Container	StbMGeneral		
Description	Reference to EcucPartition, where StbM module is assigned to.		
Multiplicity	0..1		
Type	Reference to EcucPartition		
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		

]

[ECUC_StbM_00039] Definition of EcucReferenceDef StbMGptTimerRef [

Parameter Name	StbMGptTimerRef		
Parent Container	StbMGeneral		
Description	This represents an optional sub-container in case any Time Notification Customer is configured. The designated GPT timer has to be configured to have a tick duration of one micro second.		
Multiplicity	0..1		
Type	Symbolic name reference to GptChannelConfiguration		
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		

]

10.2.3 StbMSynchronizedTimeBase

[ECUC_StbM_00003] Definition of EcucParamConfContainerDef StbMSynchronizedTimeBase [

Container Name	StbMSynchronizedTimeBase		
Parent Container	StbM		
Description	Synchronized time.base collects the information about a specific time-base provider within the system.		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
StbMAllowSystemWideGlobalTimeMaster	0..1	[ECUC_StbM_00066]
StbMClearTimeleapCount	0..1	[ECUC_StbM_00037]
StbMCyclicBackupInterval	0..1	[ECUC_StbM_00086]
StbMIsSystemWideGlobalTimeMaster	1	[ECUC_StbM_00036]
StbMMaxProgressionMismatchThreshold	1	[ECUC_StbM_00088]
StbMNotificationInterface	1	[ECUC_StbM_00068]





Included Parameters		
Parameter Name	Multiplicity	ECUC ID
StbMStatusNotificationCallback	0..1	[ECUC_StbM_00046]
StbMStatusNotificationMask	0..1	[ECUC_StbM_00045]
StbMStoreTimebaseNonVolatile	0..1	[ECUC_StbM_00031]
StbMSynchronizedTimeBaseIdentifier	1	[ECUC_StbM_00021]
StbMSynchronizedTimeBaseType	1	[ECUC_StbM_00100]
StbMSyncLossTimeout	0..1	[ECUC_StbM_00028]
StbMTimeLeapFutureThreshold	0..1	[ECUC_StbM_00041]
StbMTimeLeapPastThreshold	0..1	[ECUC_StbM_00042]
StbMSourceTimeBase	0..1	[ECUC_StbM_00074]

Included Containers		
Container Name	Multiplicity	Scope / Dependency
StbMDisciplinedClock	0..1	This container holds parameters relevant for the adjustable/ disciplined hardware clock of this Synchronized Time Base. Tags: atp.Status=draft
StbMFallbackTimeClock	0..1	References the hardware reference clock of this Synchronized Time Base for fallback time feature. Tags: atp.Status=draft
StbMLocalTimeClock	0..1	References the hardware reference clock of this Synchronized Time Base.
StbMNotificationCustomer	0..*	This container holds the configuration of a notification customer, which is notified is informed about the occurrence of a Time-base related event.
StbMTimeCorrection	0..1	Collects the information relevant for the rate- and offset correction of a Time Base.
StbMTimeRecording	0..1	Collects the information relevant for configuration of the precision measurement of a Time Base.
StbMTimeValidation	0..1	Container with Time Validation configuration for Time Base.

]

[ECUC_StbM_00066] Definition of EcucBooleanParamDef StbMAllowSystemWideGlobalTimeMaster [

Parameter Name	StbMAllowSystemWideGlobalTimeMaster
Parent Container	StbMSynchronizedTimeBase
Description	For postbuild variant of the StbM this parameter has to be set to true for a Global Time Master that may act as a system-wide source of time. Otherwise no corresponding service ports/interfaces is provided. The Global Time Master functionality behind the service ports/interfaces has to be enabled/disabled separately via parameter StbMIsSystemWideGlobalTimeMaster.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	–
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		

]

[ECUC_StbM_00037] Definition of EcucIntegerParamDef StbMClearTimeleap Count

Parameter Name	StbMClearTimeleapCount		
Parent Container	StbMSynchronizedTimeBase		
Description	This attribute describes the required number of updates to the Time Base where the time difference to the previous value has to remain below StbMTimeLeapPastThreshold/StbMTimeLeapFutureThreshold until the TIMELEAP_PAST/TIMELEAP_FUTURE bit within timeBaseStatus of the Time Base is cleared.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default value	1		
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		

]

[ECUC_StbM_00086] Definition of EcucFloatParamDef StbMCyclicBackupInterval

Parameter Name	StbMCyclicBackupInterval		
Parent Container	StbMSynchronizedTimeBase		
Description	Time interval to calculate the "backup" time to be stored in NvM [unit: seconds].		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range]0 .. 65535]		
Default value	–		
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		

]

[ECUC_StbM_00036] Definition of EcucBooleanParamDef StbMIsSystemWideGlobalTimeMaster [

Parameter Name	StbMIsSystemWideGlobalTimeMaster		
Parent Container	StbMSynchronizedTimeBase		
Description	<p>This parameter shall be set to true for a Global Time Master that acts as a system-wide source of time information with respect to Global Time.</p> <p>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 own several Global Time Domains on different buses it is connected to.</p>		
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		

]

[ECUC_StbM_00088] Definition of EcucFloatParamDef StbMMaxProgressionMismatchThreshold

Status: DRAFT

[

Parameter Name	StbMMaxProgressionMismatchThreshold		
Parent Container	StbMSynchronizedTimeBase		
Description	<p>This represents the maximum allowed difference between local time and fallback time of the time base [unit: seconds].</p> <p>Tags: atp.Status=draft</p>		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF[





Default value	-		
Post-Build Variant Multiplicity	true		
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		

]

[ECUC_StbM_00068] Definition of EcucEnumerationParamDef StbMNotification Interface [

Parameter Name	StbMNotificationInterface		
Parent Container	StbMSynchronizedTimeBase		
Description	The parameter defines what type of interface shall be used to notify a customer of a status event.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CALLBACK	-	
	CALLBACK_AND_SR_INTERFACE	-	
	NO_NOTIFICATION	-	
	SR_INTERFACE	-	
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		

]

[ECUC_StbM_00046] Definition of EcucFunctionNameDef StbMStatusNotificationCallback [

Parameter Name	StbMStatusNotificationCallback		
Parent Container	StbMSynchronizedTimeBase		
Description	Name of the customer specific status notification callback function, which shall be called, if a non-masked status event occurs.		



△

Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	–		
Regular Expression	–		
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 dependency: StbMStatusNotificationCallback shall be available, if and only if StbMNotificationInterface is set to either CALLBACK or CALLBACK_AND_SR_INTERFACE.		

]

[ECUC_StbM_00045] Definition of EcucIntegerParamDef StbMStatusNotification Mask [

Parameter Name	StbMStatusNotificationMask		
Parent Container	StbMSynchronizedTimeBase		
Description	The parameter defines the initial value for NotificationMask mask, which defines the events for which the event notification callback function shall be called.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
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		

]

[ECUC_StbM_00031] Definition of EcucEnumerationParamDef StbMStoreTime-baseNonVolatile [

Parameter Name	StbMStoreTimebaseNonVolatile		
Parent Container	StbMSynchronizedTimeBase		
Description	This allows for specifying that the Time Base shall be stored in the NvRam.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	NO_STORAGE	–	
	STORAGE	–	
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		

]

[ECUC_StbM_00021] Definition of EcucIntegerParamDef StbMSynchronizedTimeBaseIdentifier [

Parameter Name	StbMSynchronizedTimeBaseIdentifier		
Parent Container	StbMSynchronizedTimeBase		
Description	Identification of a Synchronized Time Base via a unique identifier. Range: <ul style="list-style-type: none"> • 0 .. 127: Synchronized Time Bases and Pure Local Time Bases • 128 .. 65535: Reserved 		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

]

[ECUC_StbM_00100] Definition of EcucEnumerationParamDef StbMSynchronizedTimeBaseType

Status: DRAFT

[

Parameter Name	StbMSynchronizedTimeBaseType		
Parent Container	StbMSynchronizedTimeBase		
Description	Definition of the type of a Time Base. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	TBTYPE_PURELOCAL	Pure Local Time Base. Tags: atp.Status=draft	
	TBTYPE_SYNCHRONIZED	Synchronized Time Base. Tags: atp.Status=draft	
Default value	TBTYPE_SYNCHRONIZED		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

]

[ECUC_StbM_00028] Definition of EcucFloatParamDef StbMSyncLossTimeout [

Parameter Name	StbMSyncLossTimeout		
Parent Container	StbMSynchronizedTimeBase		
Description	This attribute describes the timeout for the situation that the time synchronization gets lost in the scope of the time domain. Unit: seconds		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range]0 .. INF[
Default value	–		
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		

]

[ECUC_StbM_00041] Definition of EcucFloatParamDef StbMTimeLeapFuture Threshold [

Parameter Name	StbMTimeLeapFutureThreshold		
Parent Container	StbMSynchronizedTimeBase		
Description	This represents the maximum allowed positive difference between a newly received Global Time Base value and the current Local Time Base value [unit: seconds].		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF[
Default value	-		
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		

]

[ECUC_StbM_00042] Definition of EcucFloatParamDef StbMTimeLeapPast Threshold [

Parameter Name	StbMTimeLeapPastThreshold		
Parent Container	StbMSynchronizedTimeBase		
Description	This represents the maximum allowed negative difference between the current Local Time Base value and a newly received Global Time Base value [unit: seconds].		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF[
Default value	-		
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		

]

[ECUC_StbM_00074] Definition of EcucReferenceDef StbMSourceTimeBase [

Parameter Name	StbMSourceTimeBase		
Parent Container	StbMSynchronizedTimeBase		
Description	This is a reference to a Time Base, which the current Time Base is cloned from. This makes the referenced Time Base the source Time Base for cloning and the current Time the destination Time Base for cloning.		
Multiplicity	0..1		
Type	Reference to StbMSynchronizedTimeBase		
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		

]

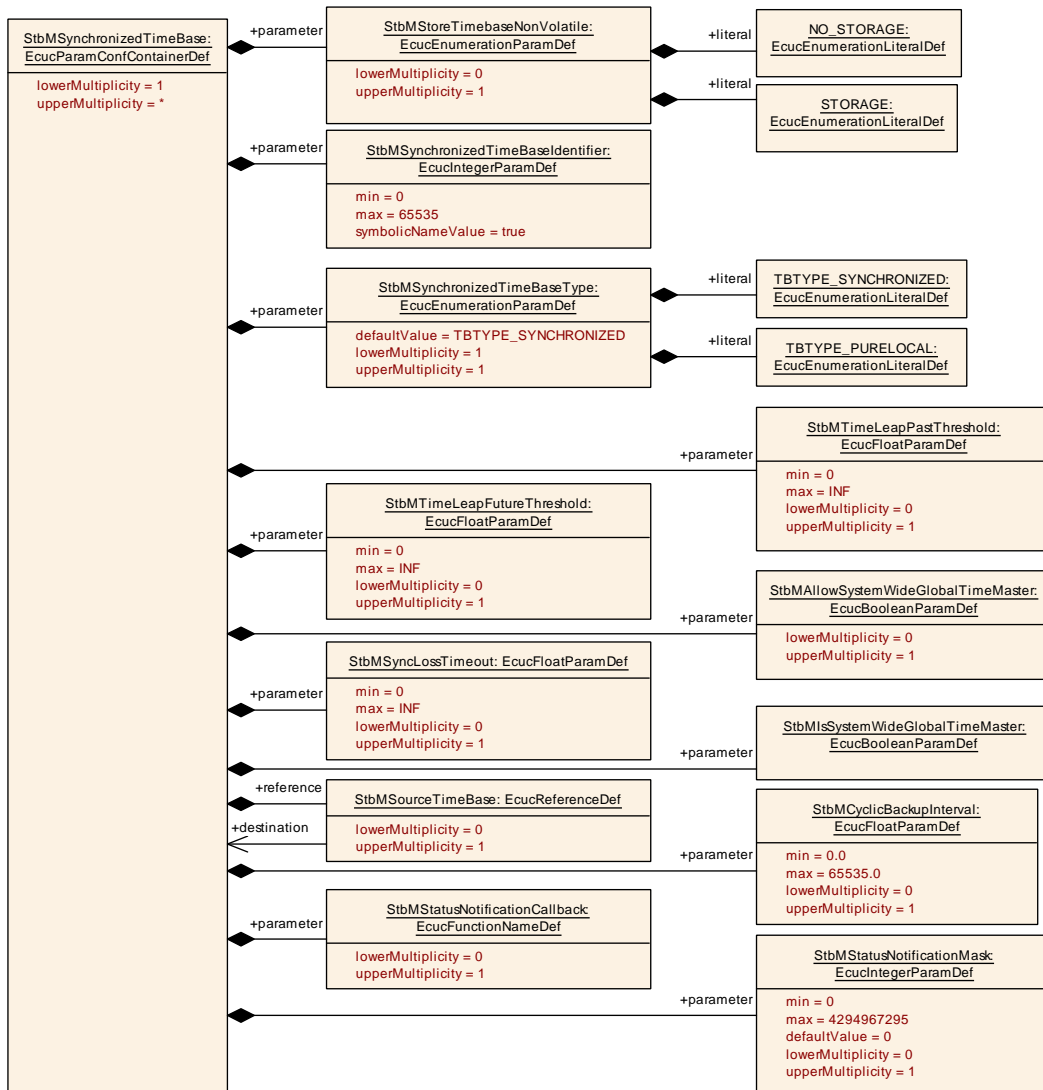


Figure 10.2

10.2.4 StbMTimeCorrection

[ECUC_StbM_00048] Definition of EcucParamConfContainerDef StbMTimeCorrection

Container Name	StbMTimeCorrection		
Parent Container	StbMSynchronizedTimeBase		
Description	Collects the information relevant for the rate- and offset correction of a Time Base.		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	-	





	Post-build time	-	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
StbMAllowMasterRateCorrection	0..1	[ECUC_StbM_00043]
StbMMasterRateDeviationMax	0..1	[ECUC_StbM_00044]
StbMOffsetCorrectionAdaptionInterval	0..1	[ECUC_StbM_00057]
StbMOffsetCorrectionJumpThreshold	0..1	[ECUC_StbM_00056]
StbMRateCorrectionMeasurementDuration	0..1	[ECUC_StbM_00054]
StbMRateCorrectionsPerMeasurementDuration	0..1	[ECUC_StbM_00055]
StbMRateCorrectionThreshold	0..1	[ECUC_StbM_00099]
StbMRateSource	0..1	[ECUC_StbM_00101]

No Included Containers

]

[[ECUC_StbM_00043](#)] Definition of EcucBooleanParamDef [StbMAllowMasterRateCorrection](#)

Parameter Name	StbMAllowMasterRateCorrection		
Parent Container	StbMTimeCorrection		
Description	<p>This attribute describes whether the rate correction value of a Time Base can be set by StbM_SetRateCorrection():</p> <ul style="list-style-type: none"> • false: the rate correction value can not be set by StbM_SetRateCorrection() • true: the rate correction value can be set by StbM_SetRateCorrection() 		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
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		

]

[ECUC_StbM_00044] Definition of EcucIntegerParamDef StbMMasterRateDeviationMax [

Parameter Name	StbMMasterRateDeviationMax		
Parent Container	StbMTimeCorrection		
Description	This attribute describes the maximum allowed absolute value of the rate deviation value to be set by StbM_SetRateCorrection() [unit: ppm].		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 32000		
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		

]

[ECUC_StbM_00057] Definition of EcucFloatParamDef StbMOffsetCorrectionAdaptionInterval [

Parameter Name	StbMOffsetCorrectionAdaptionInterval		
Parent Container	StbMTimeCorrection		
Description	Defines the interval during which the adaptive rate correction cancels out the rate- and time deviation [unit: seconds].		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range]0 .. INF[
Default value	–		
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		

]

[ECUC_StbM_00056] Definition of EcucFloatParamDef StbMOffsetCorrection JumpThreshold [

Parameter Name	StbMOffsetCorrectionJumpThreshold		
Parent Container	StbMTimeCorrection		
Description	Threshold for the correction method. Deviations below this value will be corrected by a linear reduction over a defined timespan. Values equal- and greater than this value will be corrected by immediately setting the correct time- and rate in form of a jump [unit: seconds].		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF[
Default value	-		
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		

]

[ECUC_StbM_00054] Definition of EcucFloatParamDef StbMRateCorrectionMeasurementDuration [

Parameter Name	StbMRateCorrectionMeasurementDuration		
Parent Container	StbMTimeCorrection		
Description	Definition of the time span [s] which is used to calculate the rate deviation.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF[
Default value	1		
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		

]

[ECUC_StbM_00055] Definition of EcucIntegerParamDef StbMRateCorrections PerMeasurementDuration [

Parameter Name	StbMRateCorrectionsPerMeasurementDuration		
Parent Container	StbMTimeCorrection		
Description	Number of simultaneous rate measurements to determine the current rate deviation.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default value	1		
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		

]

[ECUC_StbM_00099] Definition of EcucIntegerParamDef StbMRateCorrection Threshold [

Parameter Name	StbMRateCorrectionThreshold		
Parent Container	StbMTimeCorrection		
Description	Threshold for rate correction calculation to determine whether the absolute value of the calculated rate deviation exceeds the accepted range. A value of 0 deactivates the check.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 32000		
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		

]

[ECUC_StbM_00101] Definition of EcucEnumerationParamDef StbMRateSource

Status: DRAFT

[

Parameter Name	StbMRateSource		
Parent Container	StbMTimeCorrection		
Description	Selects which module does the rate measurement. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	RATE_SOURCE_STBM	the StbM itself does a rate measurement	
	RATE_SOURCE_TIMESYNC_MODULE	the Timesync module does the rate measurement and provides the rate value to the StbM	
Post-Build Variant Multiplicity	true		
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: ECU		

]

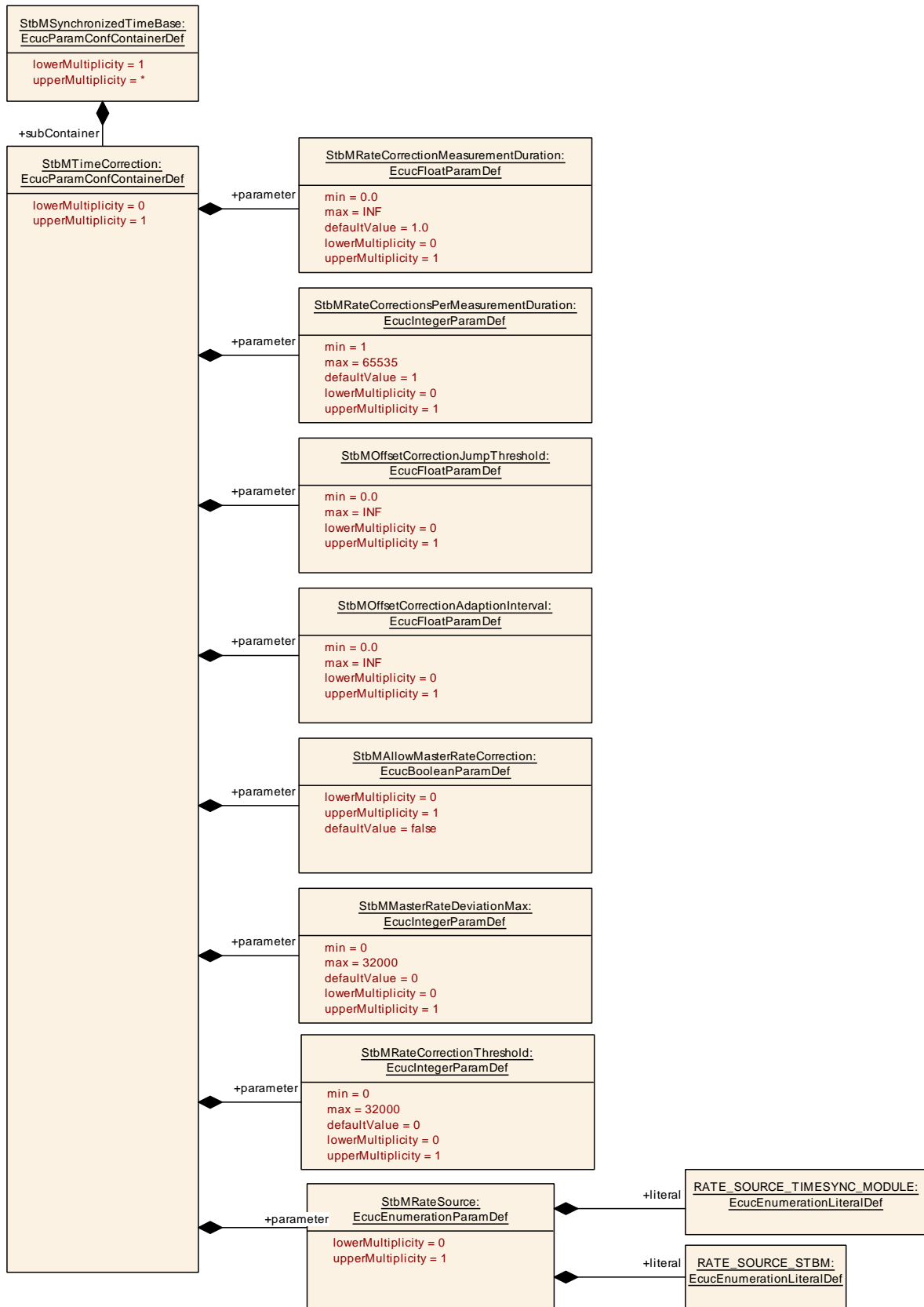


Figure 10.3

10.2.5 StbMLocalTimeClock

[ECUC_StbM_00047] Definition of EcucParamConfContainerDef StbMLocalTimeClock

Container Name	StbMLocalTimeClock		
Parent Container	StbMSynchronizedTimeBase		
Description	References the hardware reference clock of this Synchronized Time Base.		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
StbMClockFrequency	0..1	[ECUC_StbM_00051]
StbMClockPrescaler	0..1	[ECUC_StbM_00052]
StbMLocalTimeHardware	1	[ECUC_StbM_00053]

No Included Containers

]

[ECUC_StbM_00051] Definition of EcucIntegerParamDef StbMClockFrequency

Parameter Name	StbMClockFrequency		
Parent Container	StbMLocalTimeClock		
Description	Represents the frequency [Hz] of the HW reference clock used by the StbM.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
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		

]

[ECUC_StbM_00052] Definition of EcucIntegerParamDef StbMClockPrescaler [

Parameter Name	StbMClockPrescaler		
Parent Container	StbMLocalTimeClock		
Description	Represents the prescaler to calculate the resulting frequency of the HW reference clock used by the StbM.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
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		

]

[ECUC_StbM_00053] Definition of EcucChoiceReferenceDef StbMLocalTime Hardware [

Parameter Name	StbMLocalTimeHardware		
Parent Container	StbMLocalTimeClock		
Description	Reference to the local time hardware.		
Multiplicity	1		
Type	Choice reference to [CanTSynGlobalTimeDomain, EthTSynGlobalTimeDomain, Gpt ChannelConfiguration, OsCounter]		
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		

]

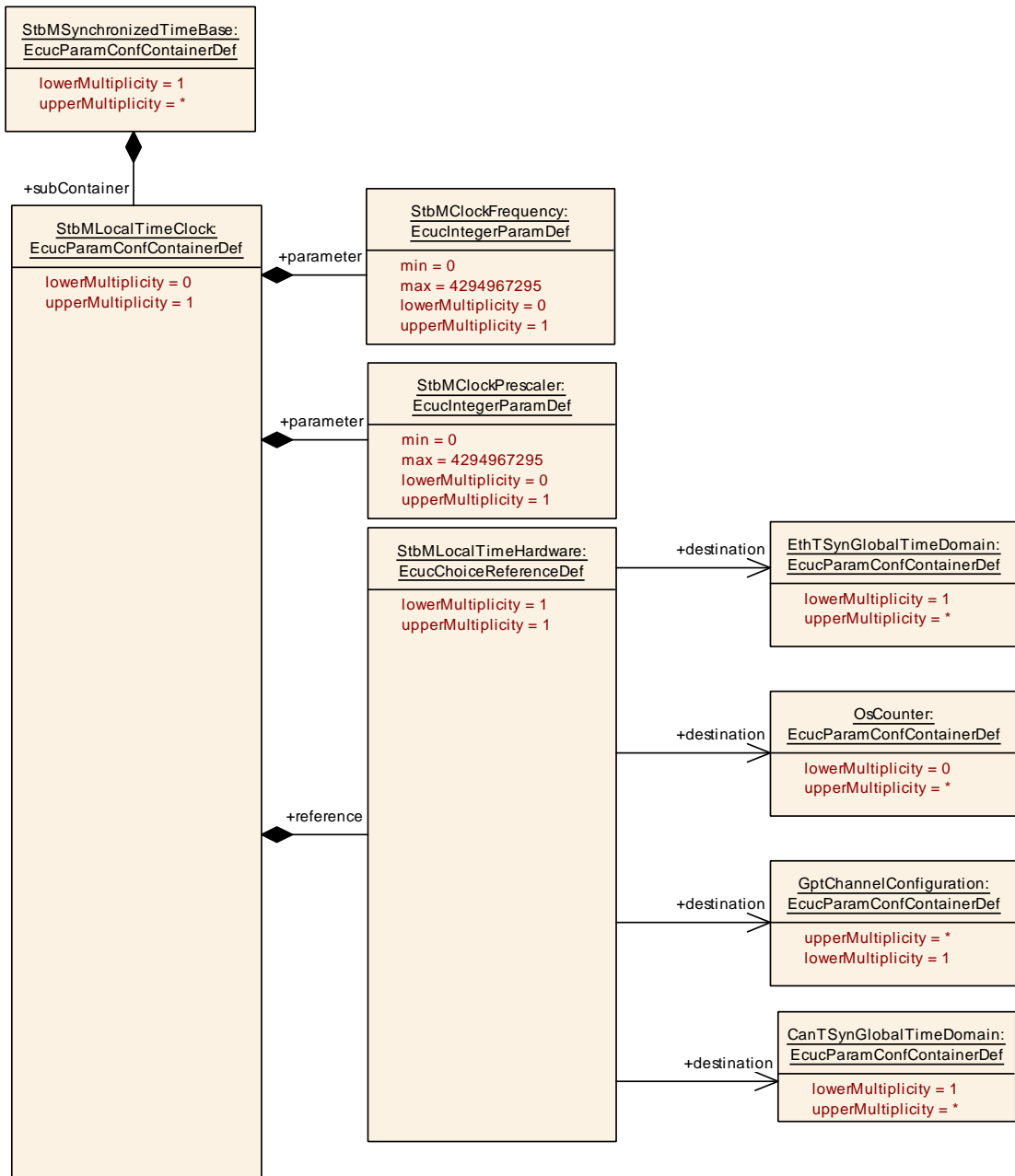


Figure 10.4

10.2.6 StbMDisciplinedClock

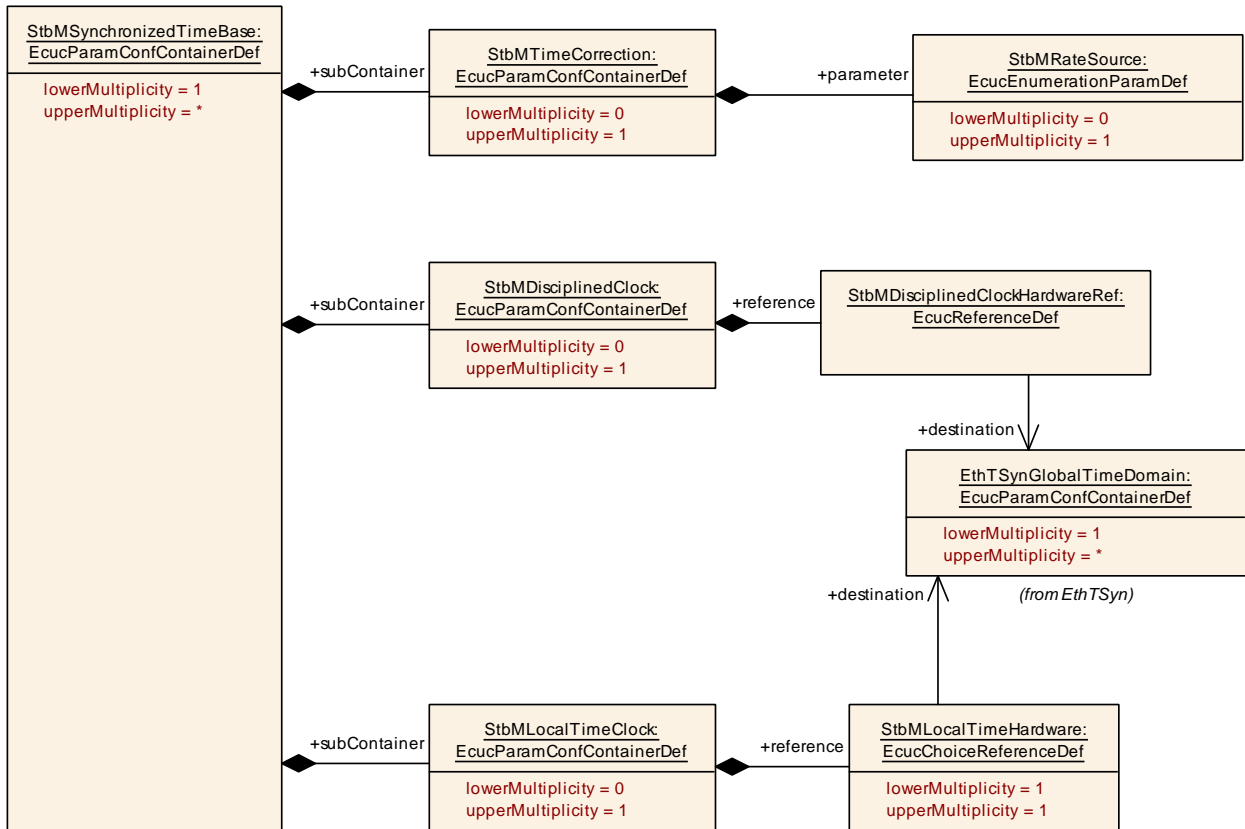


Figure 10.5

[ECUC_StbM_00103] Definition of EcucParamConfContainerDef StbMDisciplined Clock

Status: DRAFT

[

Container Name	StbMDisciplinedClock		
Parent Container	StbMSynchronizedTimeBase		
Description	This container holds parameters relevant for the adjustable/disciplined hardware clock of this Synchronized Time Base. Tags: atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
StbMDisciplinedClockHardwareRef	1	[ECUC_StbM_00102]

No Included Containers

]

[ECUC_StbM_00102] Definition of EcucReferenceDef StbMDisciplinedClock HardwareRef

Status: DRAFT

[

Parameter Name	StbMDisciplinedClockHardwareRef		
Parent Container	StbMDisciplinedClock		
Description	Reference to the adjustable/disciplined hardware clock. Tags: atp.Status=draft		
Multiplicity	1		
Type	Reference to EthTSynGlobalTimeDomain		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

]

10.2.7 StbMFallbackTimeClock

[ECUC_StbM_00089] Definition of EcucParamConfContainerDef StbMFallbackTimeClock

Status: DRAFT

[

Container Name	StbMFallbackTimeClock		
Parent Container	StbMSynchronizedTimeBase		
Description	References the hardware reference clock of this Synchronized Time Base for fallback time feature. Tags: atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
StbMFallbackTimeClockFrequency	0..1	[ECUC_StbM_00091]
StbMFallbackTimeClockPrescaler	0..1	[ECUC_StbM_00092]
StbMFallbackTimeHardware	1	[ECUC_StbM_00093]

No Included Containers

]

[[ECUC_StbM_00091](#)] Definition of EcucIntegerParamDef [StbMFallbackTimeClockFrequency](#)

Status: DRAFT

[

Parameter Name	StbMFallbackTimeClockFrequency		
Parent Container	StbMFallbackTimeClock		
Description	Represents the frequency [Hz] of the HW reference clock used by the StbM. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	-		
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		

]

[[ECUC_StbM_00092](#)] Definition of EcucIntegerParamDef [StbMFallbackTimeClockPrescaler](#)

Status: DRAFT

[

Parameter Name	StbMFallbackTimeClockPrescaler		
Parent Container	StbMFallbackTimeClock		
Description	Represents the prescaler to calculate the resulting frequency of the HW reference clock used by the StbM. Tags: atp.Status=draft		



△

Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	-		
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		

]

[ECUC_StbM_00093] Definition of EcucChoiceReferenceDef StbMFallbackTime Hardware

Status: DRAFT

[

Parameter Name	StbMFallbackTimeHardware		
Parent Container	StbMFallbackTimeClock		
Description	Reference to the local time hardware. Tags: atp.Status=draft		
Multiplicity	1		
Type	Choice reference to [CanTSynGlobalTimeDomain, EthTSynGlobalTimeDomain, Gpt ChannelConfiguration, OsCounter]		
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		

]

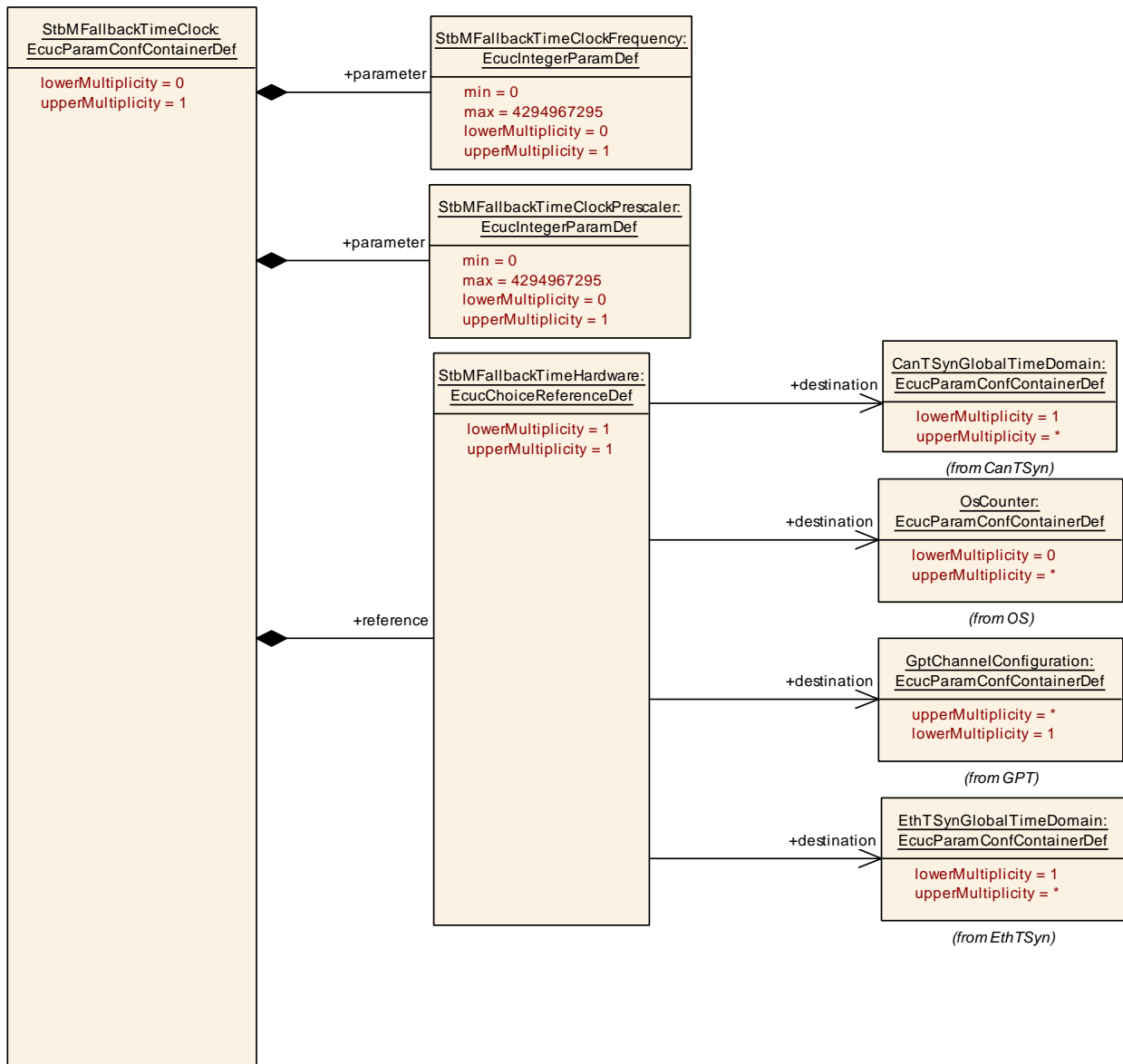


Figure 10.6

10.2.8 StbMTimeRecording

[ECUC_StbM_00049] Definition of EcucParamConfContainerDef StbMTimeRecording

Container Name	StbMTimeRecording
Parent Container	StbMSynchronizedTimeBase
Description	Collects the information relevant for configuration of the precision measurement of a Time Base.





Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
StbMSyncTimeRecordBlockCallback	0..1	[ECUC_StbM_00060]
StbMSyncTimeRecordTableBlockCount	1	[ECUC_StbM_00058]

No Included Containers

]

[ECUC_StbM_00060] Definition of EcucFunctionNameDef StbMSyncTimeRecordBlockCallback [

Parameter Name	StbMSyncTimeRecordBlockCallback		
Parent Container	StbMTimeRecording		
Description	Name of the customer specific callback function, which shall be called, if a measurement data for a Synchronized Time Base are available.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	–		
Regular Expression	–		
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		

]

[ECUC_StbM_00058] Definition of EcucIntegerParamDef StbMSyncTimeRecordTableBlockCount [

Parameter Name	StbMSyncTimeRecordTableBlockCount		
Parent Container	StbMTimeRecording		
Description	Represents the number of Blocks used for queuing time measurement events for the Synchronized Time Base Record Table.		





Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	-		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	-	
	Post-build time	-	
Scope / Dependency	scope: local		

]

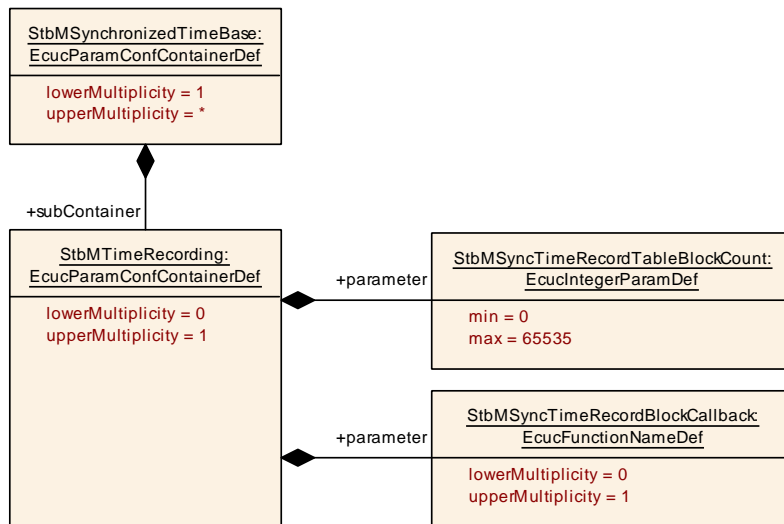


Figure 10.7

10.2.9 StbMTimeValidation

[ECUC_StbM_00072] Definition of EcucParamConfContainerDef StbMTimeValidation [

Container Name	StbMTimeValidation		
Parent Container	StbMSynchronizedTimeBase		
Description	Container with Time Validation configuration for Time Base.		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	-	
	Post-build time	-	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
StbMPdelayValidationThreshold	0..1	[ECUC_StbM_00097]
StbMRateJitterThreshold	0..1	[ECUC_StbM_00095]
StbMRateWanderIntervalWindow	0..1	[ECUC_StbM_00098]
StbMRateWanderThreshold	0..1	[ECUC_StbM_00096]
StbMTimeValidationRecordTableBlockCount	1	[ECUC_StbM_00073]

No Included Containers

]

[ECUC_StbM_00097] Definition of EcucFloatParamDef StbMPdelayValidation Threshold [

Parameter Name	StbMPdelayValidationThreshold		
Parent Container	StbMTimeValidation		
Description	This represents the maximum allowed threshold of the Pdelay value of the time base [unit: seconds].		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF[
Default value	-		
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		

]

[ECUC_StbM_00095] Definition of EcucFloatParamDef StbMRateJitterThreshold [

Parameter Name	StbMRateJitterThreshold		
Parent Container	StbMTimeValidation		
Description	This represents the maximum allowed threshold for rate jitter.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF[
Default value	-		
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		

]

[ECUC_StbM_00098] Definition of EcucIntegerParamDef StbMRateWanderIntervalWindow [

Parameter Name	StbMRateWanderIntervalWindow		
Parent Container	StbMTimeValidation		
Description	This represents the window to take sample for calculation of rate wander.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	–		
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		

]

[ECUC_StbM_00096] Definition of EcucFloatParamDef StbMRateWanderThreshold [

Parameter Name	StbMRateWanderThreshold		
Parent Container	StbMTimeValidation		
Description	This represents the maximum allowed threshold for rate wander.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF[
Default value	–		
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		

]

[ECUC_StbM_00073] Definition of EcucIntegerParamDef StbMTimeValidation RecordTableBlockCount [

Parameter Name	StbMTimeValidationRecordTableBlockCount		
Parent Container	StbMTimeValidation		
Description	Size of record table for Time Validation (number of blocks).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

]

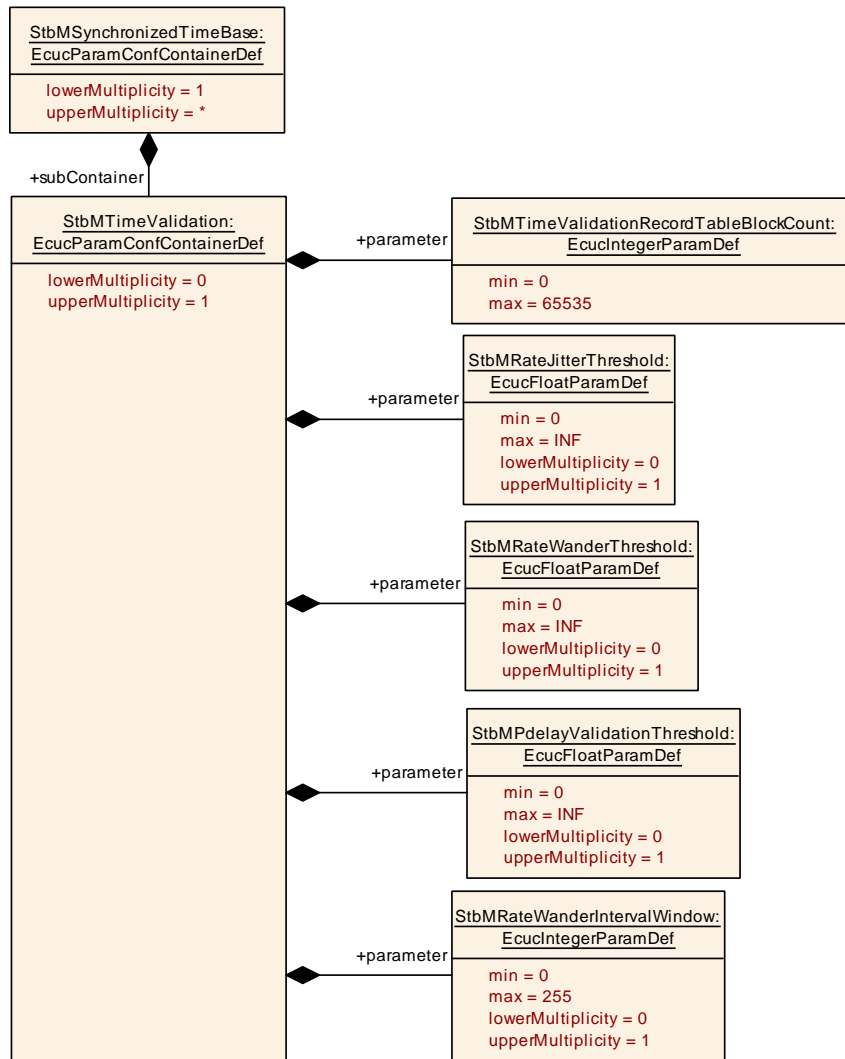


Figure 10.8

10.2.10 StbMNotificationCustomer

[ECUC_StbM_00050] Definition of EcucParamConfContainerDef StbMNotificationCustomer [

Container Name	StbMNotificationCustomer		
Parent Container	StbMSynchronizedTimeBase		
Description	This container holds the configuration of a notification customer, which is notified is informed about the occurrence of a Time-base related event.		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	-	
	Post-build time	-	





Configuration Parameters

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
StbMNotificationCustomerId	1	[ECUC_StbM_00062]
StbMTimeNotificationCallback	0..1	[ECUC_StbM_00064]

No Included Containers

]

[ECUC_StbM_00062] Definition of EcucIntegerParamDef StbMNotificationCustomerId [

Parameter Name	StbMNotificationCustomerId		
Parent Container	StbMNotificationCustomer		
Description	Identification of a event notification customer.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	-		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	-	
	Post-build time	-	
Scope / Dependency	scope: local		

]

[ECUC_StbM_00064] Definition of EcucFunctionNameDef StbMTimeNotificationCallback [

Parameter Name	StbMTimeNotificationCallback		
Parent Container	StbMNotificationCustomer		
Description	Name of the customer specific notification callback function, which shall be called, if the time previously set by the customer is reached.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	-		
Regular Expression	-		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	-	
	Post-build time	-	
Scope / Dependency	scope: local		

]

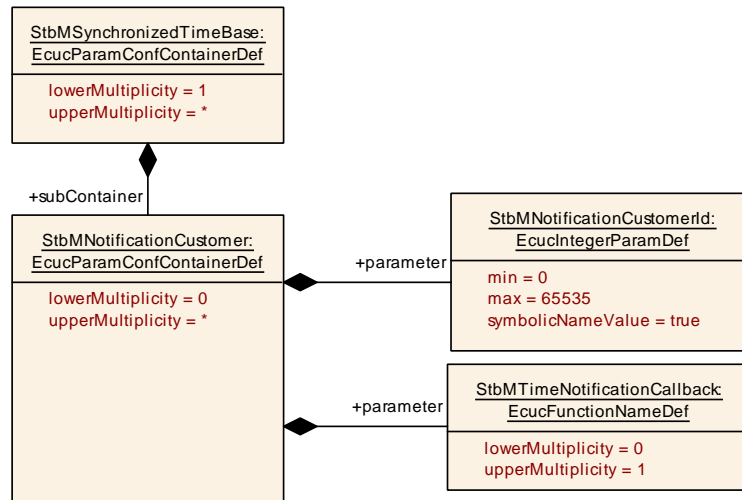


Figure 10.9

10.2.11 StbMTriggeredCustomer

[ECUC_StbM_00004] Definition of EcucParamConfContainerDef StbMTriggered Customer [

Container Name	StbMTriggeredCustomer		
Parent Container	StbM		
Description	The triggered customer is directly triggered by the Synchronized Time-base Manager by getting synchronized with the current (global) definition of time and passage of time.		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	-	
	Post-build time	-	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
StbMTriggeredCustomerPeriod	1	[ECUC_StbM_00020]
StbMOSScheduleTableRef	1	[ECUC_StbM_00007]
StbMSynchronizedTimeBaseRef	1	[ECUC_StbM_00010]

No Included Containers

]

[ECUC_StbM_00020] Definition of EcucIntegerParamDef StbMTriggeredCustomerPeriod [

Parameter Name	StbMTriggeredCustomerPeriod		
Parent Container	StbMTriggeredCustomer		
Description	The triggering period of the triggered customer, called by the StbM_MainFunction. The period is documented in microseconds.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
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		

]

[ECUC_StbM_00007] Definition of EcucReferenceDef StbMOSScheduleTableRef

[

Parameter Name	StbMOSScheduleTableRef		
Parent Container	StbMTriggeredCustomer		
Description	Mandatory reference to synchronized OS ScheduleTable, which will be explicitly synchronized by the StbM.		
Multiplicity	1		
Type	Reference to OsScheduleTable		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	-	
	Post-build time	-	
Scope / Dependency	scope: local		

]

[ECUC_StbM_00010] Definition of EcucReferenceDef StbMSynchronizedTimeBaseRef [

Parameter Name	StbMSynchronizedTimeBaseRef		
Parent Container	StbMTriggeredCustomer		
Description	Mandatory reference to the required synchronized time-base.		
Multiplicity	1		
Type	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		

]

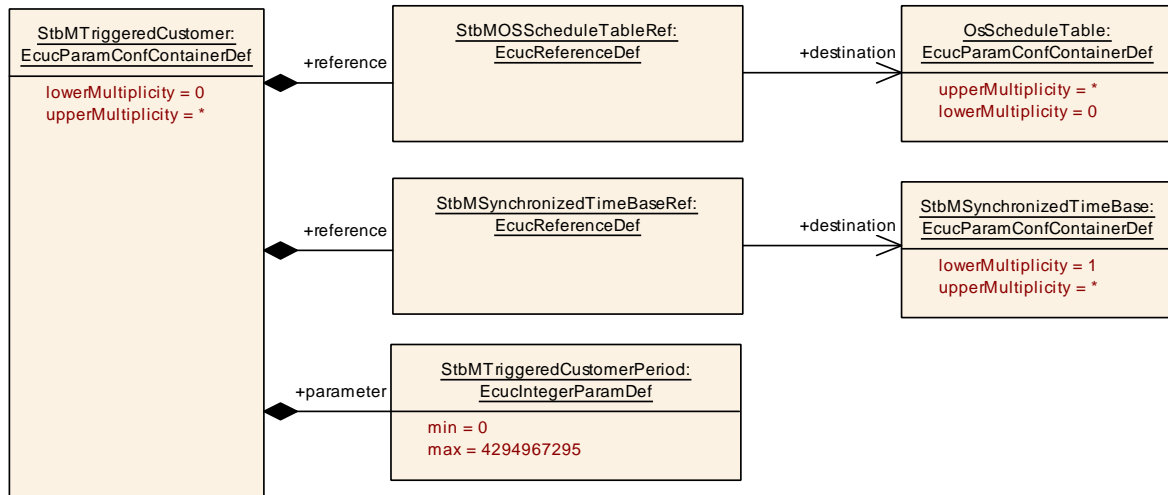


Figure 10.10

10.2.12 StbMFreshnessValueInformation

[ECUC_StbM_00075] Definition of EcucParamConfContainerDef StbMFreshness ValueInformation [

Container Name	StbMFreshnessValueInformation
Parent Container	StbM
Description	Container with the Freshness Value configurations
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
StbMGetRxFreshnessValueFuncName	0..1	[ECUC_StbM_00081]
StbMGetTxConfFreshnessValueFuncName	0..1	[ECUC_StbM_00080]
StbMGetTxFreshnessValueFuncName	0..1	[ECUC_StbM_00078]
StbMGetTxTruncFreshnessValueFuncName	0..1	[ECUC_StbM_00079]
StbMQueryFreshnessValue	1	[ECUC_StbM_00076]

Included Containers		
Container Name	Multiplicity	Scope / Dependency
StbMFreshnessValue	0..*	Container with the Freshness Value configurations

]

[ECUC_StbM_00081] Definition of EcucFunctionNameDef StbMGetRxFreshnessValueFuncName [

Parameter Name	StbMGetRxFreshnessValueFuncName		
Parent Container	StbMFreshnessValueInformation		
Description	Function pointer to call within StbM_GetRxFreshness() context.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	-		
Regular Expression	-		
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		

]

[ECUC_StbM_00080] Definition of EcucFunctionNameDef StbMGetTxConfFreshnessValueFuncName [

Parameter Name	StbMGetTxConfFreshnessValueFuncName		
Parent Container	StbMFreshnessValueInformation		
Description	Function pointer to call within StbM_SPduTxConfirmation() context.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	-		
Regular Expression	-		
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		

]

[ECUC_StbM_00078] Definition of EcucFunctionNameDef StbMGetTxFreshness ValueFuncName [

Parameter Name	StbMGetTxFreshnessValueFuncName		
Parent Container	StbMFreshnessValueInformation		
Description	Function pointer to call within StbM_GetTxFreshness() context.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	-		
Regular Expression	-		
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		

]

[ECUC_StbM_00079] Definition of EcucFunctionNameDef StbMGetTxTrunc FreshnessValueFuncName [

Parameter Name	StbMGetTxTruncFreshnessValueFuncName		
Parent Container	StbMFreshnessValueInformation		
Description	Function pointer to call within StbM_GetTxFreshnessTruncData() context.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	-		
Regular Expression	-		
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		

]

[ECUC_StbM_00076] Definition of EcucEnumerationParamDef StbMQueryFreshnessValue [

Parameter Name	StbMQueryFreshnessValue		
Parent Container	StbMFreshnessValueInformation		
Description	This parameter specifies if the freshness value shall be determined through a C-function (CD) or a software component (SW-C).		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CFUNC	The StbM queries the freshness for every time sync message using a C function API.	
	NONE	The StbM does not use the freshness value.	
	SERVICE	The StbM queries the freshness for every time sync message using the Rte service port FreshnessManagement.	
Default value	NONE		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

]

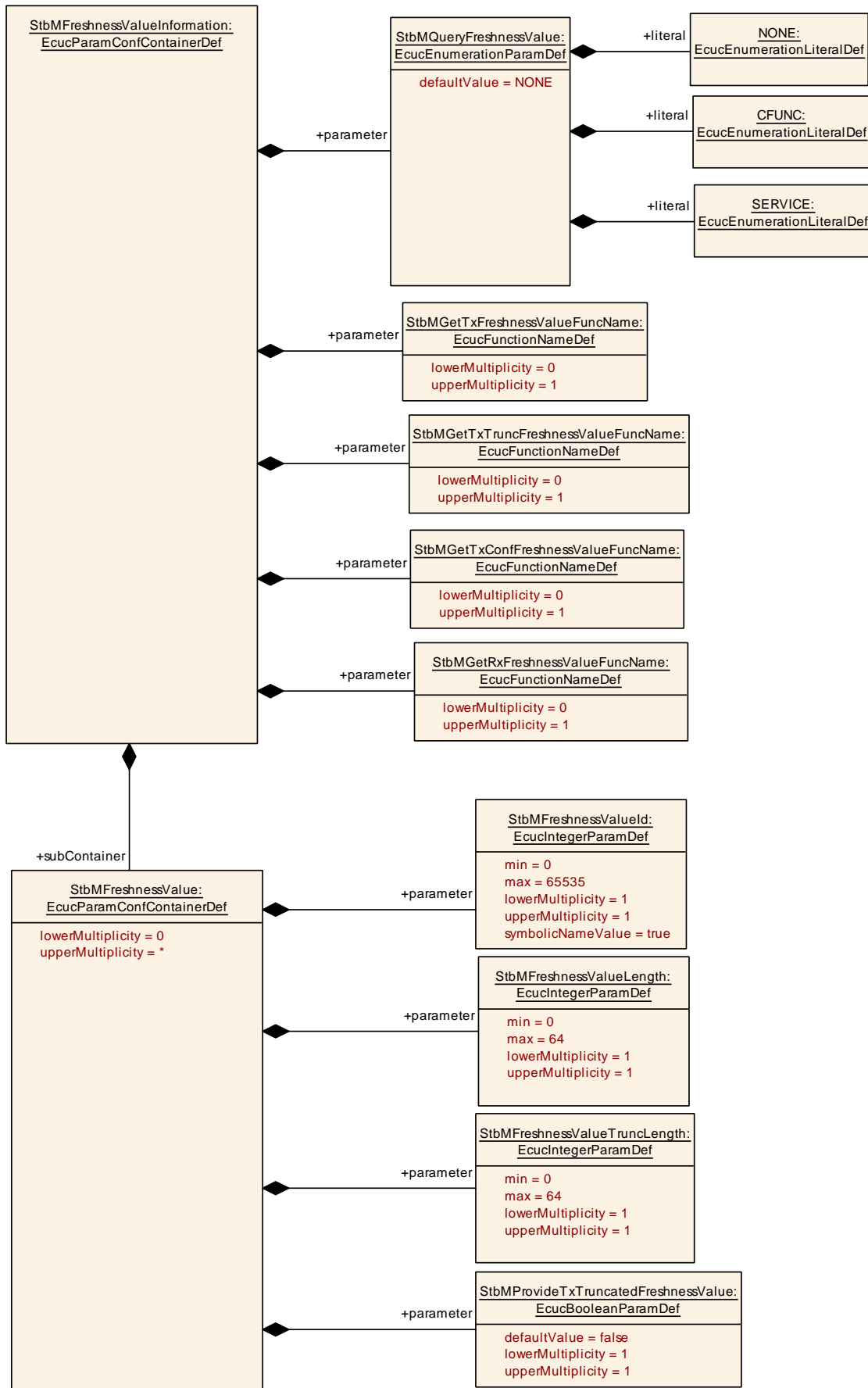


Figure 10.11

10.2.13 StbMFreshnessValue

[ECUC_StbM_00082] Definition of EcucParamConfContainerDef StbMFreshnessValue

Container Name	StbMFreshnessValue		
Parent Container	StbMFreshnessValueInformation		
Description	Container with the Freshness Value configurations		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
StbMFreshnessValueId	1	[ECUC_StbM_00083]
StbMFreshnessValueLength	1	[ECUC_StbM_00084]
StbMFreshnessValueTruncLength	1	[ECUC_StbM_00085]
StbMProvideTxTruncatedFreshnessValue	1	[ECUC_StbM_00087]

No Included Containers

]

[ECUC_StbM_00083] Definition of EcucIntegerParamDef StbMFreshnessValueId

[

Parameter Name	StbMFreshnessValueId		
Parent Container	StbMFreshnessValue		
Description	This parameter defines the Id of the Freshness Value. The Freshness Value might be a normal counter or a time value.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

]

[ECUC_StbM_00084] Definition of EcucIntegerParamDef StbMFreshnessValueLength

Parameter Name	StbMFreshnessValueLength		
Parent Container	StbMFreshnessValue		
Description	This parameter defines the complete length in bits of the Freshness Value. As long as the key doesn't change the counter shall not overflow. The length of the counter shall be determined based on the expected life time of the corresponding key and frequency of usage of the counter.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 64		
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		

]

[ECUC_StbM_00085] Definition of EcucIntegerParamDef StbMFreshnessValueTruncLength

Parameter Name	StbMFreshnessValueTruncLength		
Parent Container	StbMFreshnessValue		
Description	This parameter defines the length in bits of the Freshness Value to be included in the payload of the Secured Time Synchronization Messages. This length is specific to the least significant bits of the complete Freshness Counter. If the parameter is 0 no Freshness Value is included in the Secured Time Synchronization Messages.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 64		
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		

]

[ECUC_StbM_00087] Definition of EcucBooleanParamDef StbMProvideTxTruncatedFreshnessValue [

Parameter Name	StbMProvideTxTruncatedFreshnessValue		
Parent Container	StbMFreshnessValue		
Description	This parameter specifies if the FVM shall provide the truncated freshness value directly or if the FV shall be truncated by the StbM.		
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		

]

10.3 Constraints

[SWS_StbM_CONSTR_00001] [If variant is VARIANT-POST-BUILD, [StbMAllowSystemWideGlobalTimeMaster](#) shall be mandatory.]

[SWS_StbM_CONSTR_00002] [If variant is VARIANT-POST-BUILD, [StbMIsSystemWideGlobalTimeMaster](#) can only be set to TRUE, if [StbMAllowSystemWideGlobalTimeMaster](#) is set to TRUE.]

[SWS_StbM_CONSTR_00004]

Status: DRAFT

[If parameter [StbMSynchronizedTimeBaseType](#) is not equal to `TBTYPE_SYNCHRONIZED`, multiplicity of parameter [StbMCyclicBackupInterval](#) shall be set to 0.]

[SWS_StbM_CONSTR_00005]

Status: DRAFT

[If container [StbMTimeValidation](#) is configured for a Time Base, the container [StbMFallbackTimeClock](#) shall be mandatory for that Time Base.]

10.4 Published Information

For details refer to the chapter 10.3 "Published Information" in [4, SWS BSW General].

A Not applicable requirements

[SWS_StbM_NA_00140]

Upstream requirements: RS_TS_00027, RS_TS_20031, RS_TS_20032, RS_TS_20033, RS_TS_20034, RS_TS_20035, RS_TS_20037, RS_TS_20038, RS_TS_20039, RS_TS_20040, RS_TS_20041, RS_TS_20042, RS_TS_20043, RS_TS_20045, RS_TS_20046, RS_TS_20047, RS_TS_20048, RS_TS_20051, RS_TS_20052, RS_TS_20053, RS_TS_20054, RS_TS_20058, RS_TS_20059, RS_TS_20060, RS_TS_20061, RS_TS_20062, RS_TS_20066, RS_TS_20068, RS_TS_20071, RS_TS_20072, RS_TS_20073, RS_TS_20074, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00336, SRS_BSW_00344, SRS_BSW_00375, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00422, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00437, SRS_BSW_00438

[These requirements are not applicable to this specification.]

B Change history of AUTOSAR traceable items

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

B.1 Traceable item history of this document according to AUTOSAR Release R24-11

B.1.1 Added Specification Items in R24-11

[\[SWS_StbM_00627\]](#) [\[SWS_StbM_00628\]](#) [\[SWS_StbM_00629\]](#) [\[SWS_StbM_00630\]](#)
[\[SWS_StbM_91030\]](#)

B.1.2 Changed Specification Items in R24-11

[\[ECUC_StbM_00021\]](#) [\[ECUC_StbM_00049\]](#) [\[ECUC_StbM_00072\]](#) [\[ECUC_StbM_00095\]](#) [\[ECUC_StbM_00096\]](#) [\[ECUC_StbM_00097\]](#) [\[ECUC_StbM_00098\]](#) [\[ECUC_StbM_00100\]](#) [\[ECUC_StbM_00101\]](#) [\[SWS_StbM_00041\]](#) [\[SWS_StbM_00181\]](#)
[\[SWS_StbM_00182\]](#) [\[SWS_StbM_00183\]](#) [\[SWS_StbM_00185\]](#) [\[SWS_StbM_00187\]](#)
[\[SWS_StbM_00240\]](#) [\[SWS_StbM_00247\]](#) [\[SWS_StbM_00262\]](#) [\[SWS_StbM_00263\]](#)
[\[SWS_StbM_00272\]](#) [\[SWS_StbM_00278\]](#) [\[SWS_StbM_00305\]](#) [\[SWS_StbM_00312\]](#)
[\[SWS_StbM_00314\]](#) [\[SWS_StbM_00317\]](#) [\[SWS_StbM_00339\]](#) [\[SWS_StbM_00350\]](#)
[\[SWS_StbM_00351\]](#) [\[SWS_StbM_00353\]](#) [\[SWS_StbM_00356\]](#) [\[SWS_StbM_00362\]](#)
[\[SWS_StbM_00371\]](#) [\[SWS_StbM_00377\]](#) [\[SWS_StbM_00386\]](#) [\[SWS_StbM_00387\]](#)
[\[SWS_StbM_00392\]](#) [\[SWS_StbM_00394\]](#) [\[SWS_StbM_00397\]](#) [\[SWS_StbM_00399\]](#)
[\[SWS_StbM_00400\]](#) [\[SWS_StbM_00412\]](#) [\[SWS_StbM_00422\]](#) [\[SWS_StbM_00424\]](#)
[\[SWS_StbM_00425\]](#) [\[SWS_StbM_00428\]](#) [\[SWS_StbM_00431\]](#) [\[SWS_StbM_00437\]](#)
[\[SWS_StbM_00510\]](#) [\[SWS_StbM_00527\]](#) [\[SWS_StbM_00531\]](#) [\[SWS_StbM_00532\]](#)
[\[SWS_StbM_00535\]](#) [\[SWS_StbM_00537\]](#) [\[SWS_StbM_00576\]](#) [\[SWS_StbM_00577\]](#)
[\[SWS_StbM_00580\]](#) [\[SWS_StbM_00583\]](#) [\[SWS_StbM_00588\]](#) [\[SWS_StbM_00595\]](#)
[\[SWS_StbM_91011\]](#)

B.1.3 Deleted Specification Items in R24-11

[ECUC_StbM_00030] [ECUC_StbM_00032] [ECUC_StbM_00059] [ECUC_StbM_00061] [ECUC_StbM_00090] [SWS_StbM_00177] [SWS_StbM_00190] [SWS_StbM_00191] [SWS_StbM_00192] [SWS_StbM_00200] [SWS_StbM_00223] [SWS_StbM_00224] [SWS_StbM_00225] [SWS_StbM_00228] [SWS_StbM_00229] [SWS_StbM_00230] [SWS_StbM_00242] [SWS_StbM_00261] [SWS_StbM_00304] [SWS_StbM_00311] [SWS_StbM_00316] [SWS_StbM_00318] [SWS_StbM_00325] [SWS_StbM_00326] [SWS_StbM_00327] [SWS_StbM_00328] [SWS_StbM_00329] [SWS_StbM_00333] [SWS_StbM_00334] [SWS_StbM_00360] [SWS_StbM_00383] [SWS_StbM_00388] [SWS_StbM_00393] [SWS_StbM_00404] [SWS_StbM_00411] [SWS_StbM_00439] [SWS_StbM_00441] [SWS_StbM_00442] [SWS_StbM_00443] [SWS_StbM_00453] [SWS_StbM_00454] [SWS_StbM_00528] [SWS_StbM_00561] [SWS_StbM_00563] [SWS_StbM_00582] [SWS_StbM_91005]

B.1.4 Added Constraints in R24-11

none

B.1.5 Changed Constraints in R24-11

none

B.1.6 Deleted Constraints in R24-11

[SWS_StbM_CONSTR_00003]

B.2 Traceable item history of this document according to AUTOSAR Release R23-11

B.2.1 Added Specification Items in R23-11

[SWS_StbM_00556] [SWS_StbM_00557] [SWS_StbM_00558] [SWS_StbM_00559]
[SWS_StbM_00560] [SWS_StbM_00561] [SWS_StbM_00562] [SWS_StbM_00563]
[SWS_StbM_00564] [SWS_StbM_00565] [SWS_StbM_00566] [SWS_StbM_00567]
[SWS_StbM_00568] [SWS_StbM_00569] [SWS_StbM_00570] [SWS_StbM_00571]
[SWS_StbM_00572] [SWS_StbM_00573] [SWS_StbM_00574] [SWS_StbM_00575]
[SWS_StbM_00576] [SWS_StbM_00577] [SWS_StbM_00578] [SWS_StbM_00579]
[SWS_StbM_00580] [SWS_StbM_00581] [SWS_StbM_00582] [SWS_StbM_00583]
[SWS_StbM_00584] [SWS_StbM_00585] [SWS_StbM_00586] [SWS_StbM_00587]
[SWS_StbM_00588] [SWS_StbM_00589] [SWS_StbM_00590] [SWS_StbM_00591]

[SWS_StbM_00592] [SWS_StbM_00593] [SWS_StbM_00594] [SWS_StbM_00595]
 [SWS_StbM_00596] [SWS_StbM_00597] [SWS_StbM_00598] [SWS_StbM_00599]
 [SWS_StbM_00600] [SWS_StbM_00601] [SWS_StbM_00602] [SWS_StbM_00603]
 [SWS_StbM_00604] [SWS_StbM_00605] [SWS_StbM_00606] [SWS_StbM_00607]
 [SWS_StbM_00608] [SWS_StbM_00609] [SWS_StbM_00610] [SWS_StbM_00611]
 [SWS_StbM_00612] [SWS_StbM_00613] [SWS_StbM_00614] [SWS_StbM_00615]
 [SWS_StbM_00616] [SWS_StbM_00617] [SWS_StbM_00618] [SWS_StbM_00619]
 [SWS_StbM_00620] [SWS_StbM_00621] [SWS_StbM_00622] [SWS_StbM_00623]
 [SWS_StbM_00624] [SWS_StbM_00625] [SWS_StbM_00626] [SWS_StbM_91027]
 [SWS_StbM_91028] [SWS_StbM_91029]

B.2.2 Changed Specification Items in R23-11

[SWS_StbM_00051] [SWS_StbM_00059] [SWS_StbM_00171] [SWS_StbM_00173]
 [SWS_StbM_00177] [SWS_StbM_00178] [SWS_StbM_00179] [SWS_StbM_00191]
 [SWS_StbM_00239] [SWS_StbM_00240] [SWS_StbM_00246] [SWS_StbM_00247]
 [SWS_StbM_00261] [SWS_StbM_00284] [SWS_StbM_00304] [SWS_StbM_00308]
 [SWS_StbM_00311] [SWS_StbM_00332] [SWS_StbM_00339] [SWS_StbM_00342]
 [SWS_StbM_00353] [SWS_StbM_00355] [SWS_StbM_00356] [SWS_StbM_00359]
 [SWS_StbM_00362] [SWS_StbM_00373] [SWS_StbM_00374] [SWS_StbM_00377]
 [SWS_StbM_00384] [SWS_StbM_00387] [SWS_StbM_00393] [SWS_StbM_00395]
 [SWS_StbM_00396] [SWS_StbM_00397] [SWS_StbM_00399] [SWS_StbM_00400]
 [SWS_StbM_00411] [SWS_StbM_00422] [SWS_StbM_00424] [SWS_StbM_00426]
 [SWS_StbM_00431] [SWS_StbM_00433] [SWS_StbM_00434] [SWS_StbM_00436]
 [SWS_StbM_00437] [SWS_StbM_00438] [SWS_StbM_00439] [SWS_StbM_00440]
 [SWS_StbM_00441] [SWS_StbM_00442] [SWS_StbM_00443] [SWS_StbM_00458]
 [SWS_StbM_00459] [SWS_StbM_00460] [SWS_StbM_00461] [SWS_StbM_00462]
 [SWS_StbM_00463] [SWS_StbM_00466] [SWS_StbM_00469] [SWS_StbM_00470]
 [SWS_StbM_00507] [SWS_StbM_00508] [SWS_StbM_00516] [SWS_StbM_00517]
 [SWS_StbM_00528] [SWS_StbM_00529] [SWS_StbM_00531] [SWS_StbM_00532]
 [SWS_StbM_00534] [SWS_StbM_00535] [SWS_StbM_00538] [SWS_StbM_00541]
 [SWS_StbM_00542] [SWS_StbM_00543] [SWS_StbM_00544] [SWS_StbM_00545]
 [SWS_StbM_00546] [SWS_StbM_00547] [SWS_StbM_00548] [SWS_StbM_00549]
 [SWS_StbM_00550] [SWS_StbM_91014] [SWS_StbM_91016] [SWS_StbM_91018]
 [SWS_StbM_91023]

B.2.3 Deleted Specification Items in R23-11

none

B.2.4 Added Constraints in R23-11

[SWS_StbM_CONSTR_00005]

B.2.5 Changed Constraints in R23-11

[SWS_StbM_CONSTR_00003] [[SWS_StbM_CONSTR_00004](#)]

B.2.6 Deleted Constraints in R23-11

none