

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	R22-11

Document Change History			
Date	Release	Changed by	Description
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	11
1.1	Use Cases	11
1.2	Functional Overview	11
2	Acronyms and Abbreviations	15
2.1	Definitions	15
2.1.1	Time Base Customer	15
3	Related documentation	17
3.1	Input documents & related standards and norms	17
3.2	Related specification	17
4	Constraints and assumptions	18
4.1	Limitations	18
4.1.1	OS ScheduleTable	18
4.1.2	Synchronized Time Base Identifier	18
4.1.3	Mode switches	18
4.1.4	Configuration	18
4.1.5	Out of scope	18
4.2	Applicability to car domains	19
4.3	Conflicts	19
5	Dependencies to other modules	20
5.1	Code file structure	20
5.2	Header file structure	20
6	Requirements Tracing	21
7	Functional specification	32
7.1	Startup behavior	32
7.1.1	Preconditions	32
7.1.2	Initialization	32
7.2	Shutdown behavior	33
7.3	Normal operation	34
7.3.1	Introduction	34
7.3.1.1	Virtual Local Time	35
7.3.1.2	Types of Time Bases	37
7.3.1.3	Roles of the StbM	37
7.3.1.4	Interpolating the Global Time	39
7.3.2	Synchronized Time Bases	42
7.3.2.1	Global Time Master	43
7.3.2.2	Time Slave	43
7.3.3	Offset Time Bases	44
7.3.3.1	Global Time Master	45
7.3.3.2	Time Slave	45

7.3.4	Pure Local Time Bases	46
7.3.5	Synchronization State	47
7.3.5.1	Global Time Master	48
7.3.5.2	Time Slaves	48
7.3.6	Immediate Time Synchronization	51
7.3.7	User Data	51
7.3.8	Time Correction	52
7.3.8.1	Rate Correction Measurement (for Time Slaves)	52
7.3.8.2	Time Interpolation, Rate and Offset Correction (for Time Slaves)	57
7.3.8.3	Time Interpolation and Rate Correction for Global Time Masters	60
7.3.9	Time Base Cloning	63
7.3.10	Notification of Customers	65
7.3.10.1	Time Notifications	65
7.3.10.2	Status Notifications	68
7.3.11	Triggering Customers	70
7.3.12	Time Recording	72
7.3.12.1	General	72
7.3.12.2	Global Time Precision Measurement Support	72
7.3.12.3	Time Validation Support	77
7.3.13	Freshness Value	86
7.3.13.1	General	86
7.3.14	Interaction with User Defined Timesync Module (CDD)	87
7.4	Multicore Distribution	87
7.5	Error Handling	88
7.6	Error Classification	88
7.6.1	Development Errors	88
7.6.2	Runtime Errors	88
7.6.3	Transient Faults	89
7.6.4	Production Errors	89
7.6.5	Extended Production Errors	89
7.7	Version Check	89
8	API specification	90
8.1	Imported types	90
8.2	Type definitions	90
8.2.1	Extension to Std_ReturnType	90
8.2.2	StbM_ConfigType	91
8.2.3	StbM_VirtualLocalTimeType	91
8.2.4	StbM_MeasurementType	91
8.3	Function definitions	92
8.3.1	StbM_GetVersionInfo	92
8.3.2	StbM_Init	92
8.3.3	StbM_GetCurrentTime	93
8.3.4	StbM_GetCurrentTimeExtended	94

8.3.5	StbM_GetCurrentVirtualLocalTime	94
8.3.6	StbM_SetGlobalTime	95
8.3.7	StbM_UpdateGlobalTime	96
8.3.8	StbM_SetUserData	97
8.3.9	StbM_SetOffset	98
8.3.10	StbM_GetOffset	99
8.3.11	StbM_BusGetCurrentTime	100
8.3.12	StbM_BusSetGlobalTime	100
8.3.13	StbM_GetRateDeviation	102
8.3.14	StbM_SetRateCorrection	102
8.3.15	StbM_GetTimeLeap	103
8.3.16	StbM_GetTimeBaseStatus	104
8.3.17	StbM_CloneTimeBase	105
8.3.18	StbM_StartTimer	106
8.3.19	StbM_GetSyncTimeRecordHead	107
8.3.20	StbM_GetOffsetTimeRecordHead	108
8.3.21	StbM_TriggerTimeTransmission	109
8.3.22	StbM_GetTimeBaseUpdateCounter	109
8.3.23	StbM_GetMasterConfig	110
8.3.24	StbM_CanSetSlaveTimingData	111
8.3.25	StbM_FrSetSlaveTimingData	111
8.3.26	StbM_EthSetSlaveTimingData	112
8.3.27	StbM_CanSetMasterTimingData	113
8.3.28	StbM_FrSetMasterTimingData	113
8.3.29	StbM_EthSetMasterTimingData	114
8.3.30	StbM_EthSetPdelayInitiatorData	115
8.3.31	StbM_EthSetPdelayResponderData	116
8.3.32	StbM_GetBusProtocolParam	117
8.3.33	StbM_SetBusProtocolParam	117
8.3.34	StbM_GetTxFreshness	118
8.3.35	StbM_GetTxFreshnessTruncData	119
8.3.36	StbM_SPduTxConfirmation	120
8.3.37	StbM_GetRxFreshness	121
8.4	Callback notifications	122
8.5	Scheduled functions	122
8.5.1	StbM_MainFunction	122
8.6	Expected interfaces	122
8.6.1	Mandatory interfaces	122
8.6.2	Optional interfaces	123
8.6.3	Configurable interfaces	124
8.6.3.1	SyncTimeRecordBlockCallback	124
8.6.3.2	OffsetTimeRecordBlockCallback	124
8.6.3.3	StatusNotificationCallback	125
8.6.3.4	<Customer>_TimeNotificationCallback	126
8.6.3.5	SPduTxConfirmationFct	126
8.6.3.6	GetTxFreshnessTruncDataFct	127

8.6.3.7	GetTxFreshnessFct	127
8.6.3.8	GetRxFreshnessFct	128
8.7	Service Interfaces	129
8.7.1	Provided Ports	129
8.7.1.1	GlobalTime_Master	129
8.7.1.2	GlobalTime_Slave	130
8.7.1.3	GlobalTime_StatusEvent	130
8.7.1.4	StartTimer	130
8.7.2	Required Ports	131
8.7.2.1	GlobalTime_TimeEvent	131
8.7.2.2	GlobalTime_Measurement	131
8.7.2.3	TimeBaseProviderNotification_Eth	131
8.7.2.4	TimeBaseProviderNotification_Fr	132
8.7.2.5	TimeBaseProviderNotification_Can	132
8.7.2.6	FreshnessManagement	132
8.7.3	Sender-Receiver Interfaces	133
8.7.3.1	StatusNotification	133
8.7.4	Client-Server-Interfaces	133
8.7.4.1	GlobalTime_Master	133
8.7.4.2	GlobalTime_Slave	136
8.7.4.3	StartTimer	139
8.7.4.4	TimeNotification	140
8.7.4.5	MeasurementNotification	140
8.7.4.6	TimeBaseProviderNotification_Eth	141
8.7.4.7	TimeBaseProviderNotification_Fr	143
8.7.4.8	TimeBaseProviderNotification_Can	144
8.7.4.9	FreshnessManagement	145
8.7.5	Implementation Data Types	147
8.7.5.1	StbM_PortIdType	148
8.7.5.2	StbM_SynchronizedTimeBaseType	148
8.7.5.3	StbM_TimeBaseStatusType	148
8.7.5.4	StbM_TimeStampShortType	149
8.7.5.5	StbM_TimeStampType	150
8.7.5.6	StbM_TimeStampExtendedType	150
8.7.5.7	StbM_TimeTupleType	151
8.7.5.8	StbM_TimeDiffType	151
8.7.5.9	StbM_RateDeviationType	152
8.7.5.10	StbM_CloneConfigType	152
8.7.5.11	StbM_UserDataType	153
8.7.5.12	StbM_CustomerIdType	153
8.7.5.13	StbM_TimeBaseNotificationType	154
8.7.5.14	StbM_SyncRecordTableHeadType	155
8.7.5.15	StbM_SyncRecordTableBlockType	155
8.7.5.16	StbM_OffsetRecordTableHeadType	156
8.7.5.17	StbM_OffsetRecordTableBlockType	156
8.7.5.18	StbM_MasterConfigType	157

8.7.5.19	StbM_EthTimeMasterMeasurementType	157
8.7.5.20	StbM_FrTimeMasterMeasurementType	158
8.7.5.21	StbM_CanTimeMasterMeasurementType	159
8.7.5.22	StbM_EthTimeSlaveMeasurementType	159
8.7.5.23	StbM_FrTimeSlaveMeasurementType	160
8.7.5.24	StbM_CanTimeSlaveMeasurementType	161
8.7.5.25	StbM_PdelayInitiatorMeasurementType	162
8.7.5.26	StbM_PdelayResponderMeasurementType	163
8.7.5.27	StbM_TimeSyncType	164
8.7.5.28	StbM_ProtocolParamType	164
8.7.5.29	StbM_FreshnessArrayType	165
9	Sequence diagrams	166
9.1	StbM Initialization	166
9.2	Immediate Time Synchronisation	167
9.3	Explicit synchronization of OS ScheduleTable	168
10	Configuration specification	169
10.1	How to read this chapter	169
10.2	Containers and configuration parameters	169
10.2.1	StbM	169
10.2.2	StbMGeneral	171
10.2.3	StbMSynchronizedTimeBase	174
10.2.4	StbMTimeCorrection	182
10.2.5	StbMLocalTimeClock	186
10.2.6	StbMTimeRecording	188
10.2.7	StbMTimeValidation	191
10.2.8	StbMNotificationCustomer	192
10.2.9	StbMTriggeredCustomer	193
10.2.10	StbMFreshnessValueInformation	195
10.2.11	StbMFreshnessValue	199
10.3	Constraints	200
10.4	Published Information	200
A	Not applicable requirements	201

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

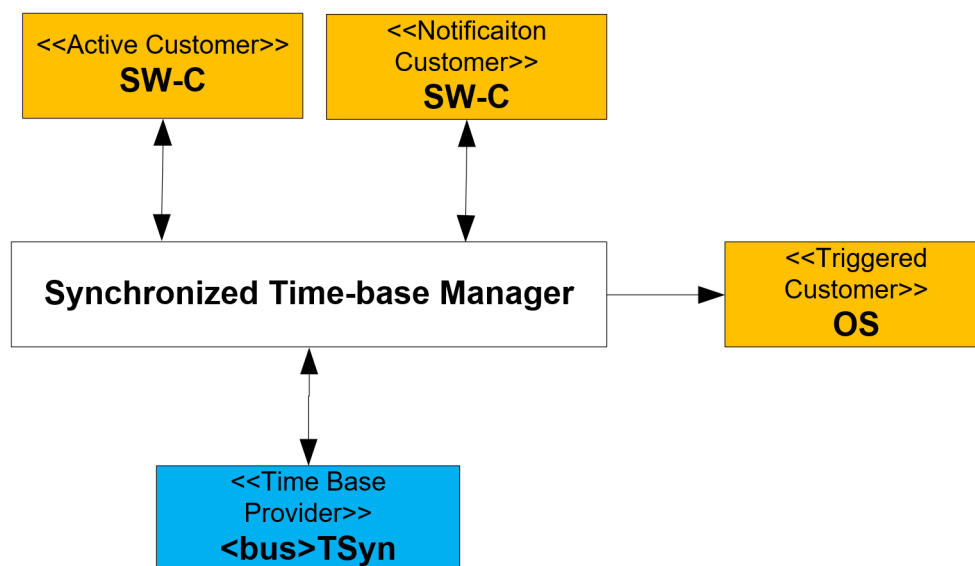


Figure 1.1: Synchronized Time-Base Manager as broker

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 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

For a detailed description of those three types refer to chapter [2.1.1 “Time Base 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, taking account their Rate Deviations and Offsets to the underlying Virtual Local Time.

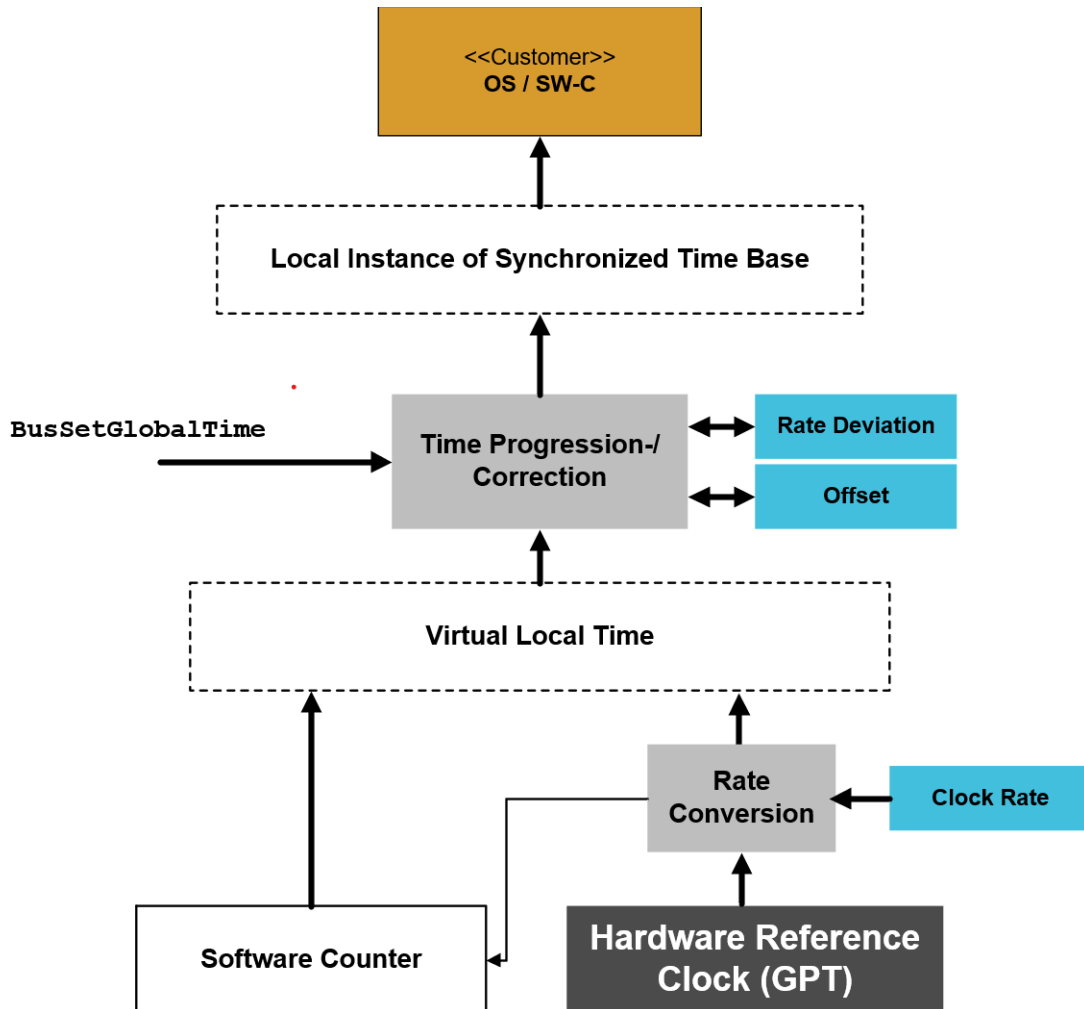


Figure 1.2: Abstract Working Principle of the Synchronized Time-Base Manager

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.

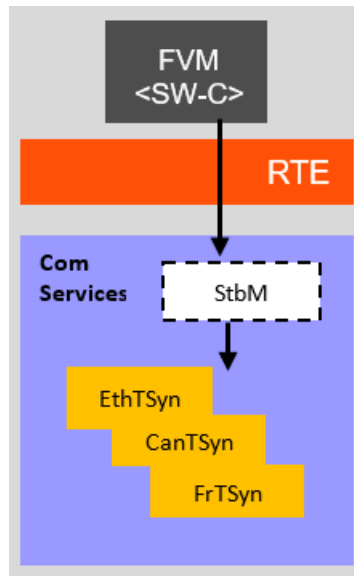


Figure 1.3: StbM as interface for the Freshness Value

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 and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the StbM module that are not included in the [2, AUTOSAR glossary].

Abbreviation	Description
GTM	Global Time Master
(G)TD	(Global) Time Domain
(G)TM	(Global)Time Master
<Bus>TSyn	A bus specific Time Synchronization Provider module
AVB	Audio Video Bridging
BMCA	Best Master Clock Algorithm
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
GM(C)	Grand Master (Clock)
GTS	Global Time Synchronization
NvM	Non-volatile Memory
OFNS message	Time Synchronization message for an Offset Time Base (containing the nanosecond part of the time)
OFS message	Time Synchronization message for an Offset Time Base
PTP	Precision Time Protocol
StbM	Synchronized Time-Base Manager
SYNC message	Time Synchronization message for a Synchronized Time Base
TG	Time Gateway
Timesync	Time Synchronization
TS	Time Slave
TSP	Time Synchronization Provider

2.1 Definitions

2.1.1 Time Base Customer

The Synchronized Time-Base Manager supports 3 types of Time Base Customers:

- **Active Customer**

This kind of customer autonomously calls the Synchronized Time-Base Manager either

- to read time information from the Synchronized Time-Base Manager or

- to update the Time Base maintained by the Synchronized Time-Base Manager according to application information.

- **Triggered Customer**

This kind of customer 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.

- **Notification Customer**

This kind of customer is notified by the Synchronized Time-Base Manager, if the following Time Base related events occur:

- 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.

3 Related documentation

3.1 Input documents & related standards and norms

- [1] Specification of Timing Extensions
AUTOSAR_TPS_TimingExtensions
- [2] Glossary
AUTOSAR_TR_Glossary
- [3] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral
- [4] Specification of Operating System
AUTOSAR_SWS_OS
- [5] Requirements on Time Synchronization
AUTOSAR_RS_TimeSync
- [6] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral
- [7] System Template
AUTOSAR_TPS_SystemTemplate
- [8] Specification of Secure Onboard Communication
AUTOSAR_SWS_SecureOnboardCommunication
- [9] Complex Driver design and integration guideline
AUTOSAR_EXP_CDDDesignAndIntegrationGuideline
- [10] Guide to BSW Distribution
AUTOSAR_EXP_BSWDistributionGuide
- [11] IEEE Standard 802.1AS-2011

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [3, 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 [4]). 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 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 [3, SWS BSW General].

5.2 Header file structure

For details, refer to the section 5.1.7 "Header file structure" of the [3, SWS BSW General].

In addition to the files defined in section 5.1.7 "Header file structure" of the [3, SWS BSW General], the StbM needs to include the file Os.h, CanIf.h, EthIf.h and Gpt.h.

[SWS_StbM_00065] [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.]([SRS_BSW_00384](#))

[SWS_StbM_00246] [If time stamping via Ethernet shall be supported (refer to [EthIfGlobalTimeSupport](#), which is referenced via [StbMLocalTimeHardware](#), if set to [EthTSynGlobalTimeDomain](#)), StbM.c shall include EthIf.h to have access to the interface of the EthIf module.]([SRS_BSW_00384](#))

[SWS_StbM_00538]{DRAFT} [If CAN hardware timestamping is supported (refer to configuration parameter [CanIfGlobalTimeSupport](#) in [CanIf](#), which is referenced via [StbMLocalTimeHardware](#), if set to [CanTSynGlobalTimeDomain](#)), StbM.c shall include CanIf.h to have access to the interface of the CanIf module.]([SRS_BSW_00384](#), [RS_TS_20070](#))

[SWS_StbM_00426] [If time stamping via GPT shall be supported (which is referenced via [StbMLocalTimeHardware](#), if set to [GptChannelConfiguration](#)), StbM.c shall include Gpt.h to have access to the interface of the GPT module.]([RS_TS_00017](#), [RS_TS_00002](#))

6 Requirements Tracing

The following tables reference the requirements specified in [5, 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]
[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]
[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_00200] [SWS_StbM_00240] [SWS_StbM_00244] [SWS_StbM_00247] [SWS_StbM_00248] [SWS_StbM_00261] [SWS_StbM_00262] [SWS_StbM_00263] [SWS_StbM_00267] [SWS_StbM_00434] [SWS_StbM_00436] [SWS_StbM_91005] [SWS_StbM_91013]
[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_91005] [SWS_StbM_91006]
[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_00393] [SWS_StbM_00438] [SWS_StbM_00439] [SWS_StbM_00528] [SWS_StbM_00529]
[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_91006]
[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_00393] [SWS_StbM_00399] [SWS_StbM_00425] [SWS_StbM_00438] [SWS_StbM_00439] [SWS_StbM_00528] [SWS_StbM_00529] [SWS_StbM_00540] [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]
[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]





Requirement	Description	Satisfied by
[RS_TS_00012]	The Implementation of Time Synchronization shall allow customers and TSP modules to read the offset value of an Offset Time Base	[SWS_StbM_00191] [SWS_StbM_00228] [SWS_StbM_CONSTR_00003]
[RS_TS_00013]	The Implementation of Time Synchronization shall allow the customers and TSP modules to set the offset value of an Offset Master Time Base	[SWS_StbM_00177] [SWS_StbM_00190] [SWS_StbM_00191] [SWS_StbM_00192] [SWS_StbM_00223] [SWS_StbM_00240] [SWS_StbM_00244] [SWS_StbM_00304] [SWS_StbM_CONSTR_00003]
[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_00192] [SWS_StbM_00195] [SWS_StbM_00200] [SWS_StbM_00243] [SWS_StbM_00247] [SWS_StbM_00248] [SWS_StbM_00434] [SWS_StbM_00436] [SWS_StbM_91005]
[RS_TS_00015]	The Implementation of Time Synchronization shall allow customers to set User Data propagated via the TSP modules.	[SWS_StbM_00190] [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_00360] [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_00411] [SWS_StbM_00412] [SWS_StbM_00422] [SWS_StbM_00424] [SWS_StbM_00431] [SWS_StbM_00440] [SWS_StbM_00441] [SWS_StbM_00442] [SWS_StbM_00443] [SWS_StbM_00527]
[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]





Requirement	Description	Satisfied by
[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] [SWS_StbM_CONSTR_00004]
[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_00027]	The TS shall provide a bus independent customer interface	[SWS_StbM_NA_00140]
[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_00223] [SWS_StbM_00228] [SWS_StbM_00244] [SWS_StbM_00408] [SWS_StbM_00490] [SWS_StbM_00491] [SWS_StbM_00492] [SWS_StbM_91001] [SWS_StbM_91002] [SWS_StbM_91005]
[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]
[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_00228] [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_91005]
[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_00311] [SWS_StbM_00312] [SWS_StbM_00313] [SWS_StbM_00314] [SWS_StbM_00315] [SWS_StbM_00316] [SWS_StbM_00317] [SWS_StbM_00318] [SWS_StbM_00319] [SWS_StbM_00320] [SWS_StbM_00322] [SWS_StbM_00323] [SWS_StbM_00325] [SWS_StbM_00326] [SWS_StbM_00328] [SWS_StbM_00329] [SWS_StbM_00331] [SWS_StbM_00332] [SWS_StbM_00333] [SWS_StbM_00334] [SWS_StbM_00339] [SWS_StbM_00382] [SWS_StbM_00383] [SWS_StbM_00384] [SWS_StbM_00387] [SWS_StbM_00388] [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]





Requirement	Description	Satisfied by
		<p>△</p> <p>[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] [SWS_StbM_00523] [SWS_StbM_00524] [SWS_StbM_00525]</p>
[RS_TS_00035]	The Implementation of Time Synchronization shall provide a system service interface to applications	<p>[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]</p>
[RS_TS_00036]	The Implementation of Time Synchronization shall provide a bus independent customer interface	<p>[SWS_StbM_00241] [SWS_StbM_00242]</p>
[RS_TS_00037]	The configuration of the Time Synchronization implementation shall allow the interaction with different types of customers	<p>[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]</p>
[RS_TS_00038]	The Implementation of Time Synchronization shall copy Time Base information upon user request	<p>[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_91011] [SWS_StbM_91012]</p>
[RS_TS_00039]	The implementation of Time Synchronization shall provide Freshness Value (FV) to TSP modules required to secure the time information	<p>[SWS_StbM_00541] [SWS_StbM_00542] [SWS_StbM_00543] [SWS_StbM_00551] [SWS_StbM_00552] [SWS_StbM_00553] [SWS_StbM_00554] [SWS_StbM_91014] [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]</p>
[RS_TS_20031]	The Timesync over CAN module shall trigger Time Base Synchronization transmission	<p>[SWS_StbM_NA_00140]</p>
[RS_TS_20032]	The Timesync over CAN module shall provide the Time Base after reception of a valid Timesync/TS messages	<p>[SWS_StbM_NA_00140]</p>
[RS_TS_20033]	The Timesync over CAN module shall support means to protect the Time synchronization protocol	<p>[SWS_StbM_NA_00140]</p>
[RS_TS_20034]	The Timesync over CAN module shall detect and handle timeout and integrity errors in the Time Synchronization protocol	<p>[SWS_StbM_NA_00140]</p>





Requirement	Description	Satisfied by
[RS_TS_20035]	The Timesync over CAN module shall support a protocol for precise time measurement and synchronization over CAN	[SWS_StbM_NA_00140]
[RS_TS_20036]	The Timesync over CAN module shall use the time measurement and synchronization protocol to transmit and receive an offset value	[SWS_StbM_NA_00140]
[RS_TS_20037]	The Timesync over CAN module shall support user specific data within the time measurement and synchronization protocol	[SWS_StbM_NA_00140]
[RS_TS_20038]	The Timesync over CAN module configuration shall allow the Implementation of Time Synchronization for CAN to support different roles for a Time Base	[SWS_StbM_NA_00140]
[RS_TS_20039]	The Timesync over FlexRay module shall trigger Time Base Synchronization transmission	[SWS_StbM_NA_00140]
[RS_TS_20040]	The Timesync over FlexRay module shall provide a Time Base after reception of a valid protocol information	[SWS_StbM_NA_00140]
[RS_TS_20041]	The Timesync over FlexRay module shall support means to protect the Time Synchronization protocol	[SWS_StbM_NA_00140]
[RS_TS_20042]	The Timesync over FlexRay module shall detect and handle timeout and integrity errors in the Time Synchronization protocol	[SWS_StbM_NA_00140]
[RS_TS_20043]	The Timesync over FlexRay module shall support a protocol for precise time measurement and synchronization over FlexRay	[SWS_StbM_NA_00140]
[RS_TS_20044]	The Timesync over FlexRay module shall use the time measurement and synchronization protocol to transmit and receive an offset value	[SWS_StbM_NA_00140]
[RS_TS_20045]	The Timesync over FlexRay module shall support user specific data within the time measurement and synchronization protocol	[SWS_StbM_NA_00140]
[RS_TS_20046]	The configuration for Time synchronization over FlexRay shall allow the FlexRay Time Synchronization module to support different roles for a Time Base	[SWS_StbM_NA_00140]
[RS_TS_20047]	The Timesync over Ethernet module shall trigger Time Base Synchronization transmission	[SWS_StbM_NA_00140]
[RS_TS_20048]	The Timesync over Ethernet module shall support IEEE 802.1AS as well as AUTOSAR extensions	[SWS_StbM_NA_00140]
[RS_TS_20051]	The Timesync over Ethernet module shall detect and handle errors in synchronization protocol / communication	[SWS_StbM_NA_00140]





Requirement	Description	Satisfied by
[RS_TS_20052]	The configuration of the Time Synchronization over Ethernet module shall allow the module to work as a Time Master	[SWS_StbM_NA_00140]
[RS_TS_20053]	The configuration of the Time Synchronization over Ethernet module shall allow the module to work as a Time Slave	[SWS_StbM_NA_00140]
[RS_TS_20054]	The Implementation of the Time Synchronization shall evaluate and propagate Time Gateway relevant information	[SWS_StbM_NA_00140]
[RS_TS_20058]	The Timesync over Ethernet module shall provide the precision of Synchronized Time Bases	[SWS_StbM_NA_00140]
[RS_TS_20059]	The Timesync over Ethernet module shall access all communication ports belonging to Time Synchronization	[SWS_StbM_NA_00140]
[RS_TS_20060]	The Timesync over Ethernet module shall provide a Time Base after reception of a valid protocol information	[SWS_StbM_NA_00140]
[RS_TS_20061]	The Timesync over Ethernet module shall support means to protect the Time Synchronization protocol	[SWS_StbM_NA_00140]
[RS_TS_20062]	The Timesync over Ethernet module shall support user specific data within the time measurement and synchronization protocol	[SWS_StbM_NA_00140]
[RS_TS_20063]	The Timesync over Ethernet module shall use the Time Synchronization protocol for Synchronized Time Bases to transmit and receive Offset Time Bases	[SWS_StbM_NA_00140]
[RS_TS_20066]	The Timesync over Ethernet module shall support a static (pre)configuration of IEEE 802.1AS Pdelay	[SWS_StbM_NA_00140]
[RS_TS_20068]	The Timesync over CAN module shall support classic CAN and CAN FD	[SWS_StbM_NA_00140]
[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]
[RS_TS_20071]	The Timesync over Ethernet module shall enable time synchronization on peer-to-peer and multidrop topologies	[SWS_StbM_NA_00140]
[RS_TS_20072]	The Timesync over Ethernet module shall support means to secure the Time Synchronization protocol	[SWS_StbM_NA_00140]
[RS_TS_20073]	The Timesync over CAN module shall support means to secure the Time Synchronization protocol	[SWS_StbM_NA_00140]





Requirement	Description	Satisfied by
[RS_TS_20074]	The Timesync over FlexRay module shall support means to secure the Time Synchronization protocol	[SWS_StbM_NA_00140]
[SRS_BSW_00005]	Modules of the μ C Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	[SWS_StbM_NA_00140]
[SRS_BSW_00006]	The source code of software modules above the μ C Abstraction Layer (MCAL) shall not be processor and compiler dependent.	[SWS_StbM_NA_00140]
[SRS_BSW_00007]	All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.	[SWS_StbM_NA_00140]
[SRS_BSW_00009]	All Basic SW Modules shall be documented according to a common standard.	[SWS_StbM_NA_00140]
[SRS_BSW_00010]	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	[SWS_StbM_NA_00140]
[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_00160]	Configuration files of AUTOSAR Basic SW module shall be readable for human beings	[SWS_StbM_NA_00140]
[SRS_BSW_00161]	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	[SWS_StbM_NA_00140]
[SRS_BSW_00162]	The AUTOSAR Basic Software shall provide a hardware abstraction layer	[SWS_StbM_NA_00140]
[SRS_BSW_00164]	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	[SWS_StbM_NA_00140]
[SRS_BSW_00168]	SW components shall be tested by a function defined in a common API in the Basis-SW	[SWS_StbM_NA_00140]
[SRS_BSW_00170]	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	[SWS_StbM_NA_00140]
[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_00304]	All AUTOSAR Basic Software Modules shall use only AUTOSAR data types instead of native C data types	[SWS_StbM_NA_00140]
[SRS_BSW_00305]	Data types naming convention	[SWS_StbM_00142]
[SRS_BSW_00307]	Global variables naming convention	[SWS_StbM_NA_00140]





Requirement	Description	Satisfied by
[SRS_BSW_00308]	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	[SWS_StbM_NA_00140]
[SRS_BSW_00309]	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	[SWS_StbM_NA_00140]
[SRS_BSW_00312]	Shared code shall be reentrant	[SWS_StbM_NA_00140]
[SRS_BSW_00314]	All internal driver modules shall separate the interrupt frame definition from the service routine	[SWS_StbM_NA_00140]
[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_00224] [SWS_StbM_00225] [SWS_StbM_00229] [SWS_StbM_00230] [SWS_StbM_00234] [SWS_StbM_00235] [SWS_StbM_00264] [SWS_StbM_00268] [SWS_StbM_00269] [SWS_StbM_00296] [SWS_StbM_00298] [SWS_StbM_00327] [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_00404] [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_00453] [SWS_StbM_00454] [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]
[SRS_BSW_00325]	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	[SWS_StbM_NA_00140]
[SRS_BSW_00327]	Error values naming convention	[SWS_StbM_00041]
[SRS_BSW_00328]	All AUTOSAR Basic Software Modules shall avoid the duplication of code	[SWS_StbM_NA_00140]
[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_00334]	All Basic Software Modules shall provide an XML file that contains the meta data	[SWS_StbM_NA_00140]





Requirement	Description	Satisfied by
[SRS_BSW_00336]	Basic SW module shall be able to shutdown	[SWS_StbM_NA_00140]
[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_00341]	Module documentation shall contains all needed informations	[SWS_StbM_NA_00140]
[SRS_BSW_00342]	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	[SWS_StbM_NA_00140]
[SRS_BSW_00344]	BSW Modules shall support link-time configuration	[SWS_StbM_NA_00140]
[SRS_BSW_00347]	A Naming seperation of different instances of BSW drivers shall be in place	[SWS_StbM_NA_00140]
[SRS_BSW_00353]	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	[SWS_StbM_NA_00140]
[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_00375]	Basic Software Modules shall report wake-up reasons	[SWS_StbM_NA_00140]
[SRS_BSW_00378]	AUTOSAR shall provide a boolean type	[SWS_StbM_NA_00140]
[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_00224] [SWS_StbM_00225] [SWS_StbM_00229] [SWS_StbM_00230] [SWS_StbM_00234] [SWS_StbM_00235] [SWS_StbM_00264] [SWS_StbM_00268] [SWS_StbM_00269] [SWS_StbM_00296] [SWS_StbM_00298] [SWS_StbM_00327] [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_00404] [SWS_StbM_00405] [SWS_StbM_00406] [SWS_StbM_00415]





Requirement	Description	Satisfied by
		[SWS_StbM_00416] [SWS_StbM_00444] [SWS_StbM_00445] [SWS_StbM_00448] [SWS_StbM_00449] [SWS_StbM_00451] [SWS_StbM_00452] [SWS_StbM_00453] [SWS_StbM_00454] [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]
[SRS_BSW_00398]	The link-time configuration is achieved on object code basis in the stage after compiling and before linking	[SWS_StbM_NA_00140]
[SRS_BSW_00399]	Parameter-sets shall be located in a separate segment and shall be loaded after the code	[SWS_StbM_NA_00140]
[SRS_BSW_00400]	Parameter shall be selected from multiple sets of parameters after code has been loaded and started	[SWS_StbM_NA_00140]
[SRS_BSW_00404]	BSW Modules shall support post-build configuration	[SWS_StbM_NA_00140]
[SRS_BSW_00405]	BSW Modules shall support multiple configuration sets	[SWS_StbM_NA_00140]
[SRS_BSW_00406]	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	[SWS_StbM_00100] [SWS_StbM_00121]
[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_00413]	An index-based accessing of the instances of BSW modules shall be done	[SWS_StbM_NA_00140]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_StbM_00052] [SWS_StbM_00249]
[SRS_BSW_00415]	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	[SWS_StbM_NA_00140]
[SRS_BSW_00416]	The sequence of modules to be initialized shall be configurable	[SWS_StbM_NA_00140]
[SRS_BSW_00417]	Software which is not part of the SW-C shall report error events only after the Dem is fully operational.	[SWS_StbM_NA_00140]
[SRS_BSW_00422]	Pre-de-bouncing of error status information is done within the Dem	[SWS_StbM_NA_00140]
[SRS_BSW_00426]	BSW Modules shall ensure data consistency of data which is shared between BSW modules	[SWS_StbM_NA_00140]





Requirement	Description	Satisfied by
[SRS_BSW_00427]	ISR functions shall be defined and documented in the BSW module description template	[SWS_StbM_NA_00140]
[SRS_BSW_00428]	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	[SWS_StbM_NA_00140]
[SRS_BSW_00429]	Access to OS is restricted	[SWS_StbM_00020] [SWS_StbM_00092]
[SRS_BSW_00432]	Modules should have separate main processing functions for read/receive and write/transmit data path	[SWS_StbM_NA_00140]
[SRS_BSW_00433]	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	[SWS_StbM_NA_00140]
[SRS_BSW_00437]	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	[SWS_StbM_NA_00140]
[SRS_BSW_00438]	Configuration data shall be defined in a structure	[SWS_StbM_NA_00140]
[SRS_BSW_00439]	Enable BSW modules to handle interrupts	[SWS_StbM_NA_00140]
[SRS_BSW_00440]	The callback function invocation by the BSW module shall follow the signature provided by RTE to invoke servers via <code>Rte_Call</code> API	[SWS_StbM_NA_00140]
[SRS_BSW_00453]	BSW Modules shall be harmonized	[SWS_StbM_NA_00140]
[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: RequirementsTracing

7 Functional specification

7.1 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.

7.1.1 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 `StbMLocalTimeHardware` and configured in `GPT_CH_MODE_CONTINUOUS` mode with `GptChannelTickValueMax` as target time. Timer overflows are counted by the notification function `Gpt_Notification` for updating the Virtual Local Time. This timer is not stopped/reconfigured before ECU shutdown.

7.1.2 Initialization

[SWS_StbM_00170] [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.] (*RS_TS_00003*)

[SWS_StbM_00345] [For each Time Base the StbM shall initialize the corresponding event status `NotificationEvents` with 0.] (*RS_TS_00016*)

[SWS_StbM_00344] [For each Time Base the StbM shall initialize the corresponding update counter `timeBaseUpdateCounter` with 0.] (*RS_TS_00011*)

[SWS_StbM_00171] [For each Synchronized Time Base configured

- for a 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.](RS_TS_00004)

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_00306] [If `StbMTimeRecordingSupport` is set to `TRUE`, the StbM shall initialize all Block Elements of the measurement recording tables with zero.](RS_TS_00034)

Note: For details on the measurement recording tables `Synchronized Time Base Record Table` and `Offset Time Base Record Table` refer to chapter 7.3.12.2 “Global Time Precision Measurement Support”

[SWS_StbM_00427] [For each Time Base the StbM shall initialize all of the corresponding User Data bytes with 0.](RS_TS_00015)

7.2 Shutdown behavior

[SWS_StbM_00172] [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.](RS_TS_00024)

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.

[SWS_StbM_00555] [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

](RS_TS_00024)

7.3 Normal operation

7.3.1 Introduction

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.

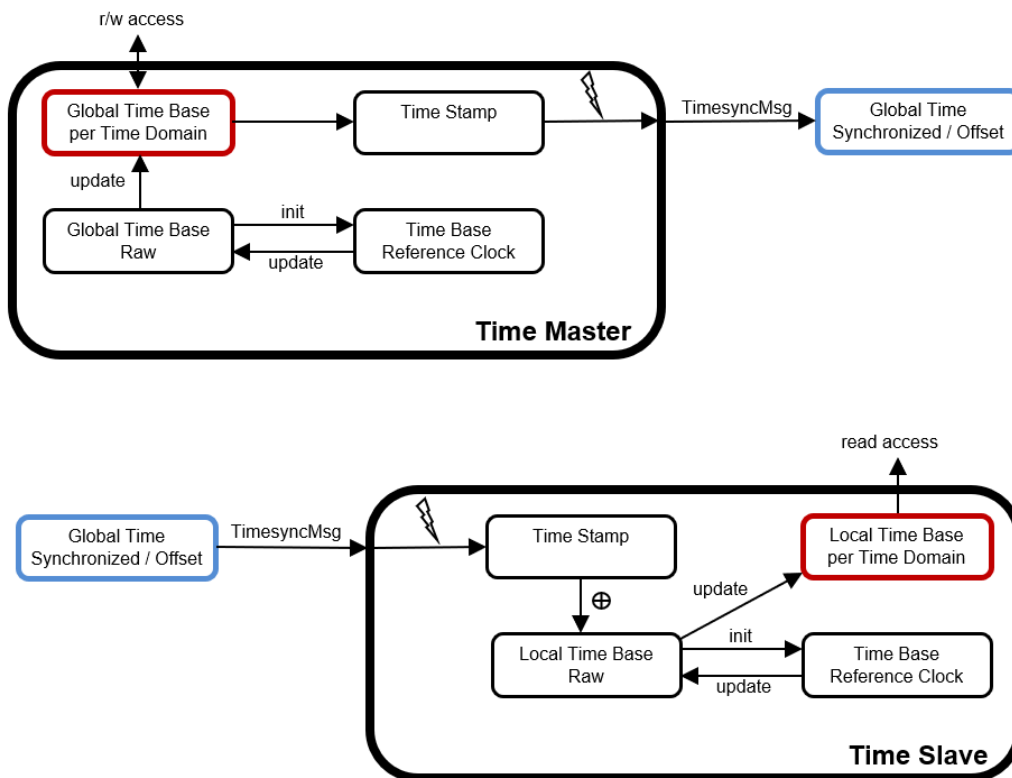


Figure 7.1: Global Time Base Distribution

7.3.1.1 Virtual Local Time

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

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

It is possible to use different Virtual Local Time Bases in parallel.

[SWS_StbM_00512] [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.`] ([RS_TS_00008](#), [RS_TS_00002](#))

[SWS_StbM_00352] [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](#)).] ([RS_TS_00018](#))

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] [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.] ([RS_TS_00008](#))

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] [If [EthIf_GlobalTimeSupport](#) (referenced via [StbMLocalTimeHardware](#), if set to [EthTSynGlobalTimeDomain](#)) is set to TRUE for a Synchronized Time Base, the StbM shall derive the current value of the Virtual Local Time (see [StbM_VirtualLocalTimeType](#)) from the freerunning HW counter from the corresponding Ethernet Controller via [EthIf_GetCurrentTime](#).

If [EthIf_GetCurrentTime](#) returns either [ETH_UNCERTAIN](#) or [ETH_INVALID](#) for parameter [timeQualPtr](#), the time value returned by [EthIf_GetCurrentTime](#) shall be ignored.] ([RS_TS_00008](#), [RS_TS_00002](#))

[SWS_StbM_00539] [If [CanIfGlobalTimeSupport](#) (referenced via [StbMLocalTimeHardware](#), if set to [CanTSynGlobalTimeDomain](#)) is set to TRUE for a Synchronized Time Base, 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.] ([RS_TS_20070](#), [RS_TS_00008](#))

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

[SWS_StbM_00437] [[StbM_GetCurrentVirtualLocalTime](#) shall return the value of the Virtual Local Time of the associated Time Base.

For Offset Time Bases the Virtual Local Time of the referenced Synchronized Time Base shall be returned.

If the Virtual Local Time could not be determined (e.g., the underlying hardware counter has not been activated yet), [StbM_GetCurrentVirtualLocalTime](#) shall return [E_NOT_OK](#).] ([RS_TS_00006](#), [RS_TS_00008](#), [RS_TS_00033](#))

Note: [StbM_GetCurrentVirtualLocalTime](#) is called by the Timesync modules with an established protection against interruptions.

7.3.1.2 Types of Time Bases

7.3.1.2.1 Synchronized and Offset Time Bases

The Time Domains 0 to 15 are Synchronized Time Bases.

The Time Domains 16 to 31 are Offset Time Bases. An Offset Time Base is linked to a Synchronized Time Base only by system wide configuration.

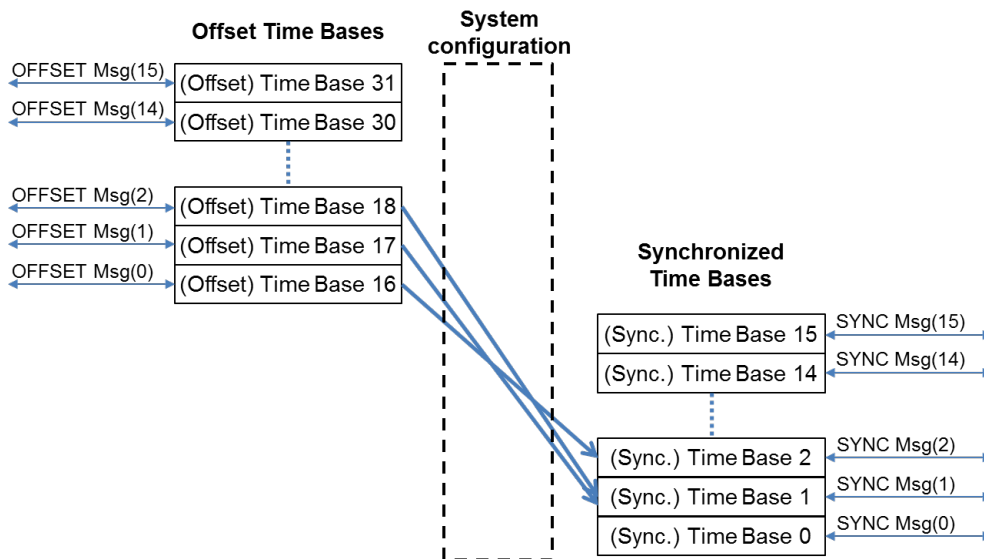


Figure 7.2: Offset Time Base to Synchronized Time Base relationship

Example:

For an Offset Time Base with Time Domain number 17 the OFFSET Timesync messages on CAN and FR always contain $17-16 = 1$ in the Time Domain field. However, the underlying Synchronized Time Base could have Time Domain number 0, i.e., SYNC and FUP Timesync messages contain 0 in the Time Domain field. Another Offset Time Base with Time Domain number 18 (2 in the Time Domain field), may also be based on the underlying Synchronized Time Base 0. An Offset Time Base might have leaps in time, e.g. after GPS time becomes available.

7.3.1.2.2 Pure Local Time Bases

For details of Pure Local Time Bases refer to chapter [7.3.4 "Pure Local Time Bases"](#).

7.3.1.3 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.

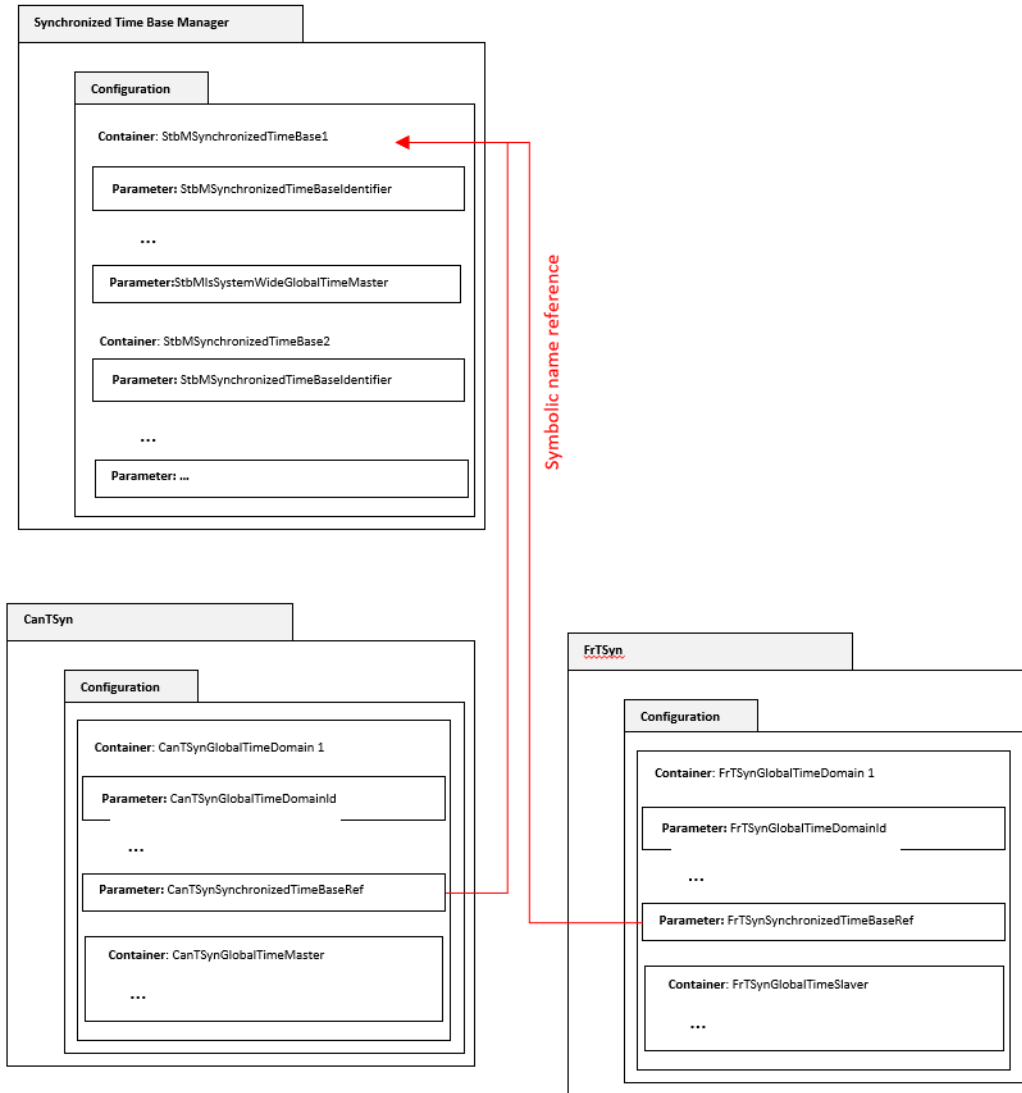


Figure 7.3: Configuration of the StbM Role per Time Base

Example: In [Figure 7.3](#) 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 `FrTSynGlobalTime`

Domain 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]

7.3.1.3.1 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] [`StbM_GetMasterConfig`] shall return the value of the configuration parameter `StbMIsSystemWideGlobalTimeMaster` for the Time Base `time-BaseId`. This is to check, if the StbM is configured as system wide Global Time Master for a specific Time Base. (*RS_TS_00029*)

7.3.1.3.2 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.

7.3.1.3.3 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 (refer to chapter 7.3.1 "Introduction" for the basic mechanisms). 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.

7.3.1.4 Interpolating the Global Time

The Synchronized Time-Base Manager has to interpolate the local instance of the Global Time Base between the updates

- from the Timesync Modules (for a Time Slave) and
- from the application (for a Time Master)

Interpolation is done based on the Virtual Local Time, which is a local time reference derived from some kind of HW counter (refer to [StbMLocalTimeClock](#)). Interpolation is done in principle according to the formula

$$TL = TG_{Sync} + (TV - TV_{Sync}) * r$$

With

- TL: Current value of the local instance of the Global Time
- TG_{Sync} : Global Time value (part of the Main Time Tuple)
- TV: Current value of the Virtual Local Time
- TV_{Sync} : Virtual Local Time value (part of the Main Time Tuple)
- r: optional Rate and Offset-By-Rate correction - if not used set to 1 for Synchronized Time Bases and 0 for Offset Time Bases

TG_{Sync} and TV_{Sync} form the **Main Time Tuple**.

For every Time Base there are multiple Time Tuples but there is only one Time Tuple which is used to interpolate the local instance of the Time Base. This Time Tuple is denoted as the Main Time Tuple.

The precision of a Time Base depends on the handling of the Main Time Tuple:

- 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 Time Master and Time Gateways

Regarding the interpolation by the StbM it is obvious that the precision depends in addition on rounding effects and the granularity of the HW counters.

If requesting a Global Time by the application would always lead to an update of the Main Tuple, the 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.

The Main Time Tuple shall be updated however in accordance with [\[SWS_StbM_00433\]](#), when, e.g.

- a new Global Time or a new Rate Correction value is set by the application
- a new Time Tuple is obtained from a Timesync Module

The Main Time Tuple shall not be updated:

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

Once a new Time Tuple (denoted as **Rx Time Tuple** [$TG_{Rx};TV_{Rx}$]) is obtained from a Timesync Module (i.e., after reception of Timesync message(s)), and which is then

updated to become the Updated Rx Time Tuple, the StbM determines a Time Tuple (denoted as **Synclocal Time Tuple** $[TL_{Sync}; TV_{Sync}]$) of the local instance of the Global Time by using the Virtual Local Time of the Updated Rx Time Tuple as reference (i.e., $TV = TV_{URx}$). Refer to [Figure 7.4](#) for an example.

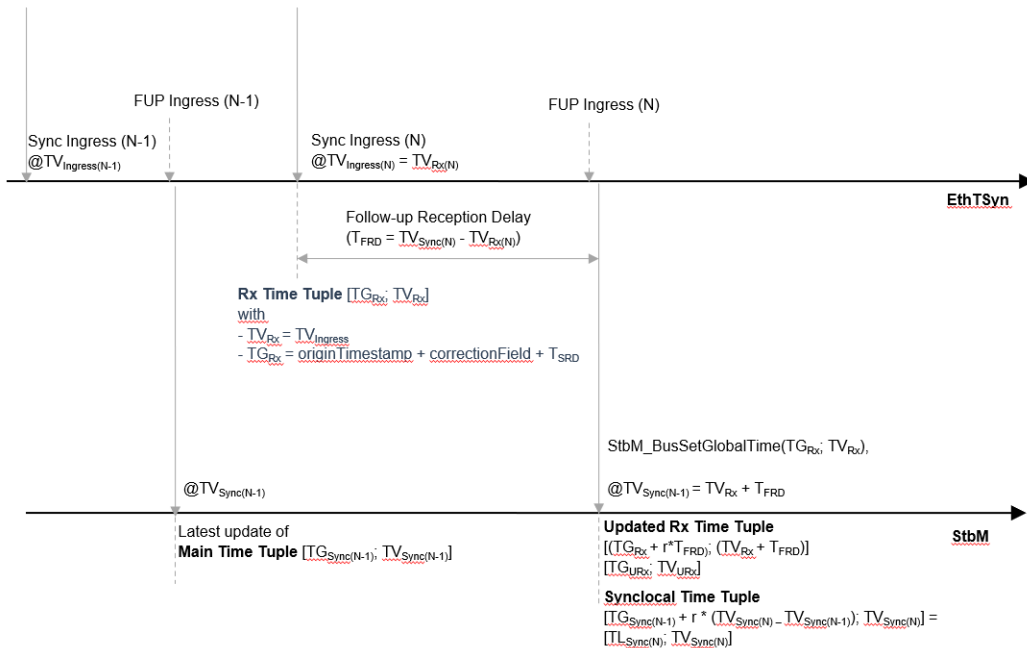


Figure 7.4: Example for Rx Time Tuple (sent by EthTSyn) processing

In case of actually performing Offset Correction By Rate Adaption (i.e., the mechanism is enabled and the prerequisites are fulfilled), the Main Time Tuple is not overwritten by the Updated Rx Time Tuple, instead the Main Time Tuple is overwritten by the Synclocal Time Tuple of the local instance of the Global Time.

Otherwise, the Main Time Tuple is overwritten by the Updated Rx Time Tuple.

The Main Time Tuple can be updated if a certain time has elapsed since the last update (refer to [[SWS_StbM_00433](#)]).

The Main Time Tuple $[TG_{Sync}; TV_{Sync}]$ is managed by the StbM. Each time TG_{Sync} is updated, TV_{Sync} has to be updated as well and vice versa.

Below the application, in the BSW, the Time of a Time Base is always managed via the Time Tuple structure:

- Timesync Modules provide the received Global Time in form of a Time Tuple to the StbM
- Timesync Modules obtain the Global Time to transmit as a Time Tuple
- A Global Time value set by the application is immediately extended to a Time Tuple by adding the current value of the Virtual Local Time

It is essential to always adhere to the integrity of the Time Tuple.

[SWS_StbM_00433] [The Main Time Tuple shall only be updated

- after setting a new Global Time or a new Rate Correction value by the application
- after obtaining a Rx Time Tuple (i.e., a new Time Tuple) from a Timesync Module
- after the Offset Correction By Rate Adaption interval (see [SWS_StbM_00353])

However, the Main Time Tuple may be updated if there has been no update for more than 3s.] ([RS_TS_00008](#), [RS_TS_00002](#))

Note: 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.

7.3.2 Synchronized Time Bases

[SWS_StbM_00180] [After initialization the StbM shall maintain the Local Time of each Time Base autonomously via a hardware reference clock (referenced by `StbMLocalTimeClock`).] ([RS_TS_00008](#), [RS_TS_00002](#))

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] [For Time Bases 0 to 15 `StbM_GetCurrentTime` shall return for the requested Time Base the current Time Tuple [TL; TV], the related Status and the User Data, where TL is the current value of the local instance of the Time Base as calculated by [SWS_StbM_00355].] ([RS_TS_00005](#), [RS_TS_00006](#), [RS_TS_00014](#))

[SWS_StbM_00434] [For Time Domains 0 to 15 `StbM_GetCurrentTime` shall return `E_NOT_OK` if the value of the associated Virtual Local Time could not be retrieved.] ([RS_TS_00005](#), [RS_TS_00006](#), [RS_TS_00014](#))

Note: Retrieving a Virtual Local Time value may fail for several reasons, e.g., if the related hardware counter was not yet activated.

[SWS_StbM_00436] [Although `StbM_GetCurrentTime` calculates and returns a new Time Tuple [TL; TV], this tuple shall only replace the Main Time Tuple if the requirements as specified in [SWS_StbM_00433] are met.] ([RS_TS_00005](#), [RS_TS_00006](#), [RS_TS_00014](#))

Note: Prohibiting the update of the Main Time Tuple after e.g. every invocation of `StbM_GetCurrentTime` prevents worsening the precision of the requested Time Base due to rounding errors.

7.3.2.1 Global Time Master

[SWS_StbM_00342] [On a valid invocation of `StbM_SetGlobalTime` or `StbM_UpdateGlobalTime` the StbM shall update the Main Time Tuple of the corresponding Synchronized Time Base.

Within the functions `StbM_SetGlobalTime` and `StbM_UpdateGlobalTime` the StbM shall retrieve the value of the Virtual Local Time (as part of the Local Time tuple) **as soon as possible** in order to improve precision of the Time Base. (RS_TS_00010, RS_TS_00002)

Note: In order to improve precision further it may be beneficial for applications to call `StbM_SetGlobalTime` or `StbM_UpdateGlobalTime` with locked interrupts.

[SWS_StbM_00516] [On invocation of `StbM_SetBusProtocolParam` for a Time Master of Time Bases 0 to 15, 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`. (RS_TS_20069)

7.3.2.2 Time Slave

[SWS_StbM_00529] [For Time Bases 0 to 15 each invocation of `StbM_BusSetGlobalTime` shall update the Rx Time Tuple $[TG_{Rx}; TV_{Rx}]$ as follows:

1. Retrieve the current Virtual Local Time value as TV_{Sync}
2. Calculate the time interval $T_{FRD} = TV_{Rx} - TV_{Sync}$ based on the Local Virtual Time value of the Rx Time Tuple (TV_{Rx}) as provided by the member `virtualLocalTime` of input parameter `timeTuplePtr`.
3. Apply the current rate value of the Synchronized Time Base $r = r_{rc}$ to the time interval T_{FRD}
4. Add the rate corrected interval $r \cdot T_{FRD}$ to the Global Time of the Rx Time Tuple (TG_{Rx}) as provided by member the `globalTime` of input parameter `timeTuplePtr`

The resulting Time Tuple $[TG_{Rx} + r \cdot T_{FRD}; TV_{Sync}]$ is denoted as Updated Rx Time Tuple $[TG_{URx}; TV_{URx}]$ of the Synchronized Time Base.] ([RS_TS_00007](#), [RS_TS_00009](#))

Note: [Figure 7.4](#) illustrates the sequence of actions how a Time Slave calculates the Updated Rx Time Tuple $[TG_{URx}; TV_{URx}]$.

[SWS_StbM_00179] [For Time Bases 0 to 15 each invocation of `StbM_BusSetGlobalTime` shall update the corresponding Main Time Tuple and set the User Data and the Time Base Status accordingly.] ([RS_TS_00007](#), [RS_TS_00009](#))

Note: To update the Main Time Tuple does not mean to automatically overwrite the Main Time Tuple with the Updated Rx Time Tuple.

[SWS_StbM_00438] [The StbM shall determine for Time Bases 0 to 15 on each invocation of `StbM_BusSetGlobalTime` the Synclocal Time Tuple $[TL_{Sync}; TV_{Sync}]$ by using the value of the Virtual Local Time of the Updated Rx Time Tuple as reference (i.e., TV_{Rx} is used for TV when calculating TL in [\[SWS_StbM_00355\]](#)). The Synclocal Time Tuple shall be determined using the Main Time Tuple before the Main Time Tuple itself is updated.] ([RS_TS_00007](#), [RS_TS_00009](#))

[SWS_StbM_00517] [On invocation of `StbM_GetBusProtocolParam` for Time Bases 0 to 15, 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`.] ([RS_TS_20069](#))

7.3.3 Offset Time Bases

An Offset Time Base only exists in combination with its underlying Synchronized Time Base.

The **Absolute Time** value of an Offset Time Base is given by adding the **Offset Time** value of an Offset Time Base to the time value of the underlying Synchronized Time Base.

[SWS_StbM_00191] [`StbM_SetOffset` and `StbM_GetOffset` shall only accept Offset Time Bases with a `timeBaseId` 16 to 31.] ([RS_TS_00012](#), [RS_TS_00013](#))

[SWS_StbM_00177] [For Time Bases 16 to 31 `StbM_GetCurrentTime` shall return for the requested Time Base the current Time Tuple $[TL_{abs}, TV]$, the related Status and the User Data, where TL_{abs} is the absolute time value calculated by adding the Offset

Value of the Offset Time Base to the current value TL of the underlying Synchronized Time Base referenced via `StbMOffsetTimeBase`.](RS_TS_00013)

7.3.3.1 Global Time Master

[SWS_StbM_00190] [Each valid invocation of `StbM_SetOffset` shall update the Main Time Tuple of the corresponding Offset Time Base. The Offset Time value and the User Data shall be set accordingly.](RS_TS_00013, RS_TS_00015)

[SWS_StbM_00192] [Each invocation of `StbM_GetOffset` shall return the Offset Time value and the User Data of the corresponding Offset Time Base.](RS_TS_00013, RS_TS_00014)

[SWS_StbM_00304] [On invocation of `StbM_SetGlobalTime` or `StbM_UpdateGlobalTime` for Time Bases 16 to 31 the StbM shall check the `GLOBAL_TIME_BASE` bit within `timeBaseStatus` of the underlying Synchronized Time Base and shall return `E_NOT_OK` if it is not set.

If the `GLOBAL_TIME_BASE` bit is set:

1. the StbM shall calculate the Offset Time by obtaining the actual Time Base value of the underlying Synchronized Time Base and subtract that from the Absolute Time value which is passed by `StbM_SetGlobalTime` or `StbM_UpdateGlobalTime`
2. (a) if the calculated Offset Time value is equal or greater than zero, the StbM shall update the corresponding Offset Time Base with the calculated Offset Time value and the User Data that was passed by `StbM_SetGlobalTime` or `StbM_UpdateGlobalTime`,
(b) otherwise (calculated Offset Time value is less than zero) the StbM shall return `E_NOT_OK` via `StbM_SetGlobalTime` or `StbM_UpdateGlobalTime`, respectively.

](RS_TS_00013)

7.3.3.2 Time Slave

[SWS_StbM_00528] [For Time Bases 16 to 31 each invocation of `StbM_BusSetGlobalTime` shall update the Rx Time Tuple $[T_{Rx}; TV_{Rx}]$ as follows:

1. Retrieve the current Virtual Local Time value as TV_{Sync}
2. Calculate the time interval $T_{FRD} = TV_{Rx} - TV_{Sync}$ based on the Local Virtual Time value of the Rx Time Tuple (TV_{Rx}) as provided by the member `virtualLocalTime` of the input parameter `timeTuplePtr`

3. Apply the current rate value of the Offset Time Base $r = (r_{orc} - 1)$ to the time interval T_{FRD}
4. Add the rate corrected interval $r \cdot T_{FRD}$ to the Global Time of the Rx Time Tuple (TG_{Rx}) as provided by the member `globalTime` of the input parameter `time-TuplePtr`

The resulting Time Tuple $[TG_{Rx} + r \cdot T_{FRD}; TV_{Sync}]$ is denoted as Updated Rx Time Tuple $[TG_{URx}; TV_{URx}]$ of the Offset Time Base. *(RS_TS_00007, RS_TS_00009)*

Note: Figure 7.4 illustrates the sequence of actions how a Time Slave calculates the Updated Rx Time Tuple $[TG_{URx}; TV_{URx}]$.

Note: The calculation of the Updated Rx Time Tuple $[TG_{URx}; TV_{URx}]$ ensures, that the delay between the ingress of the SYNC/OFS Message and the actual processing in the StbM is rate corrected. Otherwise, precision could be significantly impaired.

[SWS_StbM_00393] [For Time Bases 16 to 31 each invocation of `StbM_BusSetGlobalTime` shall update the corresponding Main Time Tuple and set the User Data and the Time Base Status accordingly.] *(RS_TS_00007, RS_TS_00009)*

Note: To update the Main Time Tuple does not mean to automatically overwrite the Main Time Tuple with the Updated Rx Time Tuple.

[SWS_StbM_00439] [The StbM shall determine for Time Bases 16 to 31 on each invocation of `StbM_BusSetGlobalTime` the Synclocal Time Tuple $[TL_{Sync}; TV_{Sync}]$ by using the value of the Virtual Local Time of the Updated Rx Time Tuple as reference (i.e., TV_{Rx} is used for TV when calculating TL in *[SWS_StbM_00355]*). The Synclocal Time Tuple shall be determined using the Main Time Tuple before the Main Time Tuple is updated.] *(RS_TS_00007, RS_TS_00009)*

7.3.4 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. Pure Local Time Bases behaving like an Offset Time Bases are not supported.

[SWS_StbM_00413] [After initialization the StbM shall maintain the Time of each Pure Local Time Base autonomously via a hardware reference clock (referenced by `StbM_LocalTimeClock`).] *(RS_TS_00008, RS_TS_00002)*

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] [For Pure Local Time Bases `StbM_GetCurrentTime` shall return the User Data as set by `StbM_SetGlobalTime`, `StbM_UpdateGlobalTime` or `StbM_SetUserData` by the Pure Local Time Master.] (*RS_TS_00015*)

[SWS_StbM_00399] [For Pure Local Time Bases all bits of the Time Base status `timeBaseStatus` shall be set to 0, except for bit `GLOBAL_TIME_BASE`.

`GLOBAL_TIME_BASE` shall be set to 1, by a valid invocation of `StbM_SetGlobalTime` or `StbM_UpdateGlobalTime` and only set to 0 by `StbM_Init`.] (*RS_TS_00009*)

7.3.5 Synchronization State

[SWS_StbM_00261] [For Offset Time Bases `StbM_GetCurrentTime` shall derive the status `timeBaseStatus` to be returned with the actual time value as follows from the status of the actual Offset Time Base and the Synchronized Time Base (referenced via parameter `StbMOffsetTimeBase`):

Bit Name	Bit Position	Description
<code>TIMEOUT</code>	Bit 0 (LSB)	0: No Timeout occurred - neither for Offset nor for referenced Synchronized Time Base 1: Timeout occurred for Offset or for referenced Synchronized Time Base
Reserved	Bit 1	Always 0 (reserved for future usage)
<code>SYNC_TO_GATEWAY</code>	Bit 2	0: Local Offset and referenced Synchronized Time Base is synchronous to Global Offset Time Master 1: Local Offset or referenced Synchronized Time Base updates are based on a Time Gateway below the Global Time Master
<code>GLOBAL_TIME_BASE</code>	Bit 3	0: Local Offset or referenced Synchronized Time Base are based on Local Time Base reference clock only (never synchronized with Global Time Base) 1: Local Offset and referenced Synchronized Time Base have been synchronized with Global Time Base at least once
<code>TIMELEAP_FUTURE</code>	Bit 4	0: No leap into the future within the received time for the Offset and referenced Synchronized Time Base 1: Leap into the future within the received time for the Offset or referenced Synchronized Time Base exceeds a configured threshold
<code>TIMELEAP_PAST</code>	Bit 5	0: No leap into the past within the received time for the Offset and referenced Synchronized Time Base 1: Leap into the past within the received time for the Offset or referenced Synchronized Time Base exceeds a configured threshold

Table 7.1: Synchronization States

] (*RS_TS_00005*)

[SWS_StbM_00262] [For Synchronized Time Bases `StbM_GetTimeBaseStatus` shall return

- the status of the corresponding Synchronized Time Base via `syncTimeBaseStatus` and
- 0 via `offsetTimeBaseStatus`

For Offset Time Bases `StbM_GetTimeBaseStatus` shall return

- the status of the corresponding Offset Time Base via `offsetTimeBaseStatus` and
- the status of the related Synchronized Time Base (referenced by `StbMOffsetTimeBase`) via `syncTimeBaseStatus`.

For Pure Local Time Bases `StbM_GetTimeBaseStatus` shall return

- the status of the corresponding Time Base (refer to [SWS_StbM_00399]) via `syncTimeBaseStatus` and
- 0 via `offsetTimeBaseStatus`

]([RS_TS_00005](#), [RS_TS_00021](#))

7.3.5.1 Global Time Master

[SWS_StbM_00181] [On a valid invocation of `StbM_SetGlobalTime`, `StbM_UpdateGlobalTime`, or `StbM_SetOffset` the StbM shall set the `GLOBAL_TIME_BASE` bit within `timeBaseStatus` of the corresponding Time Base and shall clear all other bits.]([RS_TS_00009](#))

7.3.5.2 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.

[SWS_StbM_00182] [For Synchronized and Offset 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 Received 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., $TG_{Rx} - TL_{Sync} > StbMTimeLeapFutureThreshold$, if at least one Time Base value has been successfully received before.

With:

- TL_{Sync} = Global Time part of the Synclocal Time Tuple

- TG_{Rx} = 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 `StbM-TimeLeapFutureThreshold` the StbM shall clear the `TIMELEAP_FUTURE` bit within `timeBaseStatus` of the Time Base.

A threshold of 0 shall deactivate this check.] (*RS_TS_00009*)

[SWS_StbM_00305] [For Synchronized and Offset 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., $TL_{Sync} - TG_{Rx} > StbMTimeLeapPastThreshold$, if at least one Time Base value has been successfully received before.

With:

- TL_{Sync} = Global Time part of the Synclocal Time Tuple
- TG_{Rx} = 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 `StbM-TimeLeapPastThreshold` the StbM shall clear the `TIMELEAP_PAST` bit within `timeBaseStatus` of the Time Base.

A threshold of 0 shall deactivate this check.] (*RS_TS_00009*)

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 Slaves 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/StbM-TimeLeapPastThreshold` have not yet happened.

[SWS_StbM_00425] [For Synchronized and Offset 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 Received Time and the Synclocal Time, i.e., $TG_{Rx} - TL_{Sync}$, 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_{Rx} = 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. *(RS_TS_00009)*

[SWS_StbM_00183] [For Synchronized and Offset 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. *(RS_TS_00025, RS_TS_00009)*

[SWS_StbM_00540] [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

(RS_TS_00025, RS_TS_00009)

[SWS_StbM_00187] [For Synchronized and Offset 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]**). *(RS_TS_00025, RS_TS_00009)*

[SWS_StbM_00184] [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`. *(RS_TS_00009)*

[SWS_StbM_00185] [For Synchronized and Offset 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. *(RS_TS_00009)*

7.3.6 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] [`StbM_GetTimeBaseUpdateCounter` shall return the value of the `timeBaseUpdateCounter` of the corresponding Time Base.](*RS_TS_00011*)

[SWS_StbM_00351] [For Synchronized and Offset Time Bases, the `timeBaseUpdateCounter` of a Time Base shall have the value range 0 to 255.](*RS_TS_00011*)

[SWS_StbM_00350] [

- For Synchronized and Offset Time Bases on a valid invocation of `StbM_SetGlobalTime`, `StbM_BusSetGlobalTime`, or `StbM_TriggerTimeTransmission` and
- for Offset Time Bases on a valid invocation of `StbM_SetOffset`,

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

At 255 the `timeBaseUpdateCounter` shall wrap around to 0.](*RS_TS_00011*)

Note: For Offset Time Bases the term "corresponding Time Base" refers to the Offset Time Base only and not to the underlying Synchronized Time Base.

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

7.3.7 User Data

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

User Data can be used to characterize the Time Base, e.g., regarding the quality of the underlying clock source or regarding the progress of time.

User Data consists 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] [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.] (*RS_TS_00015*)

7.3.8 Time Correction

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

- Rate Correction and
- Offset Correction.

of their Time Base(s).

For Global Time Masters the StbM provides the ability to perform

- Rate Correction

of their Time Base(s).

Time Correction can be configured individually for each Time Base.

7.3.8.1 Rate Correction Measurement (for Time Slaves)

Rate Correction detects and eliminates rate deviations of local instances of Time Bases. Rate Correction determines the rate deviation in the scope of a measurement. This rate deviation is used as correction factor which the StbM uses to correct the Time Base's time whenever it is determined (e.g., in the scope of `StbM_GetCurrentTime`).

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

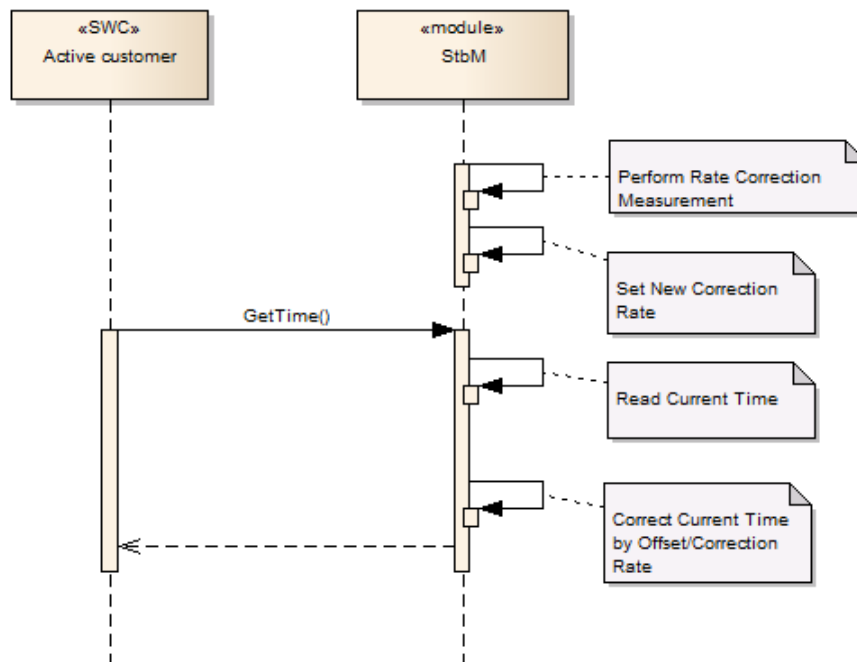


Figure 7.5: Rate Correction

[SWS_StbM_00377] [The StbM shall not perform Rate Correction when the measurement duration `StbMRateCorrectionMeasurementDuration` is set to 0.] ([RS_TS_00018](#))

[SWS_StbM_00376] [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.] ([RS_TS_00018](#))

[SWS_StbM_00375] [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.] ([RS_TS_00018](#))

[SWS_StbM_00374] [For Rate Correction measurements, the StbM shall evaluate the `TIMEOUT` flag. The StbM shall discard the measurement, if the flag is set.] ([RS_TS_00018](#))

[SWS_StbM_00373] [For Rate Correction, the StbM shall evaluate the `TIMELEAP_FUTURE/ TIMELEAP_PAST` flags at the start of a measurement. The StbM shall not start a Rate Correction measurement, when any of these flags is set.] ([RS_TS_00018](#))

[SWS_StbM_00372] [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.] ([RS_TS_00018](#))

[SWS_StbM_00371] [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 or Offset Time Bases.] ([RS_TS_00018](#))

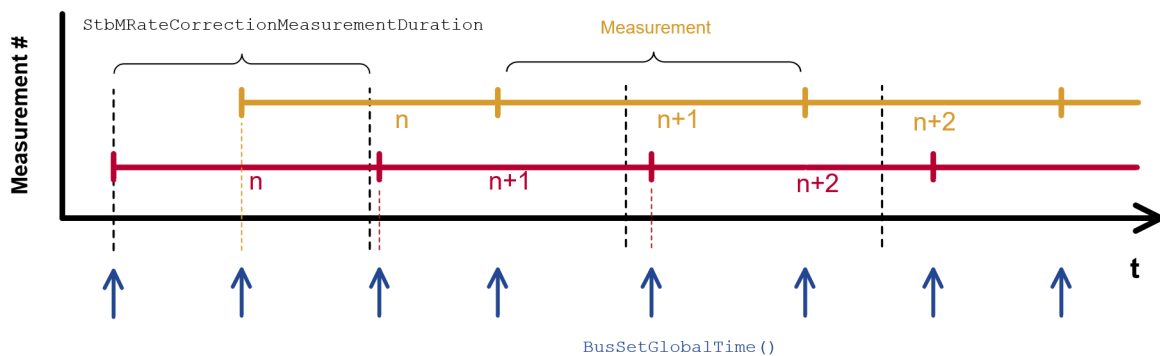


Figure 7.6: Visualization of two parallel measurements

[SWS_StbM_00370] [During runtime the StbM shall determine the timespan of a Rate Correction measurement on the basis of the Virtual Local Time.] ([RS_TS_00018](#))

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 `StbM_MainFunctionPeriod` 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] [The StbM shall perform as many simultaneous Rate Correction measurements as configured by parameter `StbMRateCorrectionsPerMeasurementDuration` for each configured Time Base.] ([RS_TS_00018](#))

[SWS_StbM_00367] [Simultaneous Rate Correction measurements shall be started with a defined offset (to_n) to yield Rate Corrections evenly distributed over the measurement duration:

- $to_n = n * (StbMRateCorrectionMeasurementDuration / StbMRateCorrectionsPerMeasurementDuration)$

with

- n : zero-based index of the current measurement

](RS_TS_00018)

Note: If a Rate Correction measurement start is delayed e.g. due to a late reception of time values for Synchronized or Offset 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 or Offset 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] [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

](RS_TS_00018)

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

[SWS_StbM_00364] [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

](RS_TS_00018)

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

[SWS_StbM_00361] [At the end of a Rate Correction measurement, the StbM shall calculate the resulting correction rate (r_{rc}) for Synchronized Time Bases as shown:

- $r_{rc} = (TG_{Stop} - TG_{Start}) / (TV_{Stop} - TV_{Start})$

](RS_TS_00018)

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

[SWS_StbM_00362] [The StbM shall use the same value for r_{rc} and r_{orc} until a new value has been calculated.](RS_TS_00018)

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

[SWS_StbM_00360] [At the end of a Rate Correction measurement, the StbM shall calculate the resulting correction rate (r_{orc}) for Offset Time Bases as shown:

- $r_{orc} = (TG_{Stop} - TG_{Start}) / (TV_{Stop} - TV_{Start}) + 1$

](RS_TS_00018)

Note: TG_{Stop} and TG_{Start} refer to the Offset value of the Offset Time Base at the end and start of the measurement respectively, not to the corresponding absolute values of the Offset Time Base (i.e., Offset value + value of underlying Synchronized Time Base). Since the Offset value is almost constant over time, $TG_{Stop} - TG_{Start}$ is close to 0 for Offset Time Bases.

The rate for Offset Time Bases (r_{orc}) would therefore be close to 0, while r_{rc} for Synchronized Time Bases is close to 1 (refer to [SWS_StbM_00361]), if +1 was not added. By adding +1 in the formula for r_{orc} value ranges for rate correction r_{orc} and r_{rc} and the corresponding rate deviation values can be aligned which allows for more generic expressions, e.g. in [SWS_StbM_00355].

However, when doing the actual rate correction for Offset Time Bases the extra +1 needs to be ignored (refer to [SWS_StbM_00441] and [SWS_StbM_00424])

[SWS_StbM_00527] [The StbM shall calculate the rate deviation r_{Dev} as

- $r_{rc} - 1$ for Synchronized Time Bases and
- $r_{orc} - 1$ for Offset Time Bases.

](RS_TS_00018)

[SWS_StbM_00397] [For Time Bases with `StbMSynchronizedTimeBaseIdentifier` 0 to 31 and `StbMIsSystemWideGlobalTimeMaster = 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`.](RS_TS_00018)

[SWS_StbM_00412] [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.

For an Offset Time Base the StbM shall use $r_{orc} = 1$, if a valid correction rate (r_{orc}) has not yet been calculated or is not being calculated (refer [SWS_StbM_00377]) but shall be applied. (RS_TS_00018)

7.3.8.2 Time Interpolation, Rate and Offset Correction (for Time Slaves)

Time interpolation happens whenever the current value of the local instance of a Time Base shall be determined. The calculation is based on the Main Time Tuple.

If Rate Correction is enabled for a given Time Base the calculation includes the Calculated Rate Correction value (r_{rc} for Synchronized Time Bases, r_{orc} for Offset Time Bases).

Whenever a new Global Time Tuple is received, there is a difference between the received Global Time and the Global Time of the Synclocal Time Tuple (see [SWS_StbM_00438], [SWS_StbM_00439]). This difference is denoted as offset.

Offset Correction can be done in two ways:

- **Offset Correction By Jump:**

the Main Time Tuple is overwritten by the Updated Rx Time Tuple, i.e., the time of the local instance of the Time Base jumps to the value of the received Global Time (refer to Figure 7.7).

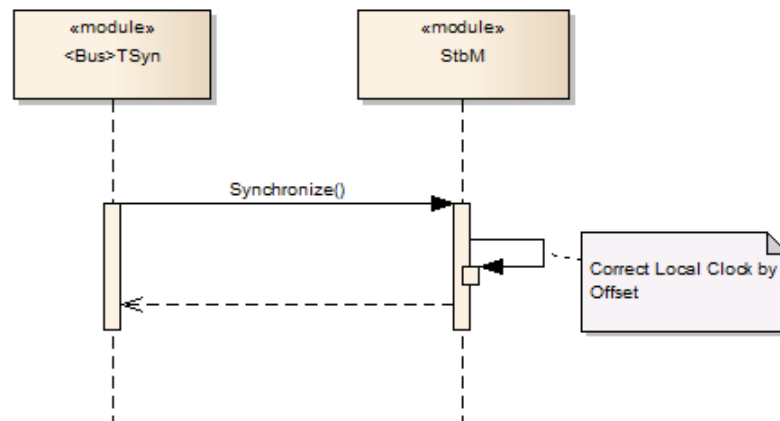


Figure 7.7: Offset Jump Correction

- **Offset Correction By Rate Adaption:**

the Main Time Tuple is not overwritten by the Updated Rx Time Tuple, instead the applied Rate Correction is adapted such that the existing offset is steadily reduced to zero within a configured time span. Offset Correction By Rate Adaption

can only be applied if Rate Correction is enabled, of course (refer to [Figure 7.8](#)).

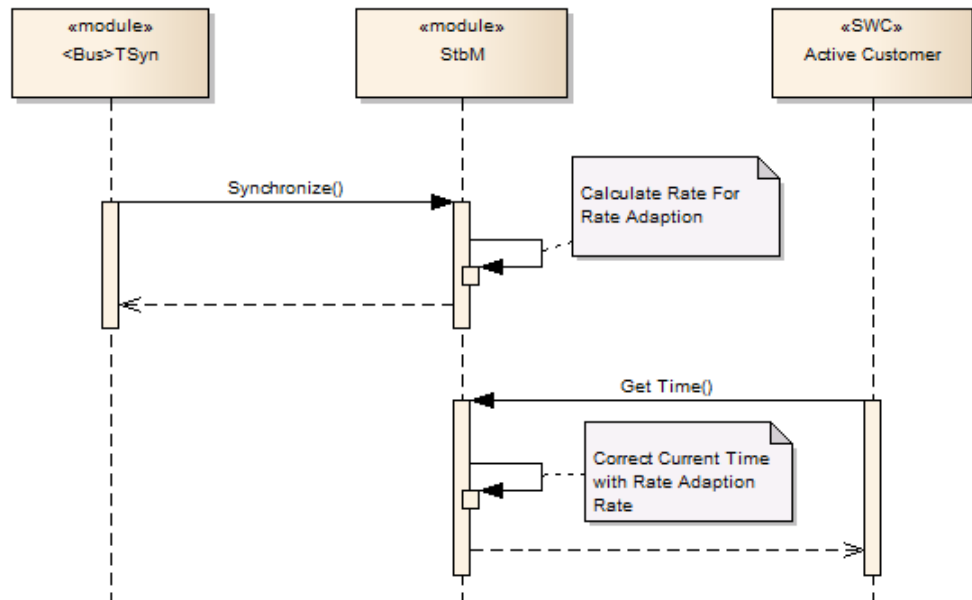


Figure 7.8: Offset Rate Adaption

[SWS_StbM_00359] [The StbM shall calculate the Global Time offset (i.e., difference) between the Received Time and the Synclocal Time upon each, except the very first, valid invocation of `StbM_BusSetGlobalTime`. The elements of the Time Tuples used for calculating the Global Time offset have the following denotation:

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

]([RS_TS_00018](#))

[SWS_StbM_00355] [The StbM shall calculate the current value of a Time Base based on the Main Tuple and the current rate correction term according to:

- $TL = TG_{Sync} + (TV - TV_{Sync}) * r$

With:

- TL = Current value of the Time Base
- TV = Current value of the Virtual Local Time
- TV_{Sync} = Virtual Local Time part of the Main Time Tuple
- TG_{Sync} = Global Time part of the Main Time Tuple
- r = Current rate for correcting the local instance of the Time Base

]([RS_TS_00018](#))

Note: For Offset Time Bases TG_{Sync} and TL refer to the Offset values of the Offset Time Base respectively, not to the corresponding absolute values of the Offset Time Base (i.e., Offset value + value of underlying Synchronized Time Base)

[SWS_StbM_00440] [For Synchronized Time Bases and if rate correction is enabled (see [SWS_StbM_00377]) and if the absolute value of the time offset between the Received Time and the Synclocal Time ($abs(TG_{Rx} - TL_{Sync})$) is equal or greater than `StbMOffsetCorrectionJumpThreshold`, the StbM shall use the factor r_{rc} for the rate correction term r :

- $r = r_{rc}$

Otherwise r shall be set to 1, unless r shall be set accordingly to [SWS_StbM_00356] or [SWS_StbM_00353].] (*RS_TS_00018*)

[SWS_StbM_00441] [For Offset Time Bases and if rate correction is enabled (see [SWS_StbM_00377]) and if the absolute value of the time offset between the Received Time and the Synclocal Time ($abs(TG_{Rx} - TL_{Sync})$) is equal or greater than `StbMOffsetCorrectionJumpThreshold`, the StbM shall use the factor r_{orc} for the rate correction term r :

- $r = r_{orc} - 1$

Otherwise r shall be set to 0, unless r shall be set accordingly to [SWS_StbM_00356] or [SWS_StbM_00353].] (*RS_TS_00018*)

[SWS_StbM_00356] [If rate correction is enabled (see [SWS_StbM_00377]) and if the absolute value of the time offset between the Received Time and the Synclocal Time ($abs(TG_{Rx} - TL_{Sync})$) is smaller than `StbMOffsetCorrectionJumpThreshold`, the StbM shall correct the time offset by temporarily applying an additional rate (r_{oc}) to r :

- for Synchronized Time Bases: $r = r_{rc} + r_{oc}$
- for Offset Time Bases: $r = (r_{orc} - 1) + r_{oc}$

This rate correction term shall be applied for the duration defined by parameter `StbMOffsetCorrectionAdaptionInterval`, starting when obtaining the Updated Rx Time Tuple (i.e., it shall be applied as long as $(TV - TV_{Sync})$ is smaller than `StbMOffsetCorrectionAdaptionInterval`).

r_{oc} shall be calculated as:

- $r_{oc} = (TG_{Rx} - TL_{Sync}) / (T_{CorrInt})$

With:

- $T_{CorrInt} = \text{StbMOffsetCorrectionAdaptionInterval}$
- $TL_{Sync} =$ Global Time part of the Synclocal Time Tuple
- $TG_{Rx} =$ Global Time part of the Updated Time Tuple

] (*RS_TS_00018, RS_TS_00019*)

[SWS_StbM_00353] [If an additional rate has been applied (Offset Correction By Rate Adaption according to [\[SWS_StbM_00356\]](#)), the StbM shall after the period of [StbMOffsetCorrectionAdaptionInterval](#) (i.e., $(TV - TV_{Sync})$ (see [\[SWS_StbM_00355\]](#)) is larger or equal than [StbMOffsetCorrectionAdaptionInterval](#) insert the following two steps if it needs to calculate the current value of a Time Base as defined by [\[SWS_StbM_00355\]](#):

- It shall first calculate a temporary Time Tuple $[TL_{Temp}; TV_{Temp}]$ using the formula in [\[SWS_StbM_00355\]](#) with
 - $TV = TV_{Temp} = TV_{Sync} + \text{StbMOffsetCorrectionAdaptionInterval}$
 - r set to
 - * for Synchronized Time Bases: $r = r_{rc} + r_{oc}$
 - * for Offset Time Bases: $r = (r_{orc} - 1) + r_{oc}$
 - TL_{Temp} shall be set to the resulting value TL
- Afterwards the Main Time Tuple $[TG_{Sync}; TV_{Sync}]$ shall be set by an atomic operation to the values of the temporary Time Tuple $[TL_{Temp}; TV_{Temp}]$.

Then, the calculation in [\[SWS_StbM_00355\]](#) shall be done by using the updated Main Time Tuple, the current value of the Virtual Local Time and $r = r_{rc}$ or, respectively, $r = (r_{orc} - 1)$.] ([RS_TS_00018](#))

Note: It is possible for the StbM to perform the first two steps (i.e., to update the Main Time Tuple) in its Main Function after expiration of [StbMOffsetCorrectionAdaptionInterval](#) without being requested to calculate the current time. However, since a request to calculate the current time might occur after expiration of [StbMOffsetCorrectionAdaptionInterval](#) but before the next Main Function invocation, it is not possible to always decouple the first two steps from the last one.

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

[SWS_StbM_00400] [If [StbMOffsetCorrectionJumpThreshold](#) is set to 0, Offset Correction shall be performed by Jump Correction only.] ([RS_TS_00018](#))

7.3.8.3 Time Interpolation and Rate Correction for Global Time Masters

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

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](#)).

The interpolation of the Time Base is based on the Main Time Tuple, the current value of the Virtual Local Time and the current Rate Correction value.

[SWS_StbM_00395] [If `StbMAllowMasterRateCorrection` equals `TRUE`, an invocation of `StbM_SetRateCorrection` shall set the rate correction value. Otherwise `StbM_SetRateCorrection` shall do nothing and return `E_NOT_OK`.] (*RS_TS_00018*)

[SWS_StbM_00411] [The StbM shall apply rate correction to a Time Base, if `StbMAllowMasterRateCorrection` equals `True` and a valid rate correction value has been set by `StbM_SetRateCorrection`.] (*RS_TS_00018*)

[SWS_StbM_00396] [If the absolute value of the rate correction parameter `rateDeviation`, which is passed to `StbM_SetRateCorrection`, is greater than `StbMMasterRateDeviationMax`, `StbM_SetRateCorrection` shall set the actually applied rate correction value to either (`StbMMasterRateDeviationMax`) or (`-StbMMasterRateDeviationMax`)(depending on sign of `rateDeviation`).] (*RS_TS_00018*)

Note: The actual applied resulting rate correction r will be

- for Synchronized Time Bases: $r = \text{rateDeviation} + 1$ (= r_{rc} as given in [*SWS_StbM_00424*])
- for Offset Time Bases: $r = \text{rateDeviation}$ (= $r_{orc} - 1$ as given in [*SWS_StbM_00424*])

If aligning the rate of one Time Base to the rate of another one, it is possible to use `StbM_GetRateDeviation` and pass the value as argument to `StbM_SetRateCorrection`.

[SWS_StbM_00424] [The StbM shall calculate the (rate corrected) time (TL) of its local instance of the Time Base as:

$$TL = TG_{Sync} + (TV - TV_{Sync}) * r$$

With:

- TV = Current value of the Virtual Local Time
- TV_{Sync} = Virtual Local Time part of the Main Time Tuple
- TG_{Sync} = Global Time part of the Main Time Tuple
- r = Rate for correcting the Time Base

with

- for Synchronized Time Bases: $r = r_{rc}$
- for Offset Time Bases: $r = r_{orc} - 1$

If `StbMAllowMasterRateCorrection` equals `FALSE`, r shall be set to

- 1 for Synchronized Time Bases

- 0 for Offset Time Bases

(i.e., no rate correction is applied).] ([RS_TS_00018](#))

Note: TL and TV form a new temporary Time Tuple.

Note: For Offset Time Bases TG_{Sync} and TL refer to the Offset values of the Offset Time Base respectively, **not** to the corresponding absolute values of the Offset Time Base (i.e., Offset value + value of underlying Synchronized Time Base)

[SWS_StbM_00442] [For Synchronized Time Bases the Main Time Tuple shall be updated according to [\[SWS_StbM_00440\]](#) and [\[SWS_StbM_00342\]](#).

Upon invocation of [StbM_SetRateCorrection](#) the StbM shall calculate a temporary Time Tuple according to [\[SWS_StbM_00424\]](#) and replace the Main Time Tuple by this temporary Time Tuple. For calculation of the temporary Time Tuple StbM shall use the r value, which is valid before it is updated by current call of [StbM_SetRateCorrection](#).] ([RS_TS_00018](#))

[SWS_StbM_00443] [For Offset Time Bases the Main Time Tuple shall be updated according to [\[SWS_StbM_00441\]](#), [\[SWS_StbM_00190\]](#) and [\[SWS_StbM_00304\]](#).

Upon invocation of [StbM_SetRateCorrection](#) the StbM shall calculate a temporary Time Tuple according to [\[SWS_StbM_00424\]](#) and replace the Main Time Tuple by this temporary Time Tuple. For calculation of the temporary Time Tuple StbM shall use the r value, which is valid before it is updated by current call of [StbM_SetRateCorrection](#).] ([RS_TS_00018](#))

[SWS_StbM_00422] [

- For Time Bases with [StbMSynchronizedTimeBaseIdentifier](#) 32 to 127 and
- for Time Bases with [StbMSynchronizedTimeBaseIdentifier](#) 0 to 31 and [StbMIsSystemWideGlobalTimeMaster](#) equals TRUE

the StbM shall return on invocation of [StbM_GetRateDeviation](#) the rate deviation that has been set by [StbM_SetRateCorrection](#) for that Time Base.

If no rate deviation has been set, [StbM_GetRateDeviation](#) shall return `E_NOT_OK`.] ([RS_TS_00018](#))

[SWS_StbM_00431] [For the Time Master of a Synchronized Time Base the StbM shall use $r_{rc} = 1$, if a valid correction rate (r_{rc}) has not yet been set.

For the Time Master of an Offset Time Base the StbM shall use $r_{orc} = 1$, if a valid correction rate (r_{orc}) has not yet been set.] ([RS_TS_00018](#))

7.3.9 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 cloning API avoids loss of precision when copying the Time Bases. Possible use cases for cloning are fallback scenarios as well as redundancy.

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).

The StbM supports cloning for Synchronized and Pure Local Time Bases. Offset Time Bases are not supported.

[SWS_StbM_00530] [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.

] ([RS_TS_00038](#))

[SWS_StbM_00531] [For Synchronized Time Bases and Pure Local Time Bases, the StbM shall check on every change of the `TIMEOUT` bit in the `timeBaseStatus` if a pending deferred clone request exists for which the Source part of the [Source;Destination] tuple is the same as the respective Time Base.

If such a pending deferred clone request exists it shall be removed.] ([RS_TS_00038](#))

Note: As a result any pending deferred clone request is removed when the related Source Time Base enters `TIMEOUT` state.

If the Source Time Base is already in `TIMEOUT` state when `StbM_CloneTimeBase` is invoked with the `DEFERRED_COPY` flag being 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] [For Synchronized Time Bases and Pure Local Time Bases, the StbM shall check on every invocation of `StbM_BusSetGlobalTime` if a pending deferred clone request exists for which the Source part of the [Source;Destination] tuple is the same as the respective Time Base.

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 it shall be removed.] (*RS_TS_00038*)

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

[SWS_StbM_00533] [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`.] (*RS_TS_00038*)

[SWS_StbM_00534] [When performing the clone operation the StbM shall copy the Time Base value and the Time Base User Bytes.

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.

](RS_TS_00038)

[SWS_StbM_00535] [When performing the clone operation the StbM shall check the flag `APPLY_RATE` of the `cloneCfg` parameter of the clone request.

If the flag is set, the StbM shall copy the Rate Correction value of the Source Time Base to the Destination Time Base.](RS_TS_00038)

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

[SWS_StbM_00536] [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.](RS_TS_00038)

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`.

7.3.10 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.

7.3.10.1 Time Notifications

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

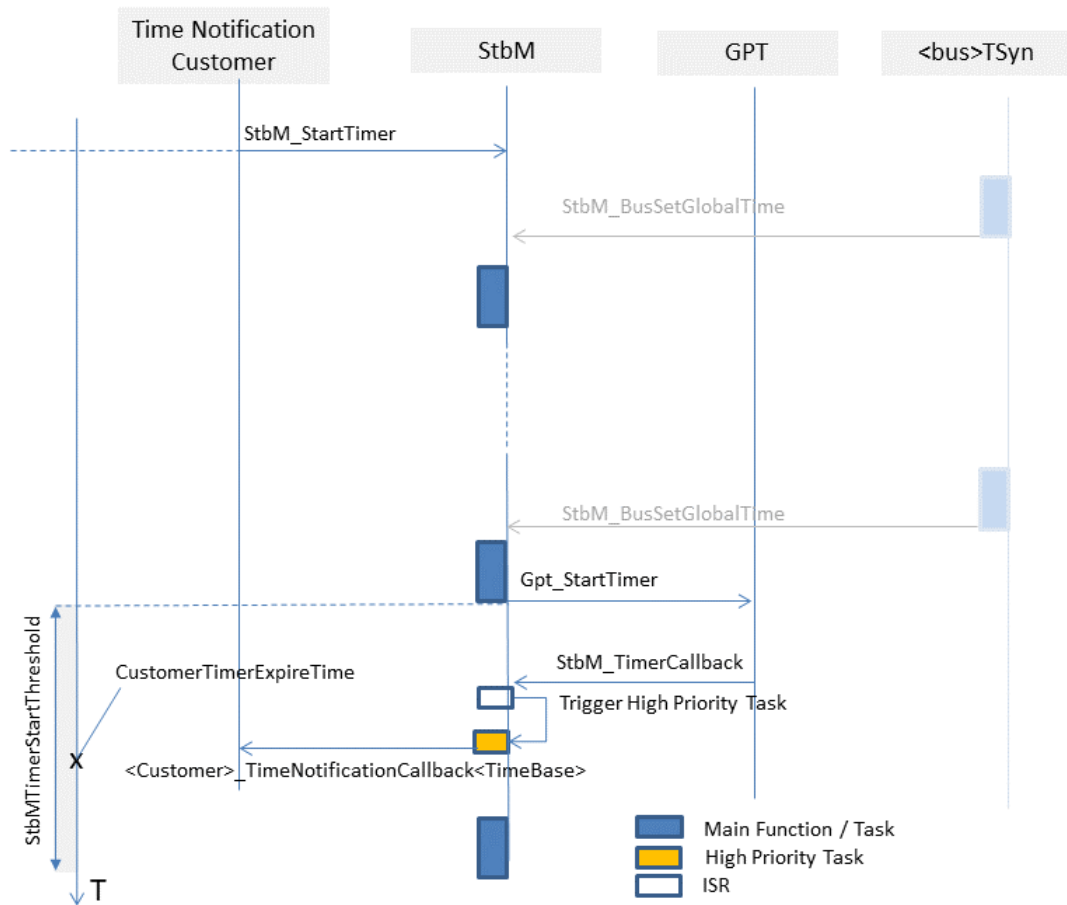


Figure 7.9: Basic mechanism of Time Notification

[SWS_StbM_00421] [If any `StbMNotificationCustomer` is configured, the StbM shall use one additional GPT source (referenced by `StbMGptTimerRef`), which is not used for other purposes.]([RS_TS_00017](#))

[SWS_StbM_00270] [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`.]([RS_TS_00017](#))

[SWS_StbM_00335] [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`.]([RS_TS_00017](#))

Note: Cyclic recalculation accounts for asynchronous updates of the Time Base e.g. by `StbM_BusSetGlobalTime`.

[SWS_StbM_00336] [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.](*RS_TS_00017*)

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] [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` 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`.](*RS_TS_00017*)

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] [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.](RS_TS_00017)

Note: This can happen, if 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.

[SWS_StbM_00337] [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.](RS_TS_00017)

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).

7.3.10.2 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] [For Synchronized, Offset 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`.](RS_TS_00037, RS_TS_00016)

[SWS_StbM_00526] [The StbM shall call the Status Notification callback from `StbM_MainFunction`.](RS_TS_00037, RS_TS_00016)

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] [For each Time Base the StbM has a configurable mask `StbM_StatusNotificationMask`, which allows to mask individually status event notifications.]([RS_TS_00037](#), [RS_TS_00016](#))

[SWS_StbM_00284] [The StbM shall detect the following status events:

Status Event Name	Status Event Set Condition
EV_GLOBAL_TIME	1: GLOBAL_TIME_BASE in <code>timeBaseStatus</code> has changed from 0 to 1 0: otherwise
EV_TIMEOUT_OCCURRED	1: TIMEOUT bit in <code>timeBaseStatus</code> in has changed from 0 to 1 0: otherwise
EV_TIMEOUT_REMOVED	1: TIMEOUT bit in <code>timeBaseStatus</code> in has changed from 1 to 0 0: otherwise
EV_TIMELEAP_FUTURE	1: TIMELEAP_FUTURE bit in <code>timeBaseStatus</code> in has changed from 0 to 1 0: otherwise
EV_TIMELEAP_FUTURE_REMOVED	1: TIMELEAP_FUTURE bit in <code>timeBaseStatus</code> in has changed from 1 to 0 0: otherwise
EV_TIMELEAP_PAST	1: TIMELEAP_PAST bit in <code>timeBaseStatus</code> in has changed from 0 to 1 0: otherwise
EV_TIMELEAP_PAST_REMOVED	1: TIMELEAP_PAST bit in <code>timeBaseStatus</code> in has changed from 1 to 0 0: otherwise
EV_SYNC_TO_SUBDOMAIN	1: SYNC_TO_GATEWAY bit in <code>timeBaseStatus</code> in has changed from 0 to 1 0: otherwise
EV_SYNC_TO_GLOBAL_MASTER	1: SYNC_TO_GATEWAY bit in <code>timeBaseStatus</code> in 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

Table 7.2: Status Event Table

]([RS_TS_00016](#))

[SWS_StbM_00278] [For Synchronized and Offset 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]).] (RS_TS_00037)

[SWS_StbM_00282] [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.] (RS_TS_00037)

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_SYNC_TO_SUBDOMAIN` and
- `EV_SYNC_TO_GLOBAL_MASTER`.

[SWS_StbM_00280] [After reporting a status event via the `StatusNotificationCallback<TimeBase>` API and the `StatusNotification` service interface the StbM shall reset `NotificationEvents` to 0.] (RS_TS_00016)

7.3.11 Triggering Customers

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

[SWS_StbM_00020] [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.] ([SRS_BSW_00429](#), [RS_TS_00037](#), [RS_TS_00032](#))

[SWS_StbM_00022] [The Synchronized Time-Base Manager shall provide means to configure the Time Base to which the OS ScheduleTable should be synchronized (see container [StbMTriggeredCustomer](#)).] ([RS_TS_00037](#), [RS_TS_00032](#))

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] [Customers of type Triggered Customer shall be invoked periodically by the Synchronized Time-Base Manager.] ([RS_TS_00032](#))

[SWS_StbM_00031] [If a Triggered Customer is configured (refer to [StbMTriggeredCustomer](#)), the Synchronized Time-Base Manager shall monitor the cyclic execution of the [StbM_MainFunction](#).

This is to guarantee cyclic synchronization of OS schedule tables.] ([RS_TS_00025](#))

[SWS_StbM_00093] [The triggering period [StbMTriggeredCustomerPeriod](#) shall be configurable for each Triggered Customer.] ([RS_TS_00037](#), [RS_TS_00032](#))

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

[SWS_StbM_00302] [The StbM shall set the synchronization count of the OS ScheduleTable via [SyncScheduleTable](#).] ([RS_TS_00032](#))

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

[SWS_StbM_00303] [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](#)).] ([RS_TS_00037](#), [RS_TS_00032](#))

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

[SWS_StbM_00077] [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

]([RS_TS_00032](#))

[SWS_StbM_00092] [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_SYNCHRONOUS`.

]([SRS_BSW_00429](#), [RS_TS_00032](#))

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).

7.3.12 Time Recording

7.3.12.1 General

[SWS_StbM_00307] [The StbM shall support the Global Time precision measurement for a Time Base, if `StbMTimeRecordingSupport` is set to `TRUE`.]([RS_TS_00034](#))

7.3.12.2 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.

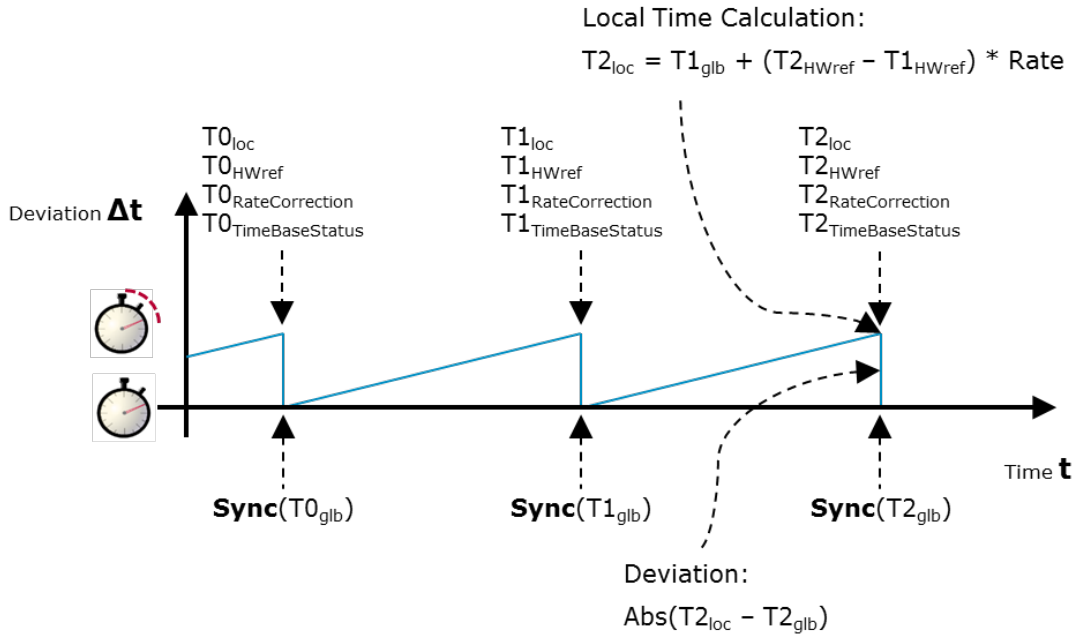


Figure 7.10: Simplified view how the recorded Time Base related snapshot data are taken

[SWS_StbM_00428] [The StbM shall do Global Time precision measurement only for Synchronized Time Bases and Offset Time Bases, for which `StbMIsSystemWideGlobalTimeMaster` is set to `FALSE`.] ([RS_TS_00034](#))

7.3.12.2.1 Synchronized Time Base Record Table

[SWS_StbM_00308] [If Global Time Precision Measurement is enabled (refer to [\[SWS_StbM_00428\]](#) and [\[SWS_StbM_00307\]](#)) for the Time Base, the StbM shall establish a table to record values depending on the Synchronized Time Base with the following structure:

	Record Table Element	Multiplicity	Range	Bytes	Type	Unit
Header		1		9		
	Synchronized-TimeDomain	1	0 .. 15	1	uint8	
	HWfrequency	1	0 .. 4294967295	4	uint32	Hz
	HWprescaler	1	0 .. 4294967295	4	uint32	
Block 0		1		27		
	GlbSeconds	1	0 .. 4294967295	4	StbM_TimeStampType.seconds	sec

	GlbNanoSeconds	1	0 .. 9999999999	4	StbM_TimeStampType .nanoseconds	ns
	TimeBaseStatus	1	0 .. 255	1	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
Block 1	...	1				
...		1				
Block (Block-Count-1)	...	1				

Table 7.3: Synchronized Time Base Record Table

](RS_TS_00034)

[SWS_StbM_00309] [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.] (RS_TS_00034)

[SWS_StbM_00310] [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.] (RS_TS_00034)

[SWS_StbM_00382] [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_00308]) shall contain a history of as many blocks as configured by `StbMSyncTimeRecordTableBlockCount`.] (RS_TS_00034)

7.3.12.2.2 Offset Time Base Record Table

[SWS_StbM_00311] [If Global Time Precision Measurement is enabled (refer to [SWS_StbM_00428] and [SWS_StbM_00307]) for the Time Base, the StbM shall establish a table to record values depending on the Offset Time Base with the following structure:

	Record Table Element	Multiplicity	Range	Bytes	Type	Unit
Header		1		9		
Block 0	OffsetTimeDomain	1	16 .. 31	1	uint8	
		1		27		
	GlbSeconds	1	0 .. 4294967295	4	StbM_TimeStampType .seconds	sec
	GlbNanoSeconds	1	0 .. 9999999999	4	StbM_TimeStampType .nanoseconds	ns
	TimeBaseStatus	1	0 .. 255	1	StbM_TimeBaseStatusType	
Block 1	...	1				
...		1				
Block (Block-Count-1)	...	1				

Table 7.4: Offset Time Base Record Table

]([RS_TS_00034](#))

[SWS_StbM_00383] [If Global Time Precision Measurement is enabled (refer to [\[SWS_StbM_00428\]](#)) for the Time Base, the Offset Time Base Record Table (refer to [\[SWS_StbM_00311\]](#)) shall contain a history of as many blocks as configured by `StbMOffsetTimeRecordTableBlockCount`.]([RS_TS_00034](#))

7.3.12.2.3 Snapshot Conditions

[SWS_StbM_00312] [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>` or
- `OffsetTimeRecordBlockCallback<TimeBase>`

has yet occurred to pass all blocks with their elements to the application, the StbM shall again overwrite the block containing the oldest measurement data with the incoming measurement data.]([RS_TS_00034](#))

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] [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_00308] 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., TL_{Sync} , see [SWS_StbM_00438]).] (RS_TS_00034)

[SWS_StbM_00314] [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

to the related Synchronized Time Base Record Table (refer to [SWS_StbM_00308] 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}).] (RS_TS_00034)

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

[SWS_StbM_00388] [For Offset 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
- and timeBaseStatus

to the related Offset Time Base Record Table (refer to [SWS_StbM_00311].] (RS_TS_00034)

[SWS_StbM_00315] [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_00308] by calling `StbM_GetSyncTimeRecordHead.`)](RS_TS_00034)

[SWS_StbM_00316] [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 Offset Time Base Record Table (refer to [SWS_StbM_00311] by calling `StbM_GetOffsetTimeRecordHead.`)](RS_TS_00034)

[SWS_StbM_00317] [If Global Time Precision Measurement is enabled (refer to [SWS_StbM_00428] and [SWS_StbM_00307]) for the Time Base, 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.](RS_TS_00034)

[SWS_StbM_00318] [If Global Time Precision Measurement is enabled (refer to [SWS_StbM_00428] and [SWS_StbM_00307]) for the Time Base, the StbM shall notify the Application by calling `OffsetTimeRecordBlockCallback<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.](RS_TS_00034)

7.3.12.3 Time Validation Support

Figure 7.11 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.

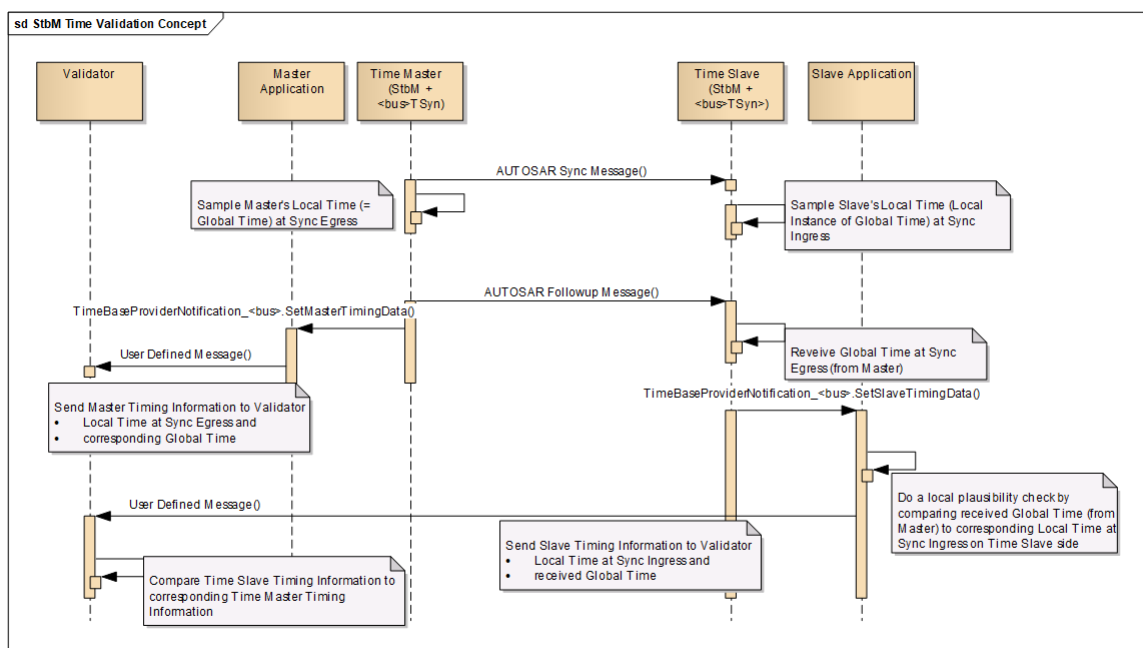


Figure 7.11: : Concept of Time Validation

[SWS_StbM_00465] [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).] ([RS_TS_00034](#))

7.3.12.3.1 Record Tables

7.3.12.3.1.1 Time Slave/Master

[SWS_StbM_00466] [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 [Table “Time Slave Validation Record Table”](#)), which holds

- N blocks (refer to [\[SWS_StbM_00470\]](#))
- of type `StbM_<bus>TimeSlaveMeasurementType` (refer to [\[SWS_StbM_00467\]](#)).

]([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>)			

Table 7.5: Time Slave Validation Record Table

[\[SWS_StbM_00467\]](#) [The type of the blocks in the Time Slave Validation Record Table (refer to [\[SWS_StbM_00466\]](#)) shall depend on the Timesync Module, which provides the data for the Time Slave Validation Record Table of the corresponding Time Base.

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>

Table 7.6: Type Mapping for Time Slave Validation Record Table

]([RS_TS_00034](#))

[\[SWS_StbM_00468\]](#) [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 [Table “Time Master Validation Record Table”](#)), which holds

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

](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>)			

Table 7.7: Time Master Validation Record Table

[SWS_StbM_00469] [The type of the blocks in the Time Master Validation Record Table (refer to [SWS_StbM_00468]) shall depend on the Timesync Module, which provides the data for the Time Master Validation Record Table of the corresponding Time Base.

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

Table 7.8: Type Mapping for Time Master Validation Record Table

](RS_TS_00034)

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.

7.3.12.3.1.2 Pdelay Initiator/Responder

[SWS_StbM_00522] [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 Table “Pdelay Initiator Validation Record Table”), which holds

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

](RS_TS_00034)

	Block Structure Element	Type	Description
Block 0	<code>pdelayInitiatorData[0]</code>	<code>StbM_PdelayInitiatorMeasurementType</code>	Structure to capture Time Validation recording data of a PdelayInitiator
Block 1	<code>pdelayInitiatorData[1]</code>	<code>StbM_PdelayInitiatorMeasurementType</code>	
...			
Block N (= <code>StbMTimeValidationRecordTableBlockCount-1</code>)			

Table 7.9: Pdelay Initiator Validation Record Table

[SWS_StbM_00523] [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 Table “Pdelay Responder Validation Record Table”), which holds

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

](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)			

Table 7.10: Pdelay Responder Validation Record Table

7.3.12.3.1.3 Common

[SWS_StbM_00470] [Each

- 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])

shall contain as many blocks as configured by `StbMTimeValidation-RecordTableBlockCount.` (RS_TS_00034)

7.3.12.3.2 Time Validation Snapshot Conditions

7.3.12.3.2.1 Time Slave/Master

[SWS_StbM_00471] [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`).] (*RS_TS_00034*)

[SWS_StbM_00472] [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`.] (*RS_TS_00034*)

[SWS_StbM_00473] [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].] (*RS_TS_00034*)

[SWS_StbM_00474] [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`.] (*RS_TS_00034*)

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] [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.] (*RS_TS_00034*)

[SWS_StbM_00476] [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. (RS_TS_00034)

7.3.12.3.2 Pdelay Initiator/Responder

[SWS_StbM_00478] [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]). (RS_TS_00034)

[SWS_StbM_00524] [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`. (RS_TS_00034)

[SWS_StbM_00479] [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. (RS_TS_00034)

[SWS_StbM_00480] [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 [\[SWS_StbM_00523\]](#)).] (*RS_TS_00034*)

[SWS_StbM_00525] [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`.] (*RS_TS_00034*)

[SWS_StbM_00481] [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.] (*RS_TS_00034*)

7.3.12.3.2.3 Common

[SWS_StbM_00477] [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.](RS_TS_00034)

7.3.13 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]

7.3.13.1 General

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]{DRAFT} [The StbM shall support the Freshness Values for a Time Base, if `StbMQueryFreshnessValue` is set to a value different than NONE.

](RS_TS_00039)

[SWS_StbM_00542]{DRAFT} [If `StbMQueryFreshnessValue` is set to CFUNC, the functions:

- `StbM_GetTxFreshness`
- `StbM_GetTxFreshnessTruncData`
- `StbM_GetRxFreshness`
- `StbM_SPduTxConfirmation`

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

- `GetTxFreshnessFct`
- `GetTxFreshnessTruncDataFct`
- `GetRxFreshnessFct`
- `SPduTxConfirmationFct`

and return the same return value as those.](RS_TS_00039)

[SWS_StbM_00543]{DRAFT} [If `StbMQueryFreshnessValue` is set to `SERVICE`, the functions:

- `StbM_GetTxFreshness`
- `StbM_GetTxFreshnessTruncData`
- `StbM_GetRxFreshness`
- `StbM_SPduTxConfirmation`

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

- `GetTxFreshness`
- `GetTxFreshnessTruncData`
- `GetRxFreshness`
- `SPduTxConfirmation`

and return the same return value as those.]([RS_TS_00039](#))

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

7.3.14 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"].

7.4 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] [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.]([SRS_BSW_00459](#))

Note: Parameter `StbMEcucPartitionRef` references the partition, which the `StbM_MainFunction` function is allocated to.

[SWS_StbM_00514] [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.]([SRS_BSW_00459](#))

7.5 Error Handling

[SWS_StbM_00199] [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.]([RS_TS_00025](#))

Note: For further details refer to the chapter 7.2 "Error Handling" in [3, SWS BSW General] and [chapter 8](#) for API specific error handling.

7.6 Error Classification

Section "Error Handling" of the document [3, 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.

7.6.1 Development Errors

[SWS_StbM_00041] [

Type of error	Related error code	Error value
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 disabled by configuration	STBM_E_SERVICE_DISABLED	0x12
API called with invalid timestamp	STBM_E_PARAM_TIMESTAMP	0x25
API called with invalid user data	STBM_E_PARAM_USERDATA	0x26

]([SRS_BSW_00337](#), [SRS_BSW_00385](#), [SRS_BSW_00386](#), [SRS_BSW_00327](#), [SRS_BSW_00323](#))

7.6.2 Runtime Errors

There are no runtime errors.

7.6.3 Transient Faults

There are no transient faults.

7.6.4 Production Errors

There are no production errors.

7.6.5 Extended Production Errors

There are no extended production errors.

7.7 Version Check

For details refer to the chapter 5.1.8 "Version Check" in [3, 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] [

<i>Module</i>	<i>Header File</i>	<i>Imported Type</i>
Can	Can_GeneralTypes.h	Can_TimeStampType (draft)
Eth	Eth_GeneralTypes.h	Eth_TimeStampQualType
	Eth_GeneralTypes.h	Eth_TimeStampType
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

]([SRS_BSW_00301](#))

8.2 Type definitions

8.2.1 Extension to Std_ReturnType

[SWS_StbM_91017]{DRAFT} [

Range	STBM_E_BUSY	0x02	The function call failed because the service is still busy
Description	– Tags: atp.Status=draft		
Available via	StbM.h		

]([RS_TS_00039](#))

8.2.2 StbM_ConfigType

[SWS_StbM_00249] [

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

]([SRS_BSW_00414](#))

8.2.3 StbM_VirtualLocalTimeType

[SWS_StbM_91003] [

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	

]([RS_TS_00009](#))

8.2.4 StbM_MeasurementType

[SWS_StbM_00384] [

Name	StbM_MeasurementType	
Kind	Structure	
Elements	pathDelay	
	Type	uint32
	Comment	Propagation delay in nanoseconds
Description	Structure which contains additional measurement data	
Available via	StbM.h	

]([RS_TS_00034](#))

8.3 Function definitions

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

8.3.1 StbM_GetVersionInfo

[SWS_StbM_00066] [

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	

](SRS_BSW_00407)

[SWS_StbM_00094] [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).](SRS_BSW_00386, SRS_BSW_00337)

8.3.2 StbM_Init

[SWS_StbM_00052] [

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	

]([SRS_BSW_00101](#), [SRS_BSW_00358](#), [SRS_BSW_00414](#)) 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] [A static status variable denoting if the StbM is initialized shall be initialized with value 0 before any APIs of the StbM are called.]([SRS_BSW_00406](#))

[SWS_StbM_00121] [`StbM_Init` shall set the static status variable to a value not equal to 0.]([SRS_BSW_00406](#))

8.3.3 StbM_GetCurrentTime

[SWS_StbM_00195] [

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 value (Local Time Base derived from Global Time Base) in standard format. Note: This API shall be called with locked interrupts / within an Exclusive Area to prevent interruption (i.e., the risk that the time stamp is outdated on return of the function call).	
Available via	StbM.h	

]([RS_TS_00005](#), [RS_TS_00006](#), [RS_TS_00029](#), [RS_TS_00030](#), [RS_TS_00031](#), [RS_TS_00014](#))

[SWS_StbM_00196] [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.

]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00197] [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`.] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.4 StbM_GetCurrentTimeExtended

[SWS_StbM_00200]{OBSOLETE} [

Service Name	StbM_GetCurrentTimeExtended (obsolete)	
Syntax	<pre>Std_ReturnType StbM_GetCurrentTimeExtended (StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeStampExtendedType* timeStamp, StbM_UserDataType* userData)</pre>	
Service ID [hex]	0x08	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	time base reference
Parameters (inout)	None	
Parameters (out)	timeStamp	Current time stamp 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 value (Local Time Base derived from Global Time Base) in extended format. Note: This API shall be called with locked interrupts / within an Exclusive Area to prevent interruption (i.e., the risk that the time stamp is outdated on return of the function call). Tags: atp.Status=obsolete	
Available via	StbM.h	

](RS_TS_00005, RS_TS_00014)

8.3.5 StbM_GetCurrentVirtualLocalTime

[SWS_StbM_91006] [

Service Name	StbM_GetCurrentVirtualLocalTime	
Syntax	<pre>Std_ReturnType StbM_GetCurrentVirtualLocalTime (StbM_SynchronizedTimeBaseType timeBaseId, StbM_VirtualLocalTimeType* localTimePtr)</pre>	
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	

](RS_TS_00006, RS_TS_00008)

[SWS_StbM_00444] [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`.]
([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00445] [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.

]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.6 StbM_SetGlobalTime

[SWS_StbM_00213] [

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	

]([RS_TS_00029](#), [RS_TS_00010](#))

[SWS_StbM_00214] [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.

]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00215] [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`.] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00448] [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).] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00449] [If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_SetGlobalTime` shall report to DET the development error `STBM_E_PARAM_USER_DATA`, if called with an invalid value of parameter `userData`, i.e., `userDataLength` > 3.] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.7 StbM_UpdateGlobalTime

[SWS_StbM_00385] [

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)	<code>timeBaseId</code>	time base reference
	<code>timeStamp</code>	New time stamp
	<code>userData</code>	New user data (if not NULL)
Parameters (inout)	None	
Parameters (out)	None	
Return value	<code>Std_ReturnType</code>	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 <code>UpdateGlobalTime</code> will not lead to an immediate transmission of the Global Time.	
Available via	StbM.h	

] ([RS_TS_00010](#))

[SWS_StbM_00340] [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.

] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00341] [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`.] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00451] [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).] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00452] [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., `userDataLength` > 3.] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.8 StbM_SetUserData

[SWS_StbM_00218] [

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	

] ([RS_TS_00015](#))

[SWS_StbM_00219] [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.

] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00220] [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`.]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00457] [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., `userDataLength > 3`.]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.9 StbM_SetOffset

[SWS_StbM_00223] [

Service Name	StbM_SetOffset	
Syntax	<pre>Std_ReturnType StbM_SetOffset (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_TimeStampType* timeStamp, const StbM_UserDataType* userData)</pre>	
Service ID [hex]	0x0d	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	<code>timeBaseId</code>	time base reference
	<code>timeStamp</code>	New offset time stamp
	<code>userData</code>	New User Data (Or <code>NULL</code> if no new user data is provided)
Parameters (inout)	None	
Parameters (out)	None	
Return value	<code>Std_ReturnType</code>	<code>E_OK</code> : successful <code>E_NOT_OK</code> : failed
Description	Allows the Customers and the Timesync Modules to set the Offset Time and the User Data.	
Available via	StbM.h	

]([RS_TS_00029](#), [RS_TS_00013](#))

[SWS_StbM_00224] [If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_SetOffset` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which

- is not configured or
- refers to a Synchronized or Pure Local Time Base or
- is within the reserved value range.

]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00225] [If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_SetOffset` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a `NULL` pointer for parameter `timeStamp`.]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00453] [If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_SetOffset` shall report to DET the development error `STBM_E_PARAM_TIMES-`

TAMP, if called with a parameter `timeStamp` that contains invalid elements (e.g., nanoseconds part > 999999999 ns).] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00454] [If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_SetOffset` shall report to DET the development error `STBM_E_PARAM_USERDATA`, if called with an invalid value of parameter `userData`, i.e., `userDataLength > 3`.] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.10 StbM_GetOffset

[SWS_StbM_00228] [

Service Name	StbM_GetOffset	
Syntax	<pre>Std_ReturnType StbM_GetOffset (StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeStampType* timeStamp, StbM_UserDataType* userData)</pre>	
Service ID [hex]	0x0e	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
Parameters (inout)	None	
Parameters (out)	timeStamp	Current Offset Time value
	userData	Current User Data
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Allows the Timesync Modules to get the current Offset Time and User Data.	
Available via	StbM.h	

] ([RS_TS_00012](#), [RS_TS_00029](#), [RS_TS_00031](#))

[SWS_StbM_00229] [If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetOffset` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which

- is not configured or
- refers to a Synchronized or Pure Local Time Base or
- is within the reserved value range.

] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00230] [If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_GetOffset` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a `NULL` pointer for parameter `timeStamp` or `userData`.] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.11 StbM_BusGetCurrentTime

[SWS_StbM_91005]{OBSOLETE} [

Service Name	StbM_BusGetCurrentTime (obsolete)	
Syntax	<pre>Std_ReturnType StbM_BusGetCurrentTime (StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeStampType* globalTimePtr, StbM_VirtualLocalTimeType* localTimePtr, StbM_UserDataType* userData)</pre>	
Service ID [hex]	0x1f	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaseId	Time Base reference
Parameters (inout)	None	
Parameters (out)	globalTimePtr	Value of the local instance of the Global Time, which is sampled when the function is called
	localTimePtr	Value of the Virtual Local Time, which is sampled when the function is called
	userData	User data of the Time Base
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Returns the current Time Tuple, status and User Data of the Time Base. Tags: atp.Status=obsolete	
Available via	StbM.h	

]([RS_TS_00005](#), [RS_TS_00006](#), [RS_TS_00029](#), [RS_TS_00031](#), [RS_TS_00014](#))

8.3.12 StbM_BusSetGlobalTime

[SWS_StbM_00233] [

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	Allows the Time Base Provider Modules to forward the Rx Time Tuple to the StbM.	
Available via	StbM.h	

]([RS_TS_00007](#), [RS_TS_00030](#), [RS_TS_00031](#), [RS_TS_00034](#))

[SWS_StbM_00234] [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

]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

Note: A parameter `timeBaseId` within the reserved value range indicates legacy use.

[SWS_StbM_00235] [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`

]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00455] [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).]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00456] [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., `userDataLength > 3`.]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.13 StbM_GetRateDeviation

[SWS_StbM_00378] [

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	

](RS_TS_00018)

[SWS_StbM_00379] [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.

](SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00380] [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`.] (SRS_BSW_00386, SRS_BSW_00323)

8.3.14 StbM_SetRateCorrection

[SWS_StbM_00390] [

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	Allows to set the rate of a Synchronized Time Base (being either a Pure Local Time Base or not).	
Available via	StbM.h	

](RS_TS_00018)

[SWS_StbM_00391] [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.

](SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00392] [If the switch `StbMDevErrorDetect` is set to `TRUE`, `StbM_SetRateCorrection` shall report to DET the development error `STBM_E_SERVICE_DISABLED`, if `StbMAllowMasterRateCorrection` is set to `FALSE` for the corresponding Time Base, i.e., it is not allowed to call `StbM_SetRateCorrection`.]
(SRS_BSW_00386, SRS_BSW_00323)

8.3.15 StbM_GetTimeLeap

[SWS_StbM_00267] [

Service Name	StbM_GetTimeLeap	
Syntax	Std_ReturnType StbM_GetTimeLeap (StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeDiffType* timeJump)	
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	

](RS_TS_00005)

[SWS_StbM_00268] [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.

]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00269] [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`.]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.16 StbM_GetTimeBaseStatus

[SWS_StbM_00263] [

Service Name	StbM_GetTimeBaseStatus	
Syntax	<pre>Std_ReturnType StbM_GetTimeBaseStatus (StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeBaseStatusType* syncTimeBaseStatus, StbM_TimeBaseStatusType* offsetTimeBaseStatus)</pre>	
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
	offsetTimeBaseStatus	Status of the Offset 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 and, if called for an Offset Time Base, for the Offset Time Base and the underlying Synchronized Time Base.	
Available via	StbM.h	

]([RS_TS_00005](#))

[SWS_StbM_00264] [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.

]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00386] [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` or `offsetTimeBaseStatus`.] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.17 StbM_CloneTimeBase

[SWS_StbM_91012] [

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 <code>StbMSourceTimeBase</code> (ECUC_StbM_00074).</p> <p><code>StbM_?CloneTimeBase</code> behaves synchronously (immediate copy of Time Base) if <code>DEFERRED_COPY</code> flag of parameter <code>cloneCfg</code> 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	

] ([RS_TS_00038](#))

[SWS_StbM_00537] [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 `timeBaseId`, which

- is not configured or
- refers to an Offset Time Base or
- is within the reserved value range.

] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.18 StbM_StartTimer

[SWS_StbM_00272] [

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	Status of the Synchronized Time Base
	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	

]([RS_TS_00017](#))

[SWS_StbM_00296] [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.

]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00406] [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.]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00298] [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`.]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.19 StbM_GetSyncTimeRecordHead

[SWS_StbM_00319] [

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)	timeBaseld	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	

](RS_TS_00034)

[SWS_StbM_00320] [The function `StbM_GetSyncTimeRecordHead` shall be pre compile time configurable ON/OFF by the configuration parameter `StbMTimeRecordingSupport`.](RS_TS_00034)

[SWS_StbM_00394] [If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_GetSyncTimeRecordHead` 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 or an Offset Time Base or
- is within the reserved value range.

](SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00405] [If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_GetSyncTimeRecordHead` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with an invalid pointer of parameter `syncRecordTableHead`.](SRS_BSW_00386, SRS_BSW_00323)

8.3.20 StbM_GetOffsetTimeRecordHead

[SWS_StbM_00325] [

Service Name	StbM_GetOffsetTimeRecordHead	
Syntax	<pre>Std_ReturnType StbM_GetOffsetTimeRecordHead (StbM_SynchronizedTimeBaseType timeBaseId, StbM_OffsetRecordTableHeadType* offsetRecordTableHead)</pre>	
Service ID [hex]	0x17	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	timeBaseId	Time Base reference
Parameters (inout)	None	
Parameters (out)	offsetRecordTableHead	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 Offset Time Base.	
Available via	StbM.h	

](RS_TS_00034)

[SWS_StbM_00326] [The function `StbM_GetOffsetTimeRecordHead` shall be pre compile time configurable ON/OFF by the configuration parameter `StbMTimeRecordingSupport`.](RS_TS_00034)

[SWS_StbM_00327] [If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_GetOffsetTimeRecordHead` 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 or a Synchronized Time Base or
- is within the reserved value range.

](SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00404] [If the switch `StbMDevErrorDetect` is set to TRUE, `StbM_GetOffsetTimeRecordHead` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with an invalid pointer of parameter `offsetRecordTableHead`.](SRS_BSW_00386, SRS_BSW_00323)

8.3.21 StbM_TriggerTimeTransmission

[SWS_StbM_00346] [

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	

](RS_TS_00011)

[SWS_StbM_00349] [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.

](SRS_BSW_00386, SRS_BSW_00323)

8.3.22 StbM_GetTimeBaseUpdateCounter

[SWS_StbM_00347] [

Service Name	StbM_GetTimeBaseUpdateCounter	
Syntax	uint8 StbM_GetTimeBaseUpdateCounter (StbM_SynchronizedTimeBaseType timeBaseId)	
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

](RS_TS_00011)

[SWS_StbM_00348] [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.

](SRS_BSW_00386, SRS_BSW_00323)

8.3.23 StbM_GetMasterConfig

[SWS_StbM_91002] [

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. <code>StbM_SetGlobalTime</code>) for a given Time Base is available or not.	
Available via	StbM.h	

](RS_TS_00029)

[SWS_StbM_00415] [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.

](SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00416] [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`.] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.24 StbM_CanSetSlaveTimingData

[SWS_StbM_00484]{DRAFT} [

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	

] ([RS_TS_00030](#), [RS_TS_00031](#), [RS_TS_00034](#))

8.3.25 StbM_FrSetSlaveTimingData

[SWS_StbM_00485]{DRAFT} [

Service Name	StbM_FrSetSlaveTimingData (draft)	
Syntax	<pre>Std_ReturnType StbM_FrSetSlaveTimingData (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_FrTimeSlaveMeasurementType* measureDataPtr)</pre>	
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

]([RS_TS_00030](#), [RS_TS_00031](#), [RS_TS_00034](#))

8.3.26 StbM_EthSetSlaveTimingData

[SWS_StbM_00486]{DRAFT} [

Service Name	StbM_EthSetSlaveTimingData (draft)	
Syntax	<pre>Std_ReturnType StbM_EthSetSlaveTimingData (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_EthTimeSlaveMeasurementType* measureDataPtr)</pre>	
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	

]([RS_TS_00030](#), [RS_TS_00031](#), [RS_TS_00034](#))

[SWS_StbM_00487] [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.]([RS_TS_00034](#))

[SWS_StbM_00488] [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.]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

Note: A parameter `timeBaseId` within the reserved value range indicates legacy use.

[SWS_StbM_00489] [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`.]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.27 StbM_CanSetMasterTimingData

[SWS_StbM_00490]{DRAFT} [

Service Name	StbM_CanSetMasterTimingData (draft)	
Syntax	<pre>Std_ReturnType StbM_CanSetMasterTimingData (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_CanTimeMasterMeasurementType* measureDataPtr)</pre>	
Service ID [hex]	0x20	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaselId	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	

]([RS_TS_00029](#), [RS_TS_00031](#), [RS_TS_00034](#))

8.3.28 StbM_FrSetMasterTimingData

[SWS_StbM_00491]{DRAFT} [

Service Name	StbM_FrSetMasterTimingData (draft)	
Syntax	<pre>Std_ReturnType StbM_FrSetMasterTimingData (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_FrTimeMasterMeasurementType* measureDataPtr)</pre>	
Service ID [hex]	0x21	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeBaselId	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	

]([RS_TS_00029](#), [RS_TS_00031](#), [RS_TS_00034](#))

8.3.29 StbM_EthSetMasterTimingData

[SWS_StbM_00492]{DRAFT} [

Service Name	StbM_EthSetMasterTimingData (draft)	
Syntax	<pre>Std_ReturnType StbM_EthSetMasterTimingData (StbM_SynchronizedTimeBaseType timeBaseId, const StbM_EthTimeMasterMeasurementType* measureDataPtr)</pre>	
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	

] ([RS_TS_00029](#), [RS_TS_00031](#), [RS_TS_00034](#))

[SWS_StbM_00493] [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>TSyn TimeValidationSupport` in `<bus>TSyn` module), `StbM_<bus>SetMasterTimingData` shall be ON, otherwise .] ([RS_TS_00034](#))

[SWS_StbM_00494] [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] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00495] [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`.] ([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.30 StbM_EthSetPdelayInitiatorData

[SWS_StbM_00496]{DRAFT} [

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	

](RS_TS_00034)

[SWS_StbM_00497] [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.](RS_TS_00034)

[SWS_StbM_00498] [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](SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00499] [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`.](SRS_BSW_00386, SRS_BSW_00323)

8.3.31 StbM_EthSetPdelayResponderData

[SWS_StbM_00500]{DRAFT} [

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)	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	<p>–</p> <p>Tags: atp.Status=draft</p>	
Available via	StbM_EthTSyn.h	

](RS_TS_00034)

[SWS_StbM_00501] [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.](RS_TS_00034)

[SWS_StbM_00502] [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](SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00503] [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`](SRS_BSW_00386, SRS_BSW_00323, RS_TS_00034)

8.3.32 StbM_GetBusProtocolParam

[SWS_StbM_91007] [

Service Name	StbM_GetBusProtocolParam	
Syntax	<pre>Std_ReturnType StbM_GetBusProtocolParam (StbM_SynchronizedTimeBaseType timeBaseId, StbM_ProtocolParamType* protocolParam)</pre>	
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	

]([RS_TS_20069](#))

[SWS_StbM_00518] [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.]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00519] [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`.]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.33 StbM_SetBusProtocolParam

[SWS_StbM_91008] [

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	

](RS_TS_20069)

[SWS_StbM_00520] [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.](SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00521] [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`.](SRS_BSW_00386, SRS_BSW_00323)

8.3.34 StbM_GetTxFreshness

[SWS_StbM_91018]{DRAFT} [

Service Name	StbM_GetTxFreshness (draft)	
Syntax	Std_ReturnType StbM_GetTxFreshness (uint16 StbMFreshnessValueId, uint32* StbMFreshnessValueLength, uint8* StbMFreshnessValue)	
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 Freshness ValueId 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. Tags: atp.Status=draft	
Available via	StbM.h	

](RS_TS_00039)

[SWS_StbM_00544]{DRAFT} [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.](SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00545]{DRAFT} [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`

]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

8.3.35 StbM_GetTxFreshnessTruncData

[SWS_StbM_91014]{DRAFT} [

Service Name	StbM_GetTxFreshnessTruncData (draft)	
Syntax	<pre>Std_ReturnType StbM_GetTxFreshnessTruncData (uint16 StbMFreshnessValueId, uint32* StbMFreshnessValueLength, uint32* StbMTruncatedFreshnessValueLength, uint8* StbMFreshnessValue, uint8* StbMTruncatedFreshnessValue)</pre>	
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 PDU. 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 Freshness ValueId 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 PDU. Tags: atp.Status=draft	
Available via	StbM.h	

]([RS_TS_00039](#))

[SWS_StbM_00546]{DRAFT} [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.]([SRS_BSW_00386](#), [SRS_BSW_00323](#))

[SWS_StbM_00547]{DRAFT} [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`

] (`SRS_BSW_00386`, `SRS_BSW_00323`)

8.3.36 StbM_SPduTxConfirmation

[SWS_StbM_91015]{DRAFT} [

Service Name	StbM_SPduTxConfirmation (draft)	
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	<p>This interface is used by the StbM to indicate that the Secured Time Synchronization Message has been initiated for transmission.</p> <p>Tags: atp.Status=draft</p>	
Available via	StbM.h	

]()

[SWS_StbM_00548]{DRAFT} [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.] (`SRS_BSW_00386`, `SRS_BSW_00323`)

8.3.37 StbM_GetRxFreshness

[SWS_StbM_91016]{DRAFT} [

Service Name	StbM_GetRxFreshness (draft)	
Syntax	<pre>Std_ReturnType StbM_GetRxFreshness (uint16 StbMFreshnessValueId, const uint8* StbMTruncatedFreshnessValue, uint32 StbMTruncatedFreshnessValueLength, uint16 StbMAuthVerifyAttempts, uint32* StbMFreshnessValueLength, uint8* StbMFreshnessValue)</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 PDU.
	StbMTruncatedFreshnessValueLength	Holds the length in bits of the truncated freshness value.
	StbMAuthVerifyAttempts	Holds the number of authentication verify attempts of this PDU 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 Freshness ValueId 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. Tags: atp.Status=draft	
Available via	StbM.h	

](RS_TS_00039)

[SWS_StbM_00549]{DRAFT} [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.](SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00550]{DRAFT} [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`

](SRS_BSW_00386, SRS_BSW_00323)

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] [

Service Name	StbM_MainFunction
Syntax	void StbM_MainFunction (void)
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

]([SRS_BSW_00172](#), [SRS_BSW_00373](#))

[SWS_StbM_00407] [The frequency of invocations of [StbM_MainFunction](#) is determined by the configuration parameter [StbMMainFunctionPeriod](#).]([SRS_BSW_00172](#))

[SWS_StbM_00107] [If OS is configured as triggered customer, the function [StbM_MainFunction](#) shall synchronize the referenced OS ScheduleTable.]([RS_TS_00032](#), [SRS_BSW_00333](#))

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] [

API Function	Header File	Description
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.

]([SRS_BSW_00301](#), [SRS_BSW_00339](#))

8.6.2 Optional interfaces

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

[SWS_StbM_00059] [

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	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.
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.

]([SRS_BSW_00301](#), [SRS_BSW_00339](#))

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 [3, 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] [

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	

)](RS_TS_00034)

[SWS_StbM_00323] [The function `SyncTimeRecordBlockCallback<timeBaseId>` shall be set by the parameter `StbMSyncTimeRecordBlockCallback`.](RS_TS_00034)

8.6.3.2 OffsetTimeRecordBlockCallback

[SWS_StbM_00328] [

Service Name	OffsetTimeRecordBlockCallback<TimeBase>	
Syntax	Std_ReturnType OffsetTimeRecordBlockCallback<TimeBase> (const StbM_OffsetRecordTableBlockType* offsetRecordTableBlock)	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	





Parameters (in)	offsetRecordTableBlock	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 Offset Time Base.	
Available via	StbM_Externals.h	

](RS_TS_00034)

[SWS_StbM_00329] [The function `OffsetTimeRecordBlockCallback<timeBaseId>` shall set by the parameter `StbMOffsetTimeRecordBlockCallback`.](RS_TS_00034)

8.6.3.3 StatusNotificationCallback

[SWS_StbM_00285] [

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	

](RS_TS_00037, RS_TS_00016, SRS_BSW_00457, SRS_BSW_00360, SRS_BSW_00333)

[SWS_StbM_00299] [The status notification callback function shall be set by the parameter `StbMTimeNotificationCallback`.](RS_TS_00016)

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.

8.6.3.4 <Customer>_TimeNotificationCallback

[SWS_StbM_00273] [

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	

]([RS_TS_00017](#), [SRS_BSW_00457](#), [SRS_BSW_00360](#), [SRS_BSW_00333](#))

[SWS_StbM_00274] [The event notification callback function shall be set by the parameter [StbMTimeNotificationCallback](#)]([RS_TS_00017](#))

8.6.3.5 SPduTxConfirmationFct

[SWS_StbM_91022]{DRAFT} [

Service Name	SPduTxConfirmationFct (draft)	
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. Tags: atp.Status=draft	
Available via	StbM_Externals.h	

]([RS_TS_00039](#))

[SWS_StbM_00551]{DRAFT} [The [SPduTxConfirmationFct](#) function shall be set by the parameter [StbMGetTxConfFreshnessValueFuncName.](#)]([RS_TS_00039](#))

8.6.3.6 GetTxFreshnessTruncDataFct

[SWS_StbM_91023]{DRAFT} [

Service Name	GetTxFreshnessTruncDataFct (draft)	
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 PDU. 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 Freshness ValueId 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 PDU. Tags: atp.Status=draft	
Available via	StbM_Externals.h	

] ([RS_TS_00039](#))

[SWS_StbM_00552]{DRAFT} [The [GetTxFreshnessTruncDataFct](#) function shall be set by the parameter [StbMGetTxTruncFreshnessValueFuncName](#).] ([RS_TS_00039](#))

8.6.3.7 GetTxFreshnessFct

[SWS_StbM_91024]{DRAFT} [

Service Name	GetTxFreshnessFct (draft)	
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)	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 Freshness ValueId 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 Tags: atp.Status=draft	
Available via	StbM_Externals.h	

](RS_TS_00039)

[SWS_StbM_00553]{DRAFT} [The `GetTxFreshnessFct` function shall be set by the parameter `StbMGetTxFreshnessValueFuncName.`](RS_TS_00039)

8.6.3.8 GetRxFreshnessFct

[SWS_StbM_91025]{DRAFT} [

Service Name	GetRxFreshnessFct (draft)	
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 PDU.
	StbMTruncatedFreshnessValueLength	Holds the length in bits of the truncated freshness value.
	StbMAuthVerifyAttempts	Holds the number of authentication verify attempts of this PDU 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 Freshness ValueId 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. Tags: atp.Status=draft	
Available via	StbM_Externals.h	

|(RS_TS_00039)

[SWS_StbM_00554]{DRAFT} [The `GetRxFreshnessFct` function shall be set by the parameter `StbMGetRxFreshnessValueFuncName.`](RS_TS_00039)

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] [

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)}		

|(RS_TS_00005, RS_TS_00035, RS_TS_00029, RS_TS_00010, RS_TS_00013, RS_TS_00015)

8.7.1.2 GlobalTime_Slave

[SWS_StbM_00248] [

Name	GlobalTime_Slave_{Name}		
Kind	ProvidedPort	Interface	GlobalTime_Slave_{Name}
Description	-		
Port Defined Argument Value(s)	Type	StbM_SynchronizedTimeBaseType	
	Value	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBase Identifier.value)}	
Variation	Name = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		

](RS_TS_00005, RS_TS_00030, RS_TS_00031, RS_TS_00035, RS_TS_00014)

8.7.1.3 GlobalTime_StatusEvent

[SWS_StbM_00290] [

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)}		

](RS_TS_00035, RS_TS_00016)

8.7.1.4 StartTimer

[SWS_StbM_91004] [

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/Stb MNotificationCustomerId.value)}	
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 128 TimeBase = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)} Customer = {ecuc(StbM/StbMSynchronizedTimeBase/StbMNotificationCustomer.SHORT-NAME)}		

](RS_TS_00017)

8.7.2 Required Ports

8.7.2.1 GlobalTime_TimeEvent

[SWS_StbM_00276] [

Name	GlobalTime_TimeEvent_{TBName}_{CName}		
Kind	RequiredPort	Interface	TimeNotification
Description	-		
Variation	({ecuc(StbM/StbMSynchronizedTimeBase/StbMNotificationCustomer/StbMTimeNotification Callback)}==NULL) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBase Identifier)} < 128) TBName={ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)} CName={ecuc(StbM/StbMSynchronizedTimeBase/StbMNotificationCustomer.SHORT-NAME)}		

](RS_TS_00035, RS_TS_00017)

8.7.2.2 GlobalTime_Measurement

[SWS_StbM_00387] [

Name	MeasurementNotification_{TBName}		
Kind	RequiredPort	Interface	MeasurementNotification_{TB_Name}
Description	-		
Variation	(({ecuc(StbM/StbMSynchronizedTimeBase/StbMIsSystemWideGlobalTimeMaster)} == FALSE) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMAllowSystemWideGlobalTimeMaster)} == FALSE)) && ({ecuc(StbM/StbMGeneral/StbMTimeRecordingSupport)} == True) && ({ecuc(StbM/Stb MSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 32) && ({ecuc(StbM/Stb MSynchronizedTimeBase/StbMTimeRecording/StbMSyncTimeRecordBlockCallback)}==NULL) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMTimeRecording/StbMOffsetTimeRecordBlock Callback)}==NULL) TBName={ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		

](RS_TS_00034)

8.7.2.3 TimeBaseProviderNotification_Eth

[SWS_StbM_00458]{DRAFT} [

Name	TimeBaseProviderNotification_Eth_{TB_Name} (draft)		
Kind	RequiredPort	Interface	TimeBaseProviderNotification_Eth_{TB_Name}
Description	- Tags: atp.Status=draft		
Variation	({ecuc(StbM/StbMSynchronizedTimeBase/StbMTimeValidation)} != NULL) && ({ecuc(StbM/Stb MSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 16) && ({ecuc(StbM/Stb MSynchronizedTimeBase)} == {ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynSynchronized TimeBaseRef->StbMSynchronizedTimeBase)})) TBName={ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		

](RS_TS_00034)

8.7.2.4 TimeBaseProviderNotification_Fr

[SWS_StbM_00459]{DRAFT} [

Name	TimeBaseProviderNotification_Fr_{TB_Name} (draft)		
Kind	RequiredPort	Interface	TimeBaseProviderNotification_Fr_{TB_Name}
Description	– Tags: atp.Status=draft		
Variation	({ecuc(StbM/StbMSynchronizedTimeBase/StbMTimeValidation)} != NULL) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 16) && ({ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(FrTSyn/FrTSynGlobalTimeDomain/FrTSynSynchronizedTimeBaseRef->StbMSynchronizedTimeBase)}) TBName={ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		

] ([RS_TS_00034](#))

8.7.2.5 TimeBaseProviderNotification_Can

[SWS_StbM_00460]{DRAFT} [

Name	TimeBaseProviderNotification_Can_{TB_Name} (draft)		
Kind	RequiredPort	Interface	TimeBaseProviderNotification_Can_{TB_Name}
Description	– Tags: atp.Status=draft		
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMTimeValidation)} != NULL) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 16) && ({ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(CanTSyn/CanTSynGlobalTimeDomain/CanTSynSynchronizedTimeBaseRef->StbMSynchronizedTimeBase)}) TBName={ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		

] ([RS_TS_00034](#))

8.7.2.6 FreshnessManagement

[SWS_StbM_91019]{DRAFT} [

Name	FreshnessManagement (draft)		
Kind	RequiredPort	Interface	FreshnessManagement
Description	Port for the provision of freshness for StbM. Tags: atp.Status=draft		
Variation	({ecuc(StbM/StbMFreshnessValueInformation/StbMQueryFreshnessValue)} == SERVICE)		

] ([RS_TS_00039](#))

8.7.3 Sender-Receiver Interfaces

8.7.3.1 StatusNotification

[SWS_StbM_00286] [

Name	StatusNotification	
Comment	Notification about a Time Base related status change	
IsService	false	
Variation	–	
Data Elements	eventNotification	
	Type	StbM_TimeBaseNotificationType
	Variation	–

]([RS_TS_00035](#), [RS_TS_00016](#))

8.7.4 Client-Server-Interfaces

8.7.4.1 GlobalTime_Master

[SWS_StbM_00240] [

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/StbMSynchronizedTimeBaseIdentifier)} < 16) ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} > 31)		
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	–	
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/StbMSynchronizedTimeBaseIdentifier)} < 16) &&({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	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	SetOffset
Comment	Allows the Customers and the Timesync Modules to set the Offset Time.
Mapped to API	StbM_SetOffset
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} > 15 && {ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 32
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/StbMSynchronizedTimeBaselIdentifier)} < 32
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	–	
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	

|(RS_TS_00005, RS_TS_00035, RS_TS_00010, RS_TS_00013, RS_TS_00015, RS_TS_00011, RS_TS_20069, RS_TS_00038)

8.7.4.2 GlobalTime_Slave

[SWS_StbM_00247] [

Name	GlobalTime_Slave_{Name}		
Comment	–		
IsService	true		
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaselIdentifier)} < 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/StbMSynchronizedTimeBaseIdentifier)) < 16) &&({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	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
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	GetCurrentTimeExtended	
Comment	Returns a time value (Local Time Base derived from Global Time Base) in extended format. Tags: atp.Status=obsolete	
Mapped to API	StbM_GetCurrentTimeExtended	
Variation	{ecuc(StbM/StbMGeneral/StbMGetCurrentTimeExtendedAvailable)}	
Parameters	timeStamp	
	Type	StbM_TimeStampExtendedType
	Direction	OUT
	Comment	–
	Variation	–
	userData	
	Type	StbM_UserDataType
	Direction	OUT
	Comment	–
	Variation	–





Possible Errors	E_OK E_NOT_OK
------------------------	------------------

Operation	GetOffsetTimeRecordHead
Comment	Reads the header of the table with recorded measurement data belonging to the Offset Time Base
Mapped to API	StbM_GetOffsetTimeRecordHead
Variation	(({ecuc(StbM/StbMSynchronizedTimeBase/StbMIsSystemWideGlobalTimeMaster)} == FALSE) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMAllowSystemWideGlobalTimeMaster)} == FALSE)) && ({ecuc(StbM/StbMGeneral/StbMTimeRecordingSupport)} == True) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} > 15) && {ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 32 && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMTimeRecording/StbMOffsetTimeRecordBlock Callback)}==NULL)
Parameters	offsetRecordTableHead
Type	StbM_OffsetRecordTableHeadType
Direction	OUT
Comment	Header of the table
Variation	–
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/StbMSynchronizedTimeBaseIdentifier)} < 16) && ({ecuc(StbM/StbMSynchronizedTimeBase/StbMTimeRecording/StbMSyncTimeRecordBlock Callback)}==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 and, if called for an Offset Time Base, for the Offset Time Base and the underlying Synchronized 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	–
	offsetTimeBaseStatus	
	Type	StbM_TimeBaseStatusType
	Direction	OUT
	Comment	Status of the Offset 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/StbMSynchronizedTimeBaseIdentifier)} < 32	
Parameters	timeJump	
	Type	StbM_TimeDiffType
	Direction	OUT
	Comment	Time leap value
	Variation	–
Possible Errors	E_OK E_NOT_OK	

|(RS_TS_00005, RS_TS_00035, RS_TS_00014, RS_TS_00017, RS_TS_00034, RS_TS_20069)

8.7.4.3 StartTimer

[SWS_StbM_00409] [

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	

]([RS_TS_00017](#))

8.7.4.4 TimeNotification

[SWS_StbM_00275] [

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	

]([RS_TS_00035](#), [RS_TS_00017](#))

8.7.4.5 MeasurementNotification

[SWS_StbM_00339] [

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/StbMSynchronizedTimeBaseIdentifier)} < 32) TBName={ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	SetOffsetTimeRecordTable		
Comment	Provides to the recorded snapshot data Block of the table belonging to the Offset Time Base.		
Mapped to API	OffsetTimeRecordBlockCallback<TimeBase>		
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} > 15 && {ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 32		
Parameters	offsetRecordTableBlock		
	Type	StbM_OffsetRecordTableBlockType	
	Direction	IN	
	Comment	Header of the table	
	Variation	-	
Possible Errors	E_OK E_NOT_OK		

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/StbMSynchronizedTimeBaseIdentifier)} < 16		
Parameters	syncRecordTableBlock		
	Type	StbM_SyncRecordTableBlockType	
	Direction	IN	
	Comment	Block of the table	
	Variation	-	
Possible Errors	E_OK E_NOT_OK		

]([RS_TS_00034](#))

8.7.4.6 TimeBaseProviderNotification_Eth

[SWS_StbM_00461]{DRAFT} [

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/StbMSynchronizedTimeBaseIdentifier)} < 16} && {{ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSyn SynchronizedTimeBaseRef->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	<pre> ({{ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynPortRole/EthTSynGlobalTime Master)}!=NULL) </pre>		
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	

](RS_TS_00034)

8.7.4.7 TimeBaseProviderNotification_Fr

[SWS_StbM_00462]{DRAFT} [

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/StbMSynchronizedTimeBaseIdentifier)} < 16} && {{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	

](RS_TS_00034)

8.7.4.8 TimeBaseProviderNotification_Can

[SWS_StbM_00463]{DRAFT} [

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/StbMSynchronizedTimeBaseIdentifier)} < 16) && ({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	–	





	Variation	–
Possible Errors	E_OK E_NOT_OK	

|(RS_TS_00034)

8.7.4.9 FreshnessManagement

[SWS_StbM_91026]{DRAFT} [

Name	FreshnessManagement (draft)		
Comment	Freshness Management for StbM Tags: atp.Status=draft		
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-IPDU	
	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 PDU		





	Variation	–
	freshnessValue	
	Type	StbM_FreshnessArrayType
	Direction	OUT
	Comment	The freshness value for this PDU
	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-IPDU.	
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-IPDU
	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	$\{ \{ \text{ecuc}(\text{CanTSyn}/\text{CanTSynGlobalTimeDomain}/\text{CanTSynGlobalTimeMaster}) \} \neq \text{NULL} \} \parallel$ $\{ \{ \text{ecuc}(\text{FrTSyn}/\text{FrTSynGlobalTimeDomain}/\text{FrTSynGlobalTimeMaster}) \} \neq \text{NULL} \} \parallel \{ \{ \text{ecuc}(\text{EthTSyn}/\text{EthTSynGlobalTimeDomain}/\text{EthTSynPortRole}/\text{EthTSynGlobalTimeMaster}) \} \neq \text{NULL} \}$	
Parameters	freshnessValueId	
	Type	uint16
	Direction	IN
	Comment	Identifier of the freshness
Variation	–	
Possible Errors	E_OK	

|(RS_TS_00039)

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] [

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		

](RS_TS_00034)

8.7.5.2 StbM_SynchronizedTimeBaseType

[SWS_StbM_00142] [

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		

](SRS_BSW_00305, RS_TS_00005, RS_TS_00032, RS_TS_00035)

8.7.5.3 StbM_TimeBaseStatusType

[SWS_StbM_00239] [

Name	StbM_TimeBaseStatusType			
Kind	Bitfield			
Derived from	uint8			
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	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
Description	Bit 1, 6, and 7 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 of the bits TIMEOUT, TIMELEAP_FUTURE, TIMELEAP_PAST and SYNC_TO_GATEWAY can only be set if the GLOBAL_TIME_BASE bit is set.			
Variation	-			
Available via	Rte_StbM_Type.h			

](RS_TS_00009)

8.7.5.4 StbM_TimeStampShortType

[SWS_StbM_00482] [

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	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 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

]([RS_TS_00034](#))

8.7.5.5 StbM_TimeStampType

[SWS_StbM_00241] [

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
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	

]([RS_TS_00036](#)) 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)

8.7.5.6 StbM_TimeStampExtendedType

[SWS_StbM_00242]{OBSOLETE} [

Name	StbM_TimeStampExtendedType (obsolete)	
Kind	Structure	
Elements	timeBaseStatus	
	Type	StbM_TimeBaseStatusType
	Comment	Status of the Time Base
	nanoseconds	





	Type	uint32
	Comment	Nanoseconds part of the time
		seconds
	Type	uint64
	Comment	48 bit 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. Tags: atp.Status=obsolete	
Variation	-	
Available via	Rte_StbM_Type.h	

|(RS_TS_00036) 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)

8.7.5.7 StbM_TimeTupleType

[SWS_StbM_91013] [

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
	Comment	Status of the Time Base
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	

|(RS_TS_00005)

8.7.5.8 StbM_TimeDiffType

[SWS_StbM_00300] [

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		

](RS_TS_00010)

8.7.5.9 StbM_RateDeviationType

[SWS_StbM_00301] [

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		

](RS_TS_00017)

8.7.5.10 StbM_CloneConfigType

[SWS_StbM_91011] [

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





Description	Bitfield to configure the cloning process. Bit 3 .. 7 are always 0 (reserved for future usage).
Variation	–
Available via	Rte_StbM_Type.h

|(RS_TS_00038)

8.7.5.11 StbM_UserDataType

[SWS_StbM_00243] [

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		

|(RS_TS_00014, RS_TS_00015)

8.7.5.12 StbM_CustomerIdType

[SWS_StbM_00288] [

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		

|(RS_TS_00035, RS_TS_00016, RS_TS_00017)

8.7.5.13 StbM_TimeBaseNotificationType

[SWS_StbM_00287] [

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
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			

] ([RS_TS_00035](#), [RS_TS_00016](#))

8.7.5.14 StbM_SyncRecordTableHeadType

[SWS_StbM_00331] [

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	

] ([RS_TS_00034](#))

8.7.5.15 StbM_SyncRecordTableBlockType

[SWS_StbM_00332] [

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
Description	Synchronized Time Base Record Table Block	
Variation	-	
Available via	Rte_StbM_Type.h	

](RS_TS_00034)

8.7.5.16 StbM_OffsetRecordTableHeadType

[SWS_StbM_00333] [

Name	StbM_OffsetRecordTableHeadType	
Kind	Structure	
Elements	OffsetTimeDomain	
	Type	uint8
	Comment	Time Domain 16..31
Description	Offset Time Base Record Table Header	
Variation	-	
Available via	Rte_StbM_Type.h	

](RS_TS_00034)

8.7.5.17 StbM_OffsetRecordTableBlockType

[SWS_StbM_00334] [

Name	StbM_OffsetRecordTableBlockType	
Kind	Structure	
Elements	GlbSeconds	
	Type	uint32
	Comment	Seconds of the Offset Time Base
	GlbNanoSeconds	
	Type	uint32
	Comment	Nanoseconds of the Offset Time Base
	TimeBaseStatus	





	Type	StbM_TimeBaseStatusType
	Comment	Time Base Status of the Local Time Base directly after synchronization with the Global Time Base
Description	Offset Time Base Record Table Block	
Variation	-	
Available via	Rte_StbM_Type.h	

]([RS_TS_00034](#))

8.7.5.18 StbM_MasterConfigType

[SWS_StbM_91001] [

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		

]([RS_TS_00029](#))

8.7.5.19 StbM_EthTimeMasterMeasurementType

[SWS_StbM_00504] [

Name	StbM_EthTimeMasterMeasurementType		
Kind	Structure		
Elements	sequenceld		
	Type	uint16	
	Comment	sequenceld 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	

] (RS_TS_00034)

8.7.5.20 StbM_FrTimeMasterMeasurementType

[SWS_StbM_00505] [

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	

|(RS_TS_00034)

8.7.5.21 StbM_CanTimeMasterMeasurementType

[SWS_StbM_00511] [

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	

|(RS_TS_00034)

8.7.5.22 StbM_EthTimeSlaveMeasurementType

[SWS_StbM_00506] [

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	

]([RS_TS_00034](#))

8.7.5.23 StbM_FrTimeSlaveMeasurementType

[SWS_StbM_00507] [

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

]([RS_TS_00034](#))

8.7.5.24 StbM_CanTimeSlaveMeasurementType

[SWS_StbM_00510] [

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)
	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

|(RS_TS_00034)

8.7.5.25 StbM_PdelayInitiatorMeasurementType

[SWS_StbM_00508] [

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	

]([RS_TS_00034](#))

8.7.5.26 StbM_PdelayResponderMeasurementType

[SWS_StbM_00509] [

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	

]([RS_TS_00034](#))

8.7.5.27 StbM_TimeSyncType

[SWS_StbM_91009] [

Name	StbM_TimeSyncType		
Kind	Enumeration		
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		

]([RS_TS_20069](#))

8.7.5.28 StbM_ProtocolParamType

[SWS_StbM_91010] [

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	

]([RS_TS_20069](#))

8.7.5.29 StbM_FreshnessArrayType

[SWS_StbM_91021]{DRAFT} [

Name	StbM_FreshnessArrayType (draft)		
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. Tags: atp.Status=draft		
Variation	(ecuc(StbM/StbMFreshnessValueInformation/StbMQueryFreshnessValue) == SERVICE)		
Available via	Rte_StbM_Type.h		

]([RS_TS_00039](#))

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

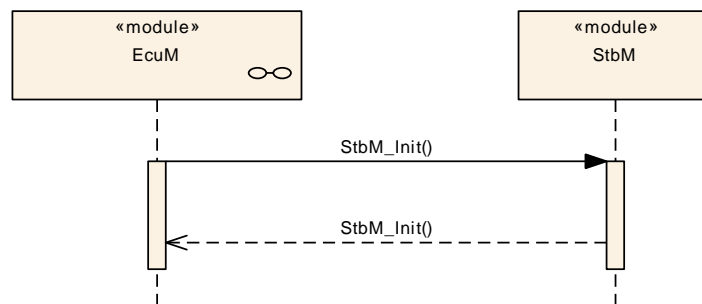


Figure 9.1: StbM Initialization

9.2 Immediate Time Synchronisation

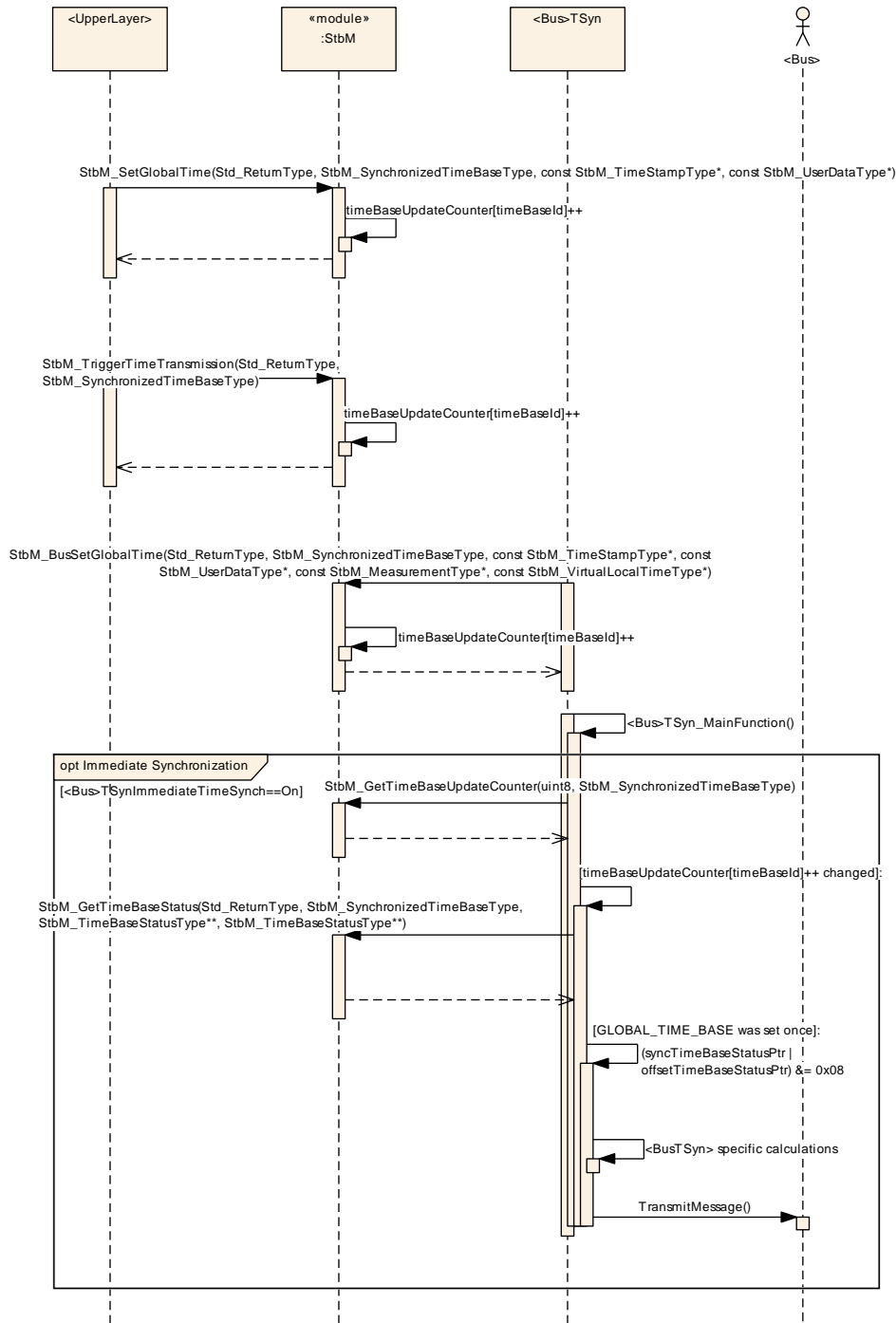


Figure 9.2: Immediate time synchronization sequence (StbM API)

9.3 Explicit synchronization of OS ScheduleTable

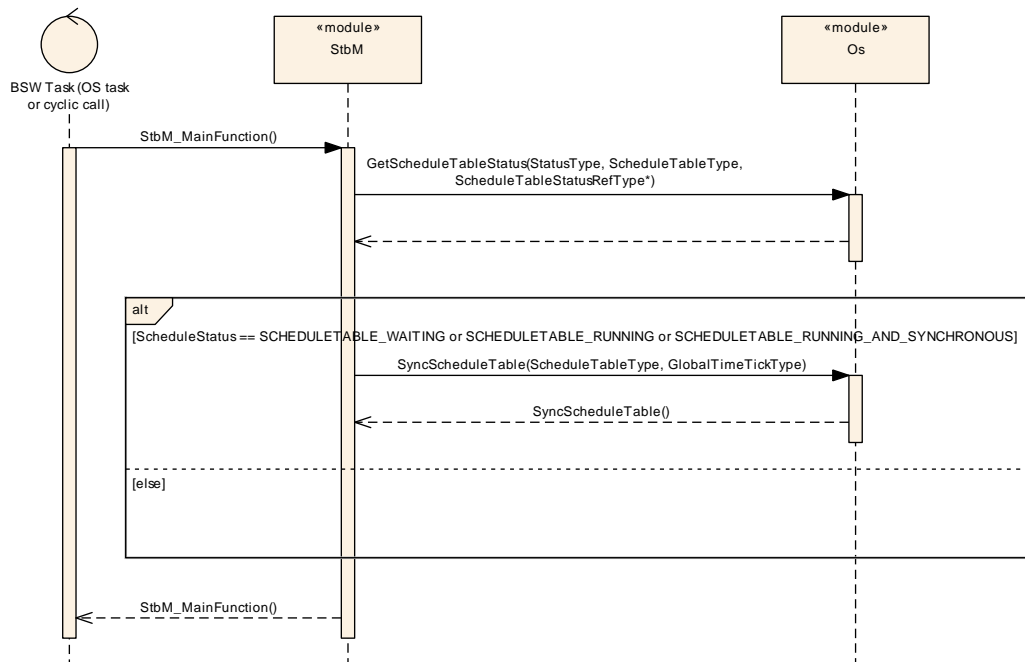


Figure 9.3: Explicit synchronization of OS Schedule Table

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 [3, 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

SWS Item	[ECUC_StbM_00065]
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 Tags: atp.Status=draft
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.

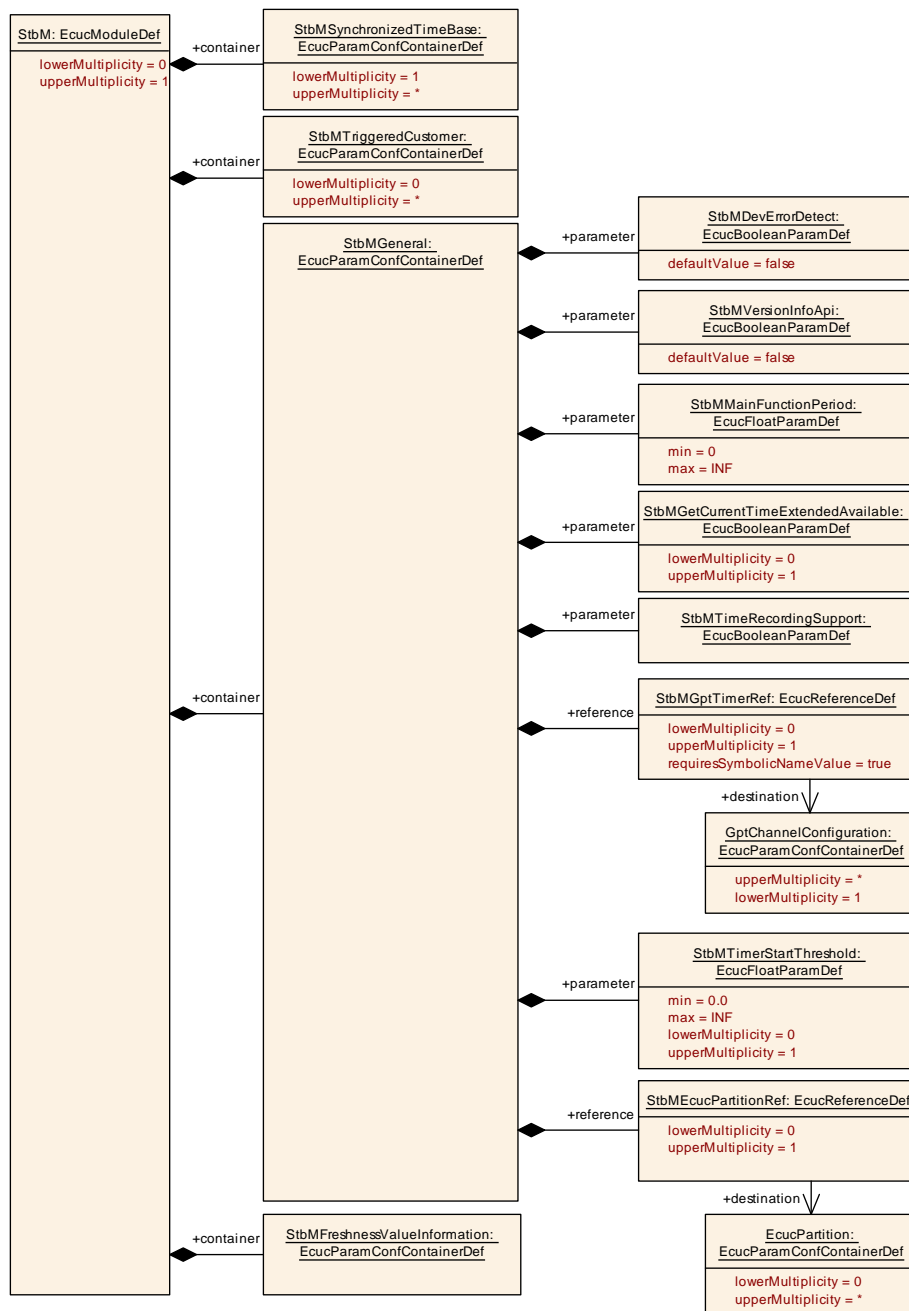


Figure 10.1

10.2.2 StbMGeneral

SWS Item	[ECUC_StbM_00002]
Container Name	StbMGeneral
Parent Container	StbM
Description	This container holds the general parameters of the Synchronized Time-base Manager
Configuration Parameters	

SWS Item	[ECUC_StbM_00012]		
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		

SWS Item	[ECUC_StbM_00032] (Obsolete)		
Parameter Name	StbMGetCurrentTimeExtendedAvailable		
Parent Container	StbMGeneral		
Description	This allows to define whether an additional variant of the API GetCurrentTime with a 64 bit argument is provided. Tags: atp.Status=obsolete		
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		

SWS Item	[ECUC_StbM_00027]
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		

SWS Item	[ECUC_StbM_00038]		
Parameter Name	StbMTimeRecordingSupport		
Parent Container	StbMGeneral		
Description	Enables/Disables the usage of the recording functionality for Synchronized and Offset timebases 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		

SWS Item	[ECUC_StbM_00063]		
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		

SWS Item	[ECUC_StbM_00013]		
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		

SWS Item	[ECUC_StbM_00069]		
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		

SWS Item	[ECUC_StbM_00039]		
Parameter Name	StbMGptTimerRef		
Parent Container	StbMGeneral		
Description	<p>This represents an optional sub-container in case any Time Notification Customer is configured.</p> <p>The designated GPT timer has to be configured to have a tick duration of one micro second.</p>		
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		

No Included Containers

10.2.3 StbMSynchronizedTimeBase

SWS Item	[ECUC_StbM_00003]		
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			

SWS Item	[ECUC_StbM_00066]		
Parameter Name	StbMAllowSystemWideGlobalTimeMaster		
Parent Container	StbMSynchronizedTimeBase		
Description	<p>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.</p> <p>The Global Time Master functionality behind the service ports/interfaces has to be enabled/disabled separately via parameter StbMIsSystemWideGlobalTimeMaster.</p>		
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		

SWS Item	[ECUC_StbM_00037]		
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		

SWS Item	[ECUC_StbM_00086]		
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		

SWS Item	[ECUC_StbM_00036]		
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		

SWS Item	[ECUC_StbM_00068]		
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		

SWS Item	[ECUC_StbM_00046]		
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.		

SWS Item	[ECUC_StbM_00045]		
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		

SWS Item	[ECUC_StbM_00031]		
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		

SWS Item	[ECUC_StbM_00021]		
Parameter Name	StbMSynchronizedTimeBaseIdentifier		
Parent Container	StbMSynchronizedTimeBase		
Description	Identification of a Synchronized TimeBase via a unique identifier. Range: <ul style="list-style-type: none"> • 0 .. 15: Synchronized Time Bases • 16 .. 31: Offset Time Bases • 32 .. 127: 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		

SWS Item	[ECUC_StbM_00028]		
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		

SWS Item	[ECUC_StbM_00041]		
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		

SWS Item	[ECUC_StbM_00042]		
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		

SWS Item	[ECUC_StbM_00030]		
Parameter Name	StbMOffsetTimeBase		
Parent Container	StbMSynchronizedTimeBase		
Description	This is the reference to the Synchronized Time-Base this Offset Time-Base is based on. This reference makes the containing StbMSynchronizedTimeBase an Offset Time-Base.		
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		

SWS Item	[ECUC_StbM_00074]		
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		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
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.

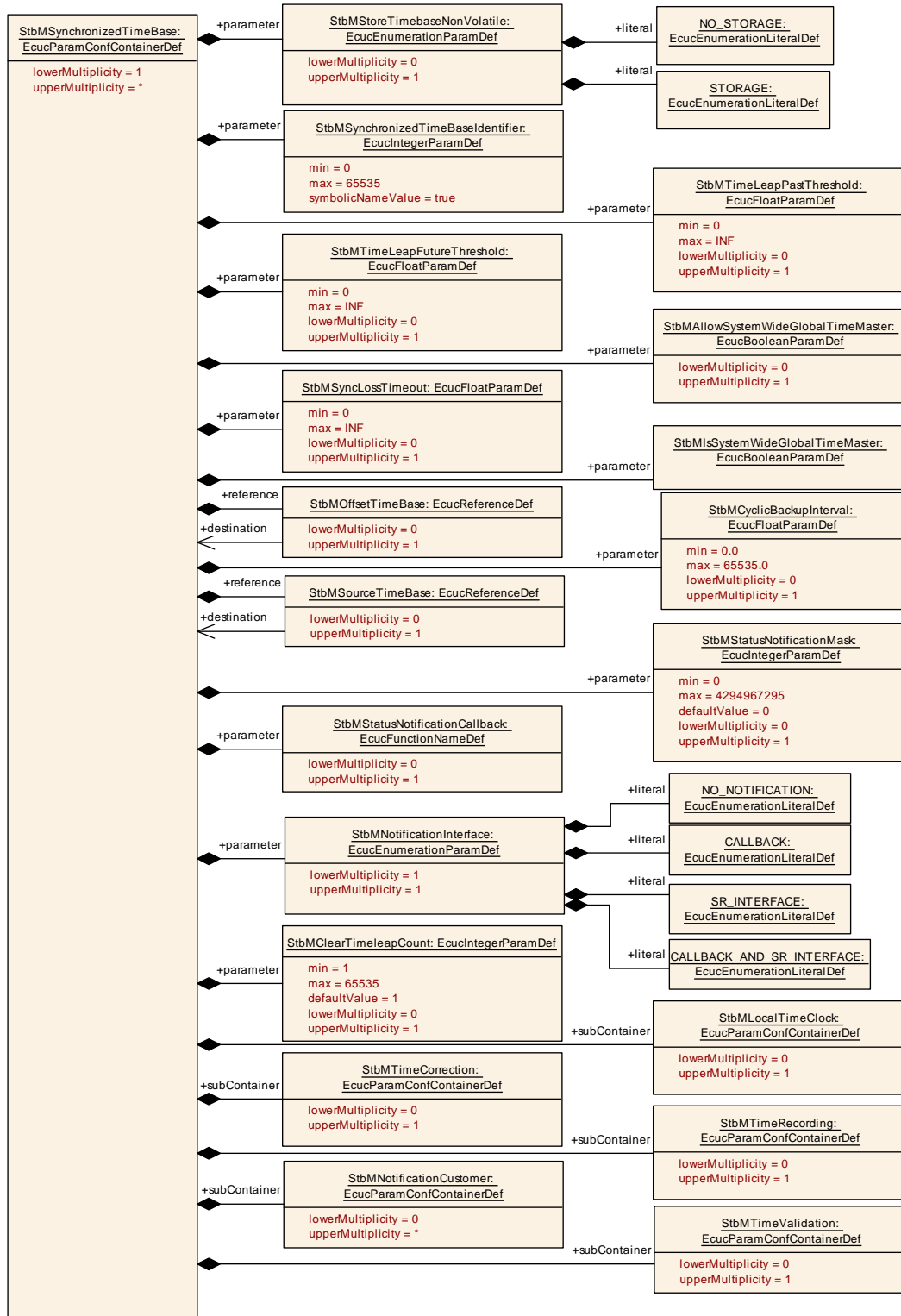


Figure 10.2

10.2.4 StbMTimeCorrection

SWS Item	[ECUC_StbM_00048]		
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			

SWS Item	[ECUC_StbM_00043]		
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		

SWS Item	[ECUC_StbM_00044]		
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		

SWS Item	[ECUC_StbM_00057]		
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		

SWS Item	[ECUC_StbM_00056]		
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		

SWS Item	[ECUC_StbM_00054]		
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		

SWS Item	[ECUC_StbM_00055]		
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		

No Included Containers

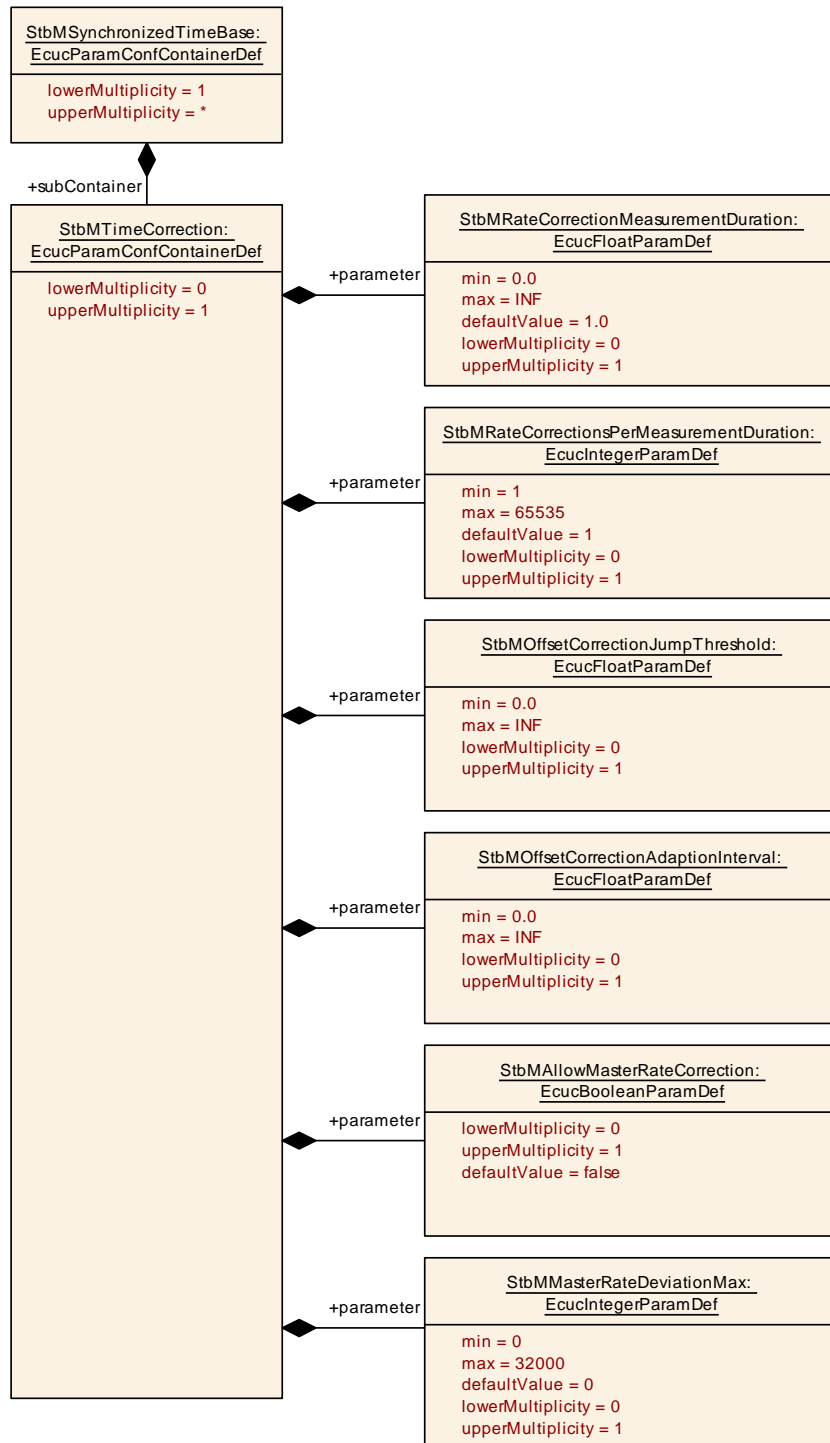


Figure 10.3

10.2.5 StbMLocalTimeClock

SWS Item	[ECUC_StbM_00047]		
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			

SWS Item	[ECUC_StbM_00051]		
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		

SWS Item	[ECUC_StbM_00052]		
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		

SWS Item	[ECUC_StbM_00053]		
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		

No Included Containers

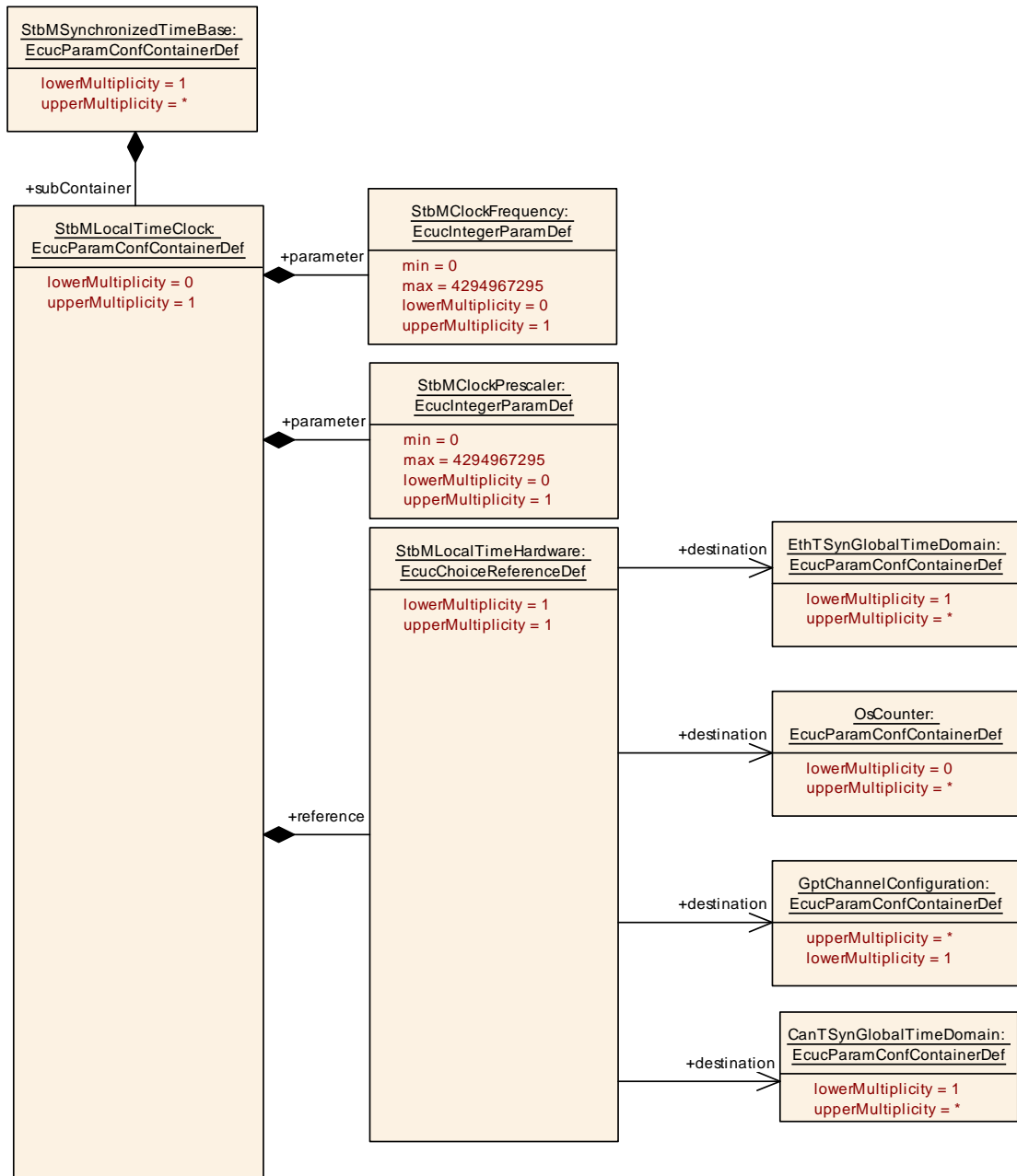


Figure 10.4

10.2.6 StbMTimeRecording

SWS Item	[ECUC_StbM_00049]
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			

SWS Item	[ECUC_StbM_00061]		
Parameter Name	StbMOffsetTimeRecordBlockCallback		
Parent Container	StbMTimeRecording		
Description	Name of the customer specific callback function, which shall be called, if a measurement data for a Offset 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		

SWS Item	[ECUC_StbM_00059]		
Parameter Name	StbMOffsetTimeRecordTableBlockCount		
Parent Container	StbMTimeRecording		
Description	Represents the number of Blocks used for queing time measurement events for the Offset 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		

SWS Item	[ECUC_StbM_00060]		
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		

SWS Item	[ECUC_StbM_00058]		
Parameter Name	StbMSyncTimeRecordTableBlockCount		
Parent Container	StbMTimeRecording		
Description	Represents the number of Blocks used for queing 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		

No Included Containers

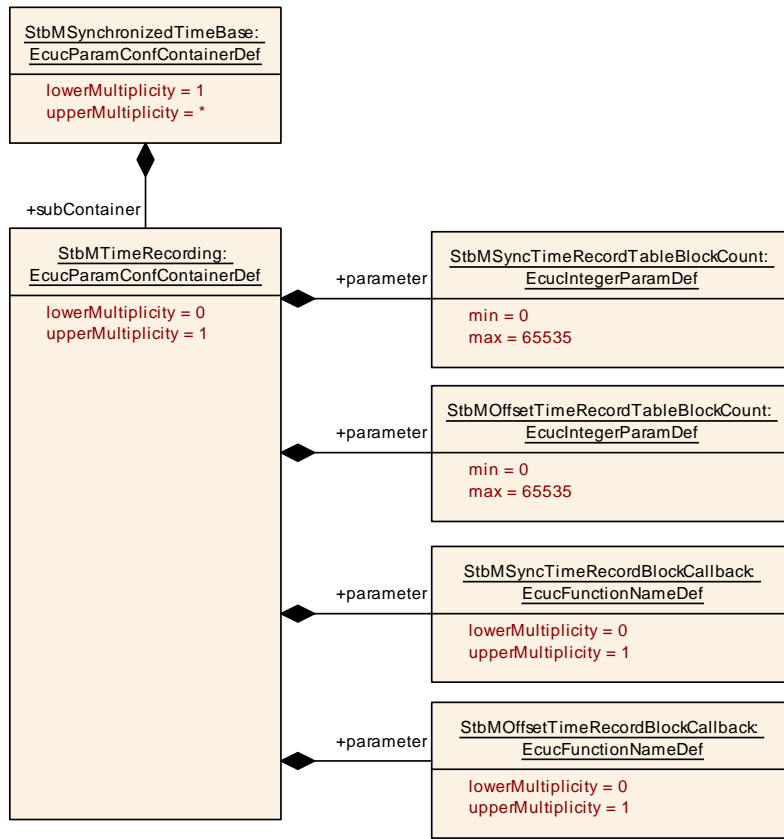


Figure 10.5

10.2.7 StbMTimeValidation

SWS Item	[ECUC_StbM_00072]		
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			

SWS Item	[ECUC_StbM_00073]		
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		

No Included Containers

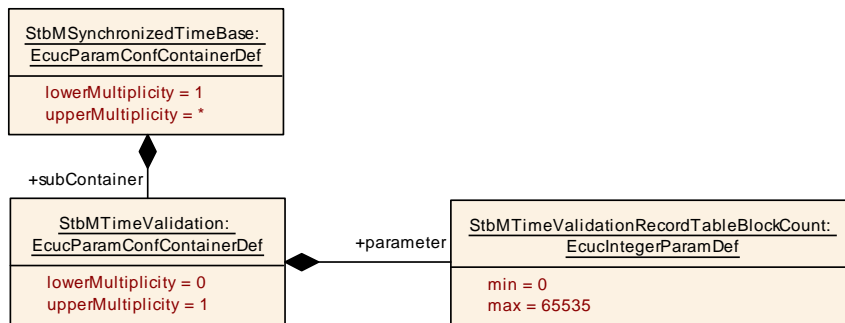


Figure 10.6

10.2.8 StbMNotificationCustomer

SWS Item	[ECUC_StbM_00050]		
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			

SWS Item	[ECUC_StbM_00062]		
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		

SWS Item	[ECUC_StbM_00064]		
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		

No Included Containers

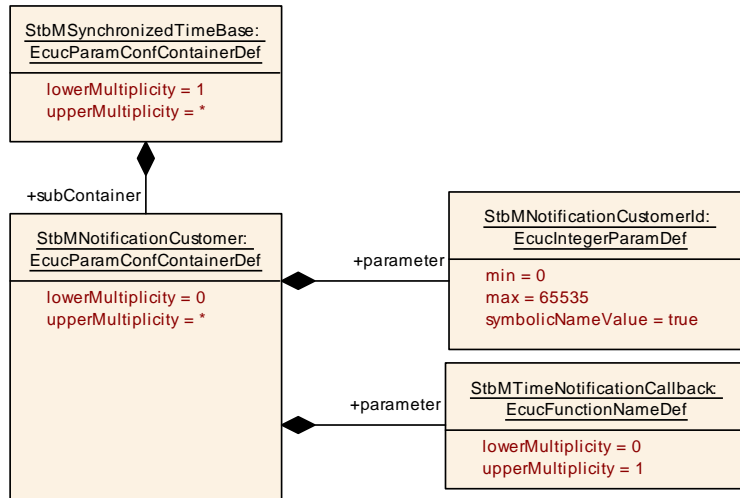


Figure 10.7

10.2.9 StbMTriggeredCustomer

SWS Item	[ECUC_StbM_00004]
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			

SWS Item	[ECUC_StbM_00020]		
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		

SWS Item	[ECUC_StbM_00007]		
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		

SWS Item	[ECUC_StbM_00010]		
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		

No Included Containers

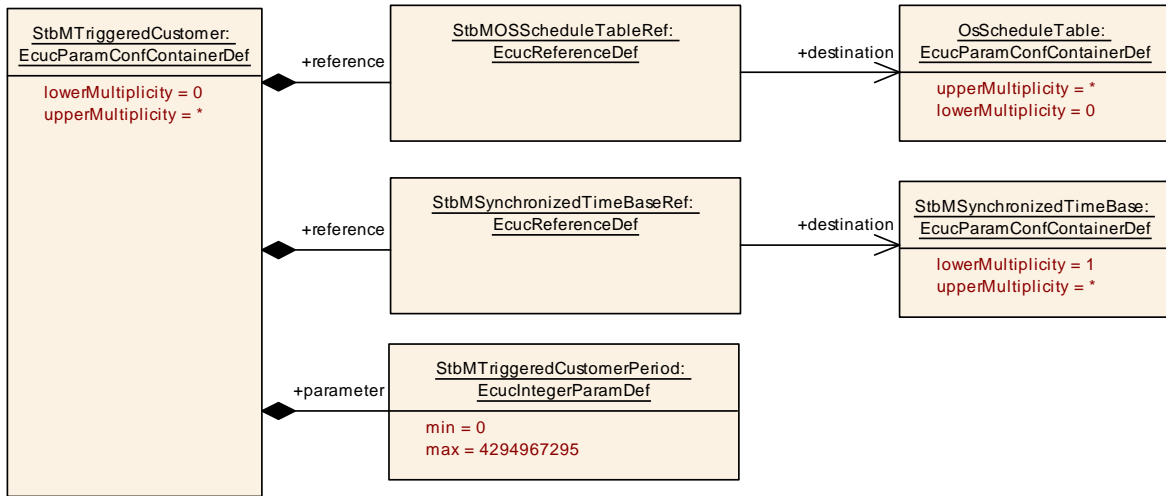


Figure 10.8

10.2.10 StbMFreshnessValueInformation

SWS Item	[ECUC_StbM_00075]
Container Name	StbMFreshnessValueInformation
Parent Container	StbM
Description	Container with the Freshness Value configurations Tags: atp.Status=draft
Configuration Parameters	

SWS Item	[ECUC_StbM_00081]		
Parameter Name	StbMGetRxFreshnessValueFuncName		
Parent Container	StbMFreshnessValueInformation		
Description	Function pointer to call within StbM_GetRxFreshness() context. Tags: atp.Status=draft		
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		

SWS Item	[ECUC_StbM_00080]		
Parameter Name	StbMGetTxConfFreshnessValueFuncName		
Parent Container	StbMFreshnessValueInformation		
Description	Function pointer to call within StbM_SPduTxConfirmation() context. Tags: atp.Status=draft		
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		

SWS Item	[ECUC_StbM_00078]		
Parameter Name	StbMGetTxFreshnessValueFuncName		
Parent Container	StbMFreshnessValueInformation		
Description	Function pointer to call within StbM_GetTxFreshness() context. Tags: atp.Status=draft		
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		

SWS Item	[ECUC_StbM_00079]		
Parameter Name	StbMGetTxTruncFreshnessValueFuncName		
Parent Container	StbMFreshnessValueInformation		
Description	Function pointer to call within StbM_GetTxFreshnessTruncData() context. Tags: atp.Status=draft		
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		

SWS Item	[ECUC_StbM_00076]		
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). Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CFUNC	The StbM queries the freshness for every PDU using a C function API. Tags: atp.Status=draft	
	NONE	The StbM does not use the freshness value. Tags: atp.Status=draft	
	SERVICE	The StbM queries the freshness for every PDU using the Rte service port Freshness Management. Tags: atp.Status=draft	
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		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
StbMFreshnessValue	0..*	Container with the Freshness Value configurations Tags: atp.Status=draft

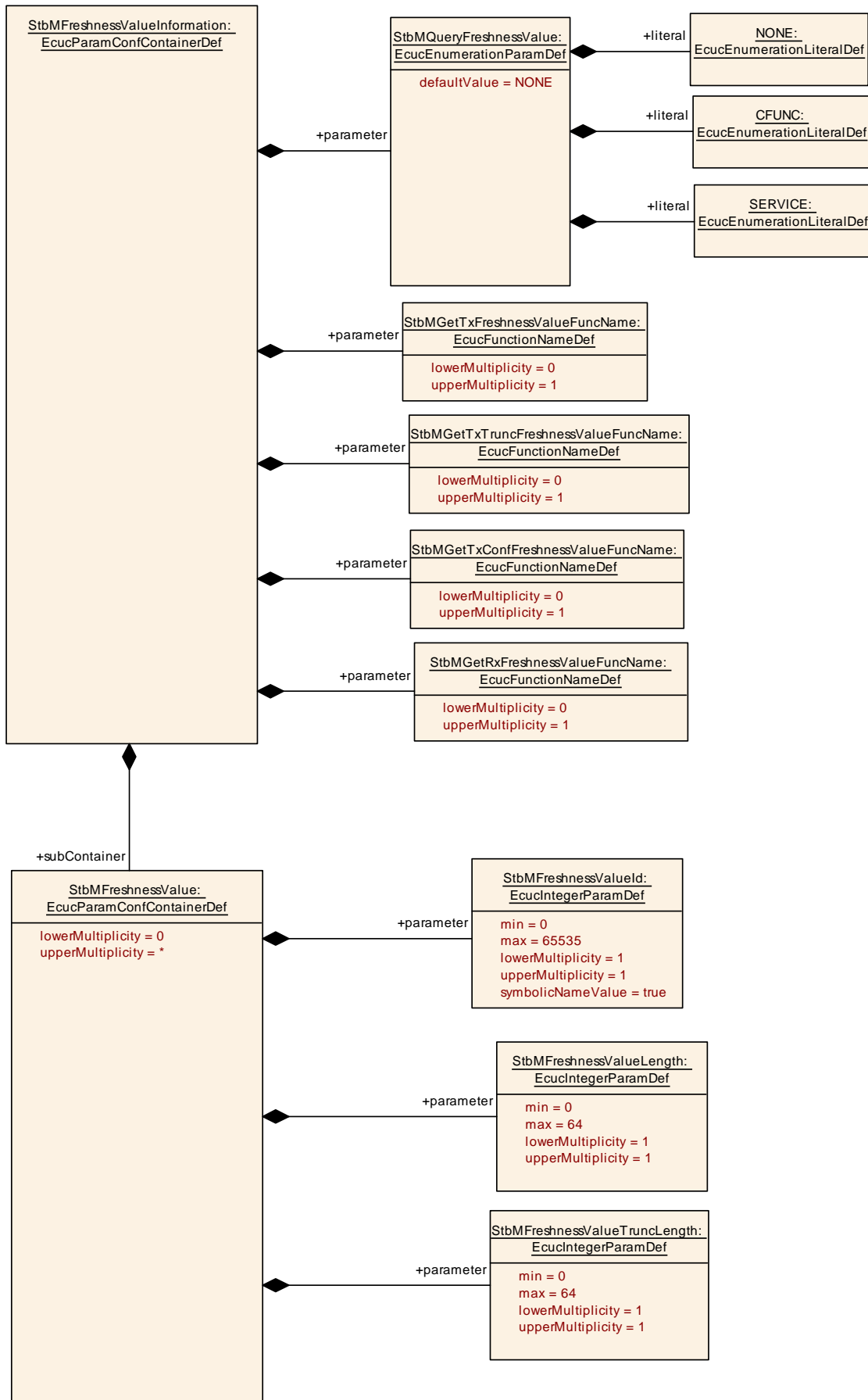


Figure 10.9

10.2.11 StbMFreshnessValue

SWS Item	[ECUC_StbM_00082]		
Container Name	StbMFreshnessValue		
Parent Container	StbMFreshnessValueInformation		
Description	Container with the Freshness Value configurations 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			

SWS Item	[ECUC_StbM_00083]		
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. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_StbM_00084]		
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. Tags: atp.Status=draft		
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		

SWS Item	[ECUC_StbM_00085]		
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. Tags: atp.Status=draft		
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		

No Included Containers

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_00003] [The parameter [StbMOffsetTimeBase](#) shall only be valid for [StbMSynchronizedTimeBaseIdentifier](#) 16 to 31.]([RS_TS_00012](#), [RS_TS_00013](#))

[SWS_StbM_CONSTR_00004]{DRAFT} [If parameter [StbMSynchronizedTimeBaseIdentifier](#) is greater than 15, multiplicity of parameter [StbMCyclicBackupInterval](#) shall be set to 0.]([RS_TS_00024](#))

10.4 Published Information

For details refer to the chapter 10.3 "Published Information" in [3, SWS BSW General].

A Not applicable requirements

[SWS_StbM_NA_00140] [These requirements are not applicable to this specification.] ([RS_TS_00027](#), [RS_TS_20031](#), [RS_TS_20032](#), [RS_TS_20033](#), [RS_TS_20034](#), [RS_TS_20035](#), [RS_TS_20036](#), [RS_TS_20037](#), [RS_TS_20038](#), [RS_TS_20039](#), [RS_TS_20040](#), [RS_TS_20041](#), [RS_TS_20042](#), [RS_TS_20043](#), [RS_TS_20044](#), [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_20063](#), [RS_TS_20066](#), [RS_TS_20068](#), [RS_TS_20071](#), [RS_TS_20072](#), [RS_TS_20073](#), [RS_TS_20074](#), [SRS_BSW_00005](#), [SRS_BSW_00006](#), [SRS_BSW_00007](#), [SRS_BSW_00009](#), [SRS_BSW_00010](#), [SRS_BSW_00160](#), [SRS_BSW_00161](#), [SRS_BSW_00162](#), [SRS_BSW_00164](#), [SRS_BSW_00168](#), [SRS_BSW_00170](#), [SRS_BSW_00304](#), [SRS_BSW_00307](#), [SRS_BSW_00308](#), [SRS_BSW_00309](#), [SRS_BSW_00312](#), [SRS_BSW_00314](#), [SRS_BSW_00325](#), [SRS_BSW_00328](#), [SRS_BSW_00334](#), [SRS_BSW_00336](#), [SRS_BSW_00341](#), [SRS_BSW_00342](#), [SRS_BSW_00344](#), [SRS_BSW_00347](#), [SRS_BSW_00353](#), [SRS_BSW_00375](#), [SRS_BSW_00378](#), [SRS_BSW_00398](#), [SRS_BSW_00399](#), [SRS_BSW_00400](#), [SRS_BSW_00404](#), [SRS_BSW_00405](#), [SRS_BSW_00413](#), [SRS_BSW_00415](#), [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](#), [SRS_BSW_00439](#), [SRS_BSW_00440](#), [SRS_BSW_00453](#))