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

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

Document Change History

Date	Release	Changed by	Change Description
2017-12-08	4.3.1	AUTOSAR Release Management	• 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	• 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	• Config parameter argument added to StbM_Init • StbM_TimeStampRawType changed uint32 • StbM_BusSetGlobalTime allow NULL as userDataPtr • 'const' added to input arguments passed by pointer • Debugging support marked as obsolete

Document Change History

Date	Release	Changed by	Change Description
2014-10-31	4.2.1	AUTOSAR Release Management	• 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	• Clarification on Autonomous Time Maintenance
2013-10-31	4.1.2	AUTOSAR Release Management	• 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	• 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	• Added functionality for absolute time provision

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2010-09-30	3.1.5	AUTOSAR Administration	• 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	• Initial Release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and Functional Overview

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 AUTOSAR Timing Extensions [10] and must be fulfilled independently of the actual deployment of the software components.

  Typcial examples of this use case are the sensor data read out or synchronous actuator triggering by different RunnableEntities.

- **Provision of absolute time value**

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

  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.

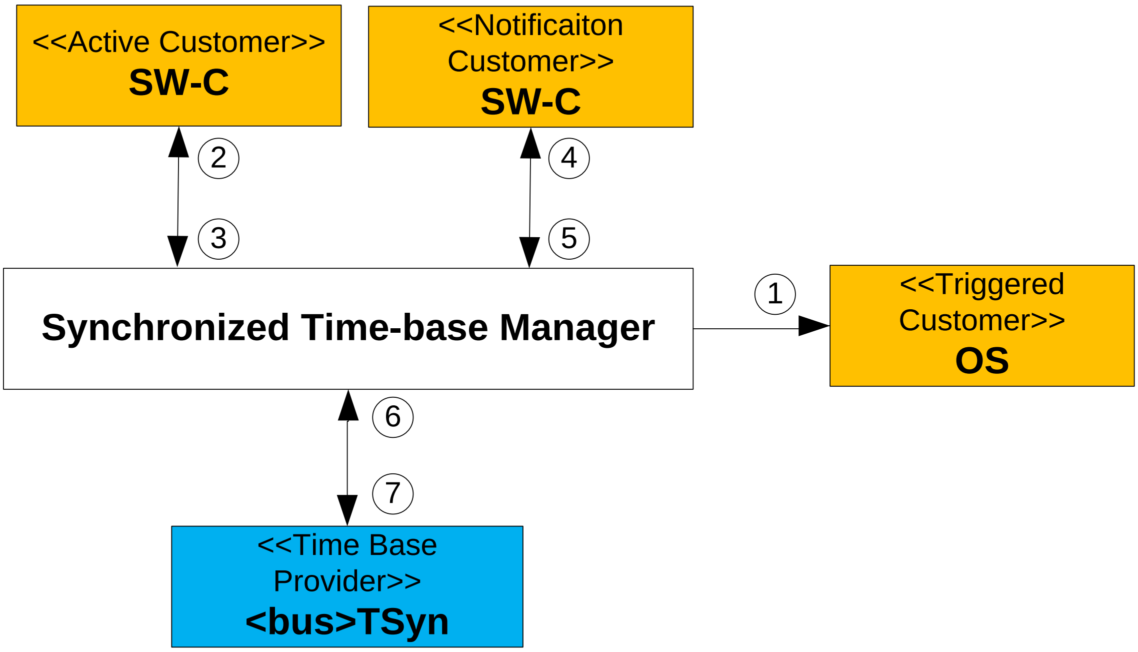


Figure 1: Synchronized Time-Base Manager as broker

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 distingushed:

a) Triggered customer
b) Active customer
c) Notification customer

For a detailed description of those three types refer to chapter 2.2.8.

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 Synchronized Time Bases between the updates of 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 a Virtual Local Time (Most likely 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 Virtual Local Time.

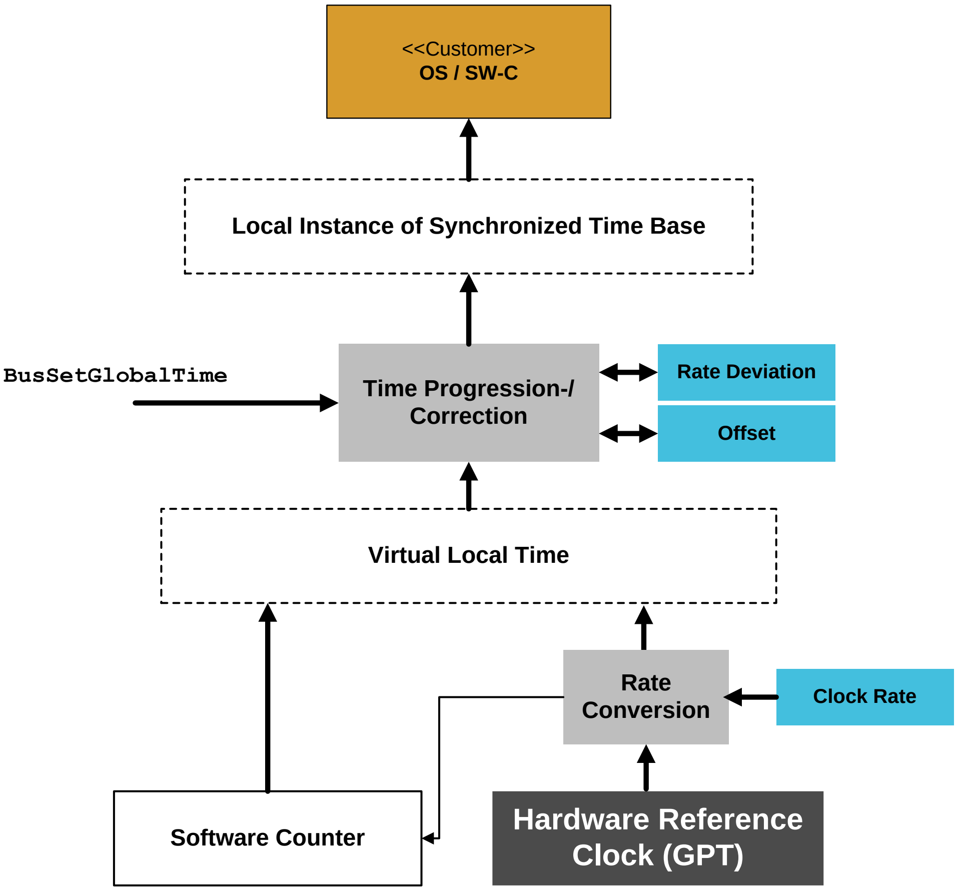


Figure 2: Abstract Working Principle of the Synchronized Time-Base Manager

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 8.2.
- For the interaction with other BSW modules, respective interfaces are specified in chapter 8.1.3.

2 Acronyms, Abbreviations, and Definitions

Acronyms, abbreviations, and definitions, which have a StbM local scope and therefore are not contained in the AUTOSAR glossary, appear in this local glossary.

2.1 Acronyms and Abbreviations

Abbreviation / Acronym:	Description
(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
FRC	Free running counter
FrTSyn	Time Synchronization Provider module for FlexRay
FUP message	Follow-Up message
GM(C)	Grand Master (Clock)
GTS	Global Time Synchronization
OFNS message	Offset adjustment message
OFS message	Offset Synchronization message
PTP	Precision Time Protocol
StbM	Synchronized Time-Base Manager
SYNC message	Time Synchronization message
TG	Time Gateway
Timesync	Time Synchronization
TS	Time Slave
TSD	Time Sub-domain

2.2 Definitions

2.2.1 Clock

Definition: A Clock references to a time capable hardware part of a microcontroller.

2.2.2 Global Time Master

Definition: A Global Time Master is the global owner and origin for a certain Time Base and on the top of the Time Base hierarchy for that Time Base.

2.2.3 Synchronized Time Base

Definition: A Synchronized Time Base is a Time Base existing at a processing entity (actor / processor / node of a distributed system) that is synchronized with Time Bases at different processing entities. A Synchronized Time Base can be achieved by time protocols or time agreement protocols that derive the Synchronized Time Base in a defined way from one or more physical Time Bases. Examples are the network time protocol (NTP) and FlexRay time agreement protocol.

The synchronization will apply to the clock rate and optionally apply also to the clock absolute value.

A Synchronized Time Base allows synchronized action of the processing units. Synchronized Time Bases are often called “Global Time”.

More than one Synchronized Time Base can exist at one processing unit, e.g. a FlexRay node will have the Synchronized Time Base retrieved from the FlexRay time agreement protocol in the network cluster but might also have a Synchronized Time Base derived from the time provided by a UTC time server (which is based on a set of atomic clocks). Both Synchronized Time Bases will probably have slightly different rate, and there is no relationship defined between their absolute values.

2.2.4 Time Base

Definition: A Time Base is a unique time entity characterized by:

- Progression of time, which denotes how time progresses, i.e. the rate (i.e. the rate is derived from a local quartz oscillator) and absolute changes of the time value at certain point in times (e.g. effects of offset correction in FlexRay).
- Ownership, which denotes who is the owner of the time base. A distributed FlexRay Time Base e.g. has multiple owners and the progression of time with respect to rate and offset corrections is a result of involving a subset of FlexRay nodes.
- Reference to the physical world, i.e. whether the Time Base is a relative Time Base counting local operation time of an ECU or representing an absolute time like UTC.
  A Time Base can have more than one reference, e.g. it can be a relative time which in combination with an offset value also represents an absolute time.

Examples of Time Bases in vehicles are:

- Absolute, which is based on a GPS based time

- Relative, which represents the accumulated overall operating time of a vehicle, i.e. this Time Base does not start with a value of zero whenever the vehicle starts operating
- Relative, starting at zero when the ECU begins its operation

A Time Base implies the availability of a Clock.

Special case “Pure Local Time Base”:
A Pure Local Time Base is a Time Base with a local scope as it is neither propagated to other nodes nor received from other nodes. A Pure Local Time Base will only locally be set and read. It is therefore possible to have multiple Pure Local Time Bases with the same Time Domain number in various nodes in parallel. A Pure Local Time Base behaves like a Synchronized Time Base since it progresses in time, however it is not synchronized via Timesync modules. Pure Local Time Bases behaving like an Offset Time Bases are not supported.

2.2.5 Time Base Provider

Definition: A Time Base Provider is the role that a <Bus>TSyn module takes for a given Time Base. Therefore a <Bus>TSyn module can contain only one Time Base provider or more than one Time Base provider. Time Base providers are either of type importer or exporter, whereas an importer acts as Time Slave and an exporter acts as Time Master. A Time Gateway consists of one Time Base importer and one or more Time Base exporters for a given Time Base. In order to limit the terminology importers are denoted as slaves and exporters are denoted as masters.

2.2.6 Time Communication Port

Definition: A Time Communication Port is a physical communication interface (in AUTOSAR coverable by the item: Physical Connector) at an ECU which is used to transport time information.

2.2.7 Time Communication Service

Definition: A Time Communication Service is an interaction between Time Bases which is performed by Time Base providers. Time communication services are message based between a Time Master and one or more Time Slaves or between one Time Slave and his Time Master.

Figure 3 shows a network topology example and the related terminology.

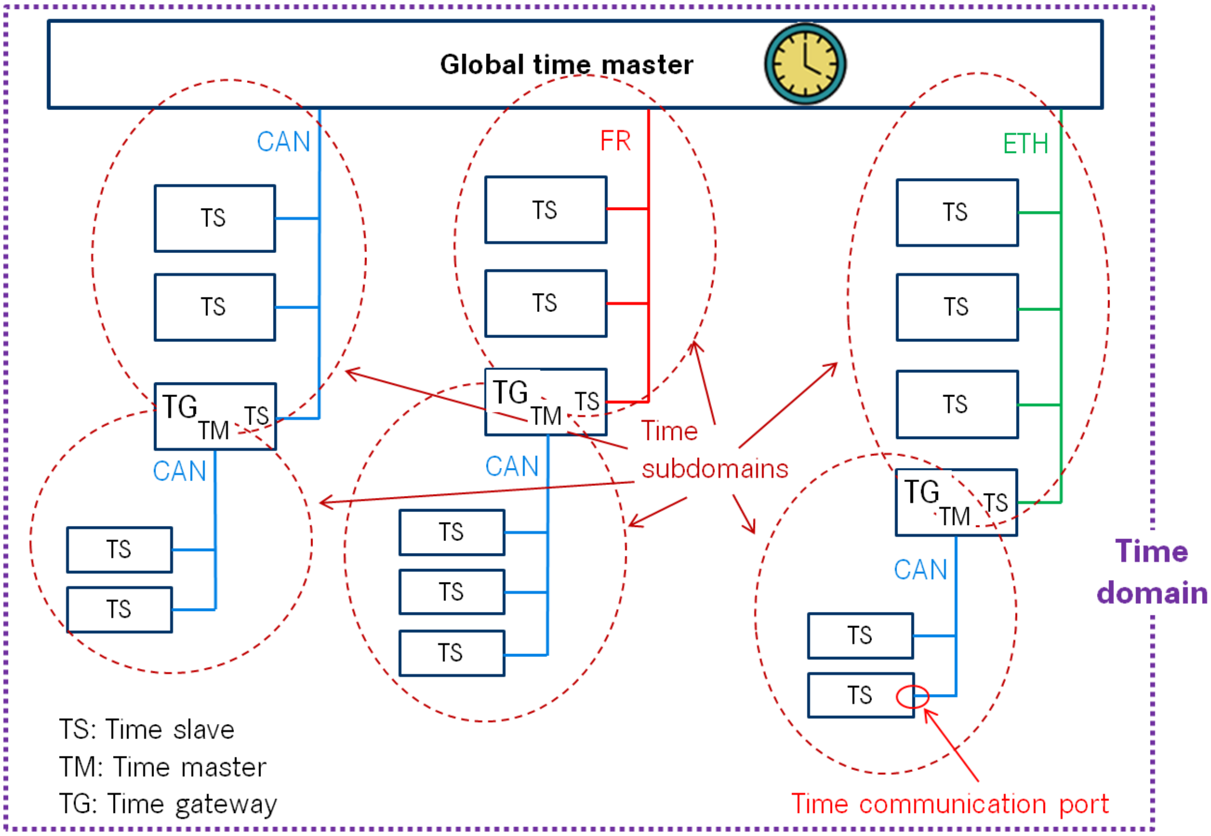


Figure 3: Terminology Example

2.2.8 Time Base Customer

a) Active Customer

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

- To read time information (arrow “2” in Figure 1) from the Synchronized Time-Base Manager or
- To update (arrow “3” in Figure 1) the Time Base maintained by the Synchronized Time-Base Manager according to application information.

b) Triggered Customer

This kind of customer is triggered by the Synchronized Time-Base Manager (arrow “1” in Figure 1). 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.

c) Notification Customer

This kind of customer is notified by the Synchronized Time-Base Manager (arrow “4” in Figure 1), 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 (arrow “5” in Figure 1).

2.2.9 Time Domain

Definition: A Time Domain denotes which components (e.g. nodes, communication systems) are linked to a certain Time Base. A Time Domain can contain no or more than one Time Sub-domains. If the timing hierarchy of a Time Domain contains no Time Gateways, i.e. all nodes are connected to the same bus system, then there is no dedicated Time Sub-domain which otherwise would be equal to the Time Domain itself.

2.2.10 Time Gateway

Definition: A Time Gateway is a set of entities where one entity is acting as Time Slave for a certain Time Base. The other (one or more) entities are acting as Time Masters which are distributing this Time Base to sets of Time Slaves. A Timesync ECU can contain multiple Time Gateways. A Time Gateway can be connected to different types of bus systems (e.g. the slave side could be connected to a FlexRay bus whereas the master side could be connected to a CAN bus system).

2.2.11 Time Hierarchy

Definition: The Time Hierarchy describes how a certain Time Base is distributed, starting at the Global Time Master and being distributed across various Time Gateways (if present) to various Time Slaves.

2.2.12 Time Master

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

Special Case “Pure Local Time Master“:
A Pure Local Time Master is an entity which is the master of a Pure Local Time Base and which does therefore not propagate this time base to any Time Slave.

2.2.13 Time Slave

Definition: A Time Slave is an entity which is the recipient for a certain Time Base within a certain segment of a communication network, being a consumer for this Time Base.

2.2.14 Time Sub-domain

Definition: A Time Sub-domain denotes which components (e.g. nodes) are linked to a certain Time Base whereas the scope is limited to one communication bus.

2.2.15 Timesync ECU

Definition: A Timesync ECU is an ECU which is part of a Time Domain by containing one or more Time Slaves or Time Masters.

2.2.16 Timesync Module

Definition: Timesync Modules (<Bus>TSyn modules) are bus specific modules to receive or transmit time information on bus systems by applying bus specific mechanisms. A Timesync module can serve multiple communication buses of the same type.

2.2.17 Virtual Local Time

Definition: Virtual Local Time is a time which is driven by a hardware reference clock and which in turn drives a Synchronized Time Base. The associated Synchronized Time Base has an offset to the Virtual Local Time. Furthermore there is usually also a deviation in rate caused by the limited precision of the hardware reference clock.

The term Virtual Local Time describes a Time Base which does not overflow and whose time progresses monotonously without jumps. In the scope of this document, it is an abstract construct used to describe functionalities (e.g. time spans) of the StbM.

Virtual Local Times could be actually implemented to simplify the realization of StbM functionalities. A typical approach would be to use a hardware timer as real-time source and count its overflows with a software counter. Hence the counter-width can be extended virtually indefinite. Depending on the hardware timer’s tick-duration, an additional conversion of its counter value to real-time has to be performed.

2.2.18 Time Correction

Definition: Time Correction in Time Slaves is the process of adjusting the value of the local instance of the Time Base to the value of the Global Time Base.
In Time Masters, Time Correction is the process of eliminating the deviation of an Offset Clock compared to its corresponding Synchronized Time Base.
Time Correction can be divided into Rate Correction, which corrects rate deviations and Offset Correction, which corrects absolute time deviations. Offset Correction can furthermore be divided into Jump Correction or Rate Adaption.

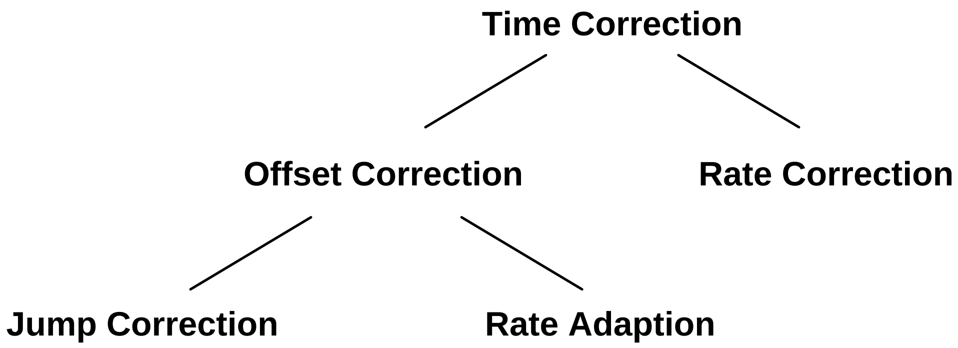


Figure 4 Time Correction Hierarchy

Note:

- Rate Deviation: This means that the time progresses at different rates in the local instance of the Time Base and the global Time Base. Such deviations can occur if, for example, the local hardware reference clock is driven by a crystal whose frequency is off due to manufacturing tolerances and/or thermal effects.
- Time Offset: This means that the local instance of the Time Base and the global Time Base are not synchronized precisely. Such deviations occur when the rate of the local hardware reference clock is not accurate and because the synchronization with the global Time Base is influenced by jitter effects, software delays and counter granularities.

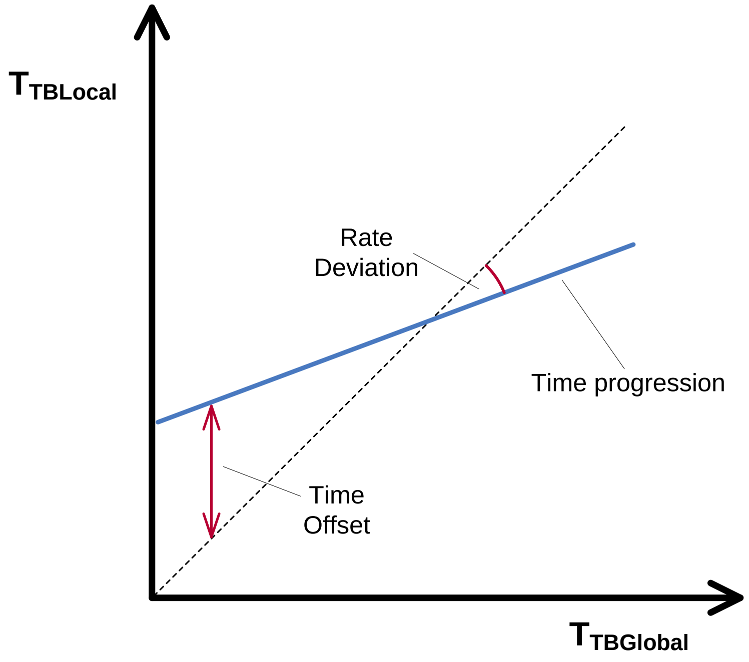


Figure 5: Time Deviations Rate Correction

Definition: Rate Correction corrects the rate-deviation of a local hardware reference clock. This correction is done by a multiplicative correction factor which is used in addition to the clock‘s preconfigured rate. Rate Correction determines the correction factor in the scope of a measurement. This correction factor is however not fixed but updated after each successful measurement.

The working principle of Rate Correction is not to adjust the local hardware reference clock in order to let it progress with the correct rate. Instead Rate Correction only corrects the values of the local instance of the Time Base on-the-fly when they are read.

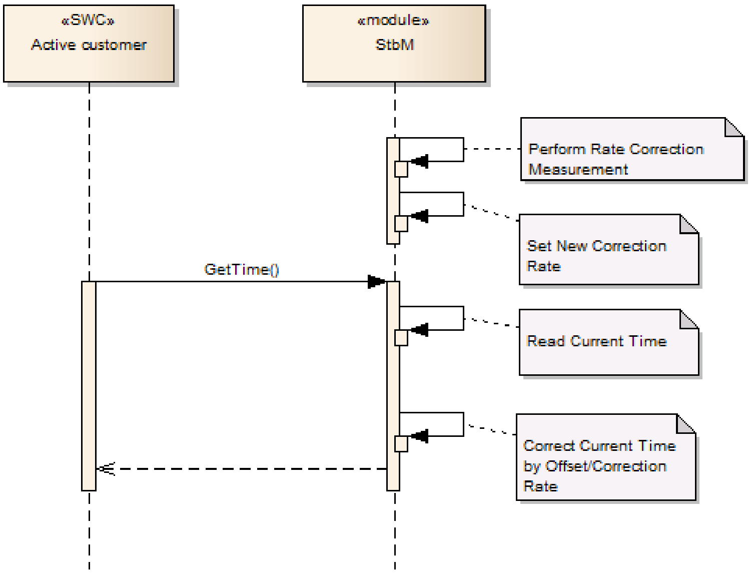

«SWC»
Active customer
«module»
StbM
Perform Rate Correction Measurement
Set New Correction Rate
GetTime()
Read Current Time
Correct Current Time by Offset/Correction Rate

Figure 6: Rate Correction

2.2.19 Offset Correction

Definition: Offset Correction corrects absolute time deviations (offsets). Depending on the magnitude of the offset and the configuration of StbM, this correction is either performed by Jump Correction or Rate Adaption.
Offset Correction is independent from Rate Correction. It is performed each time the local instance of the Time Base is synchronized to its Global Time Base.

2.2.20 Jump Correction

Definition: Jump Correction corrects absolute time offsets in a single step by adding the offset to the local instance of the Time Base (which is equivalent to taking over the value of the Global Time Base).

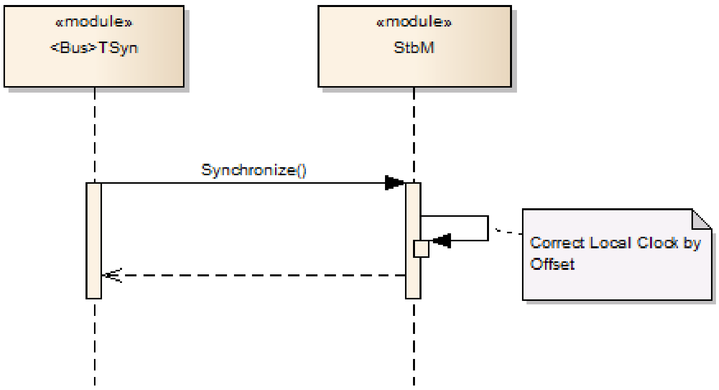


Figure 7: Offset Jump Correction

2.2.21 Rate Adaption

Definition: Rate Adaption corrects time offsets gradually within a predefined timespan. Hereto, Rate Adaption switches the rate of the local instance of the Time Base temporarily to a different value. This rate is chosen to completely eliminate the offset within the preconfigured timespan.

Like Rate Correction, Rate Adaption does not adjust the local instance of the Time Base (including hardware reference clock). It merely corrects the clock values on-the-fly when they are read.

Note: Rate Adaption and Rate Correction use a similar mechanism, they are however completely independent from each other.

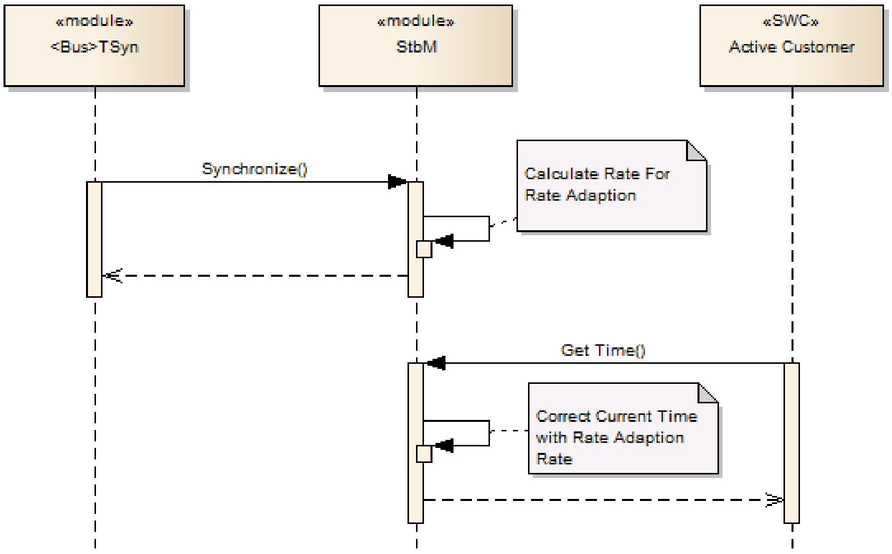

«module»
<Bus>TSyn
«module»
StbM
«SWC»
Active Customer
Synchronize()
Calculate Rate For Rate Adaption
Get Time()
Correct Current Time with Rate Adaption Rate

Figure 8: Offset Rate Adaption

3 Related documentation

3.1 Input documents

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

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[3] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[4] Specification of Operating System
AUTOSAR_SWS_OS.pdf

[5] Specification of FlexRay Interface
AUTOSAR_SWS_FlexRayInterface.pdf

[6] Specification of CAN Interface
AUTOSAR_SWS_CANInterface.pdf

[7] Virtual Functional Bus
AUTOSAR_EXP_VFB.pdf

[8] Software Component Template
AUTOSAR_TPS_SoftwareComponentTemplate.pdf

[9] Basic Software Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[10] Specification of TimingExtensions
AUTOSAR_TPS_TimingExtensions.pdf

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

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

[15] Specification of RTE
AUTOSAR_SWS_RTE.pdf

[16] Specification of Synchronized Time-Base Manager
AUTOSAR_EXP_CDDDesignAndIntegrationGuideline.pdf

3.2 Related standards and norms

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

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [14] (SWS BSW General), which is also valid for the Synchronized Time-Base Manager.

Thus, the specification SWS BSW General shall be considered as additional and required specification for the Synchronized Time-Base Manager.

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 **ECUC_StbM_00036 :**).

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 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 SWS BSW General [14]

5.2 Header file structure

For details, refer to the section 5.1.7 " Header file structure" of the SWS BSW General [14].

In addition to the files defined in section 5.1.7 “Header file structure” of the SWS BSW General, the StbM needs to include the file Os.h, EthIf.h and Gpt.h.

[SWS_StbM_00065]⌈
If a triggered customer is configured (refer to **ECUC_StbM_00004 :** 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 **ECUC_StbM_00053 : ,** if set to EthTSynGlobalTimeDomain), StbM.c shall include EthIf.h to have access to the interface of the EthIf module.
⌋ (SRS_BSW_00384)

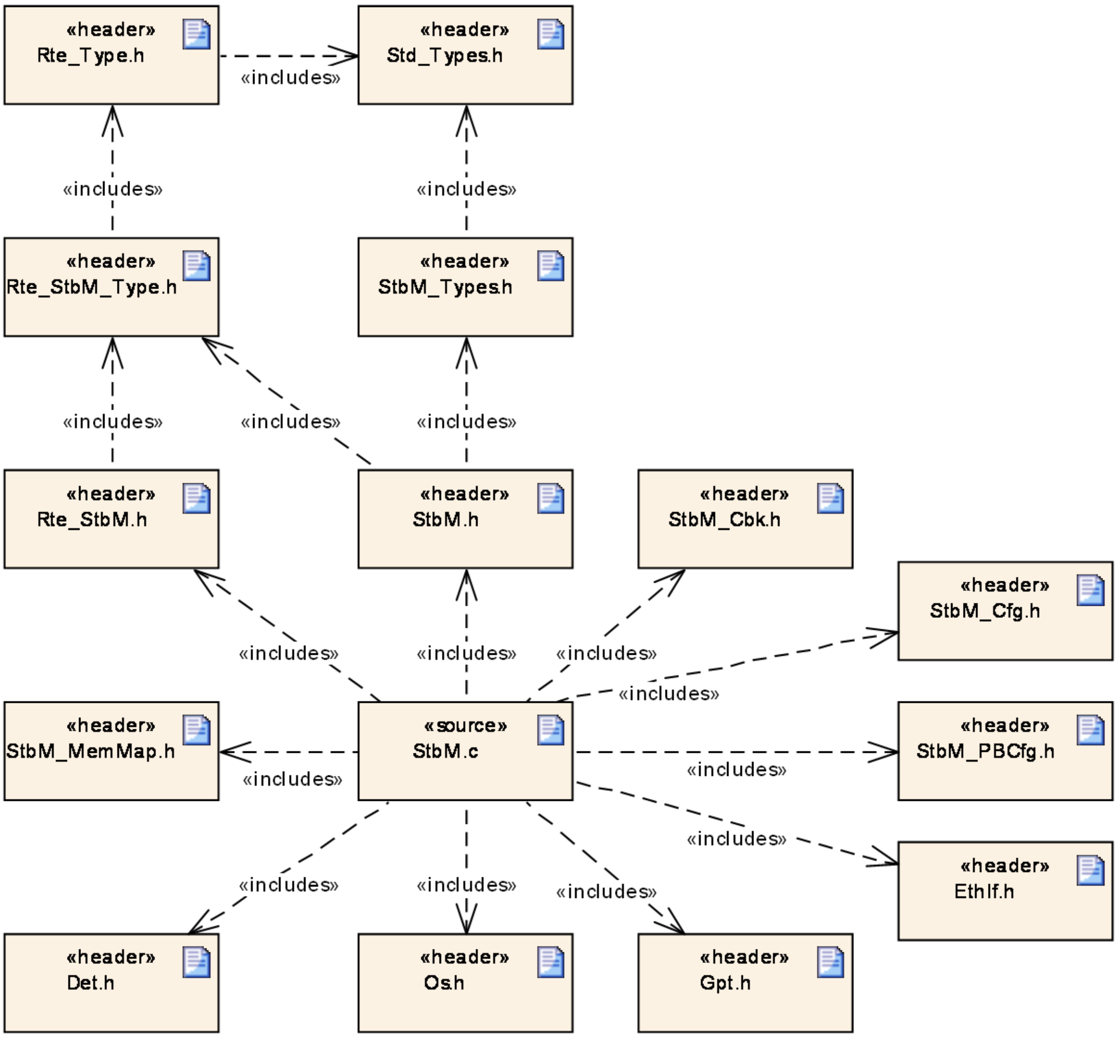


Figure 9: Header File Structure

6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00005	Modules of the µC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_StbM_00140
SRS_BSW_00006	The source code of software modules above the µC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_StbM_00140
SRS_BSW_00007	All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.	SWS_StbM_00140
SRS_BSW_00009	All Basic SW Modules shall be documented according to a common standard.	SWS_StbM_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_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_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_00140
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_StbM_00140
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_StbM_00140
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_StbM_00140
SRS_BSW_00170	The AUTOSAR SW	SWS_StbM_00140

	Components shall provide information about their dependency from faults, signal qualities, driver demands	
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 the following data types instead of native C data types	SWS_StbM_00140
SRS_BSW_00305	Data types naming convention	SWS_StbM_00142
SRS_BSW_00307	Global variables naming convention	SWS_StbM_00140
SRS_BSW_00308	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	SWS_StbM_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_00140
SRS_BSW_00312	Shared code shall be reentrant	SWS_StbM_00140
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_StbM_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_00201, SWS_StbM_00202, SWS_StbM_00206, SWS_StbM_00210, 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_00402, SWS_StbM_00403, SWS_StbM_00404, SWS_StbM_00405, SWS_StbM_00406, SWS_StbM_00415, SWS_StbM_00416, SWS_StbM_00417, SWS_StbM_00418
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_StbM_00140
SRS_BSW_00327	Error values naming convention	SWS_StbM_00041, SWS_StbM_00198
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_StbM_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_00140
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_StbM_00140
SRS_BSW_00337	Classification of development errors	SWS_StbM_00041, SWS_StbM_00094, SWS_StbM_00198
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_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_00140
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_StbM_00140
SRS_BSW_00347	A Naming seperation of different instances of BSW drivers shall be in place	SWS_StbM_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_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	SWS_StbM_00273, SWS_StbM_00285

	are allowed to have parameters	
SRS_BSW_00361	All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header	SWS_StbM_00140
SRS_BSW_00371	The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules	SWS_StbM_00140
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_00140
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_StbM_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
SRS_BSW_00385	List possible error notifications	SWS_StbM_00041
SRS_BSW_00386	The BSW shall specify the configuration for detecting an error	SWS_StbM_00041, SWS_StbM_00094, SWS_StbM_00196, SWS_StbM_00197, SWS_StbM_00198, SWS_StbM_00201, SWS_StbM_00202, SWS_StbM_00206, SWS_StbM_00210, 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_00402, SWS_StbM_00403, SWS_StbM_00404, SWS_StbM_00405, SWS_StbM_00406, SWS_StbM_00415, SWS_StbM_00416, SWS_StbM_00417, SWS_StbM_00418
SRS_BSW_00398	The link-time configuration is achieved on object code basis in the stage after compiling and before linking	SWS_StbM_00140

SRS_BSW_00399	Parameter-sets shall be located in a separate segment and shall be loaded after the code	SWS_StbM_00140
SRS_BSW_00400	Parameter shall be selected from multiple sets of parameters after code has been loaded and started	SWS_StbM_00140
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_StbM_00140
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_StbM_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_00412	References to c-configuration parameters shall be placed into a separate h-file	SWS_StbM_00140
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_StbM_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_00140
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_StbM_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_00140
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_StbM_00140
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_StbM_00140
SRS_BSW_00427	ISR functions shall be	SWS_StbM_00140

	defined and documented in the BSW module description template	
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_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_00140
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_StbM_00140
SRS_BSW_00437	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	SWS_StbM_00140
SRS_BSW_00438	Configuration data shall be defined in a structure	SWS_StbM_00140
SRS_BSW_00439	Enable BSW modules to handle interrupts	SWS_StbM_00140
SRS_BSW_00440	The callback function invocation by the BSW module shall follow the signature provided by RTE to invoke servers via Rte_Call API	SWS_StbM_00140
SRS_BSW_00453	BSW Modules shall be harmonized	SWS_StbM_00140
SRS_BSW_00457	- Callback functions of Application software components shall be invoked by the Basis SW	SWS_StbM_00273, SWS_StbM_00285
SRS_StbM_20001	The StbM configuration shall allow the interaction with different types of customers	SWS_StbM_00020, SWS_StbM_00022, SWS_StbM_00093, SWS_StbM_00277, SWS_StbM_00278, SWS_StbM_00279, SWS_StbM_00282, SWS_StbM_00285, SWS_StbM_00303
SRS_StbM_20002	The StbM 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
SRS_StbM_20003	The StbM 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
SRS_StbM_20007	The StbM shall provide fault detection mechanisms	SWS_StbM_00031, SWS_StbM_00183, SWS_StbM_00187, SWS_StbM_00199, SWS_StbM_00419, SWS_StbM_00420
SRS_StbM_20010	The StbM shall provide a system service interface to applications	SWS_StbM_00142, SWS_StbM_00240, SWS_StbM_00244, SWS_StbM_00247, SWS_StbM_00248, SWS_StbM_00275, SWS_StbM_00276, SWS_StbM_00286, SWS_StbM_00287, SWS_StbM_00288, SWS_StbM_00290
SRS_StbM_20012	The StbM shall provide a bus independent customer interface	SWS_StbM_00241, SWS_StbM_00242
SRS_StbM_20013	The StbM shall provide time information for Timesync modules	SWS_StbM_00173, SWS_StbM_00174, SWS_StbM_00175, SWS_StbM_00195, SWS_StbM_00205, SWS_StbM_00209
SRS_StbM_20014	The StbM shall synchronize on Time Slave side its Time Base on reception of a Time Master value	SWS_StbM_00179, SWS_StbM_00233, SWS_StbM_00393
SRS_StbM_20016	The StbM shall continuously maintain its Time Bases based on a Time Base reference clock	SWS_StbM_00174, SWS_StbM_00175, SWS_StbM_00178, SWS_StbM_00180, SWS_StbM_00205, SWS_StbM_00209, SWS_StbM_00413
SRS_StbM_20018	The StbM shall initialize the Local Time Base with 0 at startup if configured as Time Slave	SWS_StbM_00170
SRS_StbM_20019	The StbM shall initialize the Global Time Base with a configurable startup value if configured as Time Master	SWS_StbM_00171
SRS_StbM_20020	The StbM shall support storage of the Time Base value at shutdown if configured as Time Master	SWS_StbM_00172
SRS_StbM_20021	The StbM shall use a time format with a resolution of 1 ns	SWS_StbM_00174, SWS_StbM_00175
SRS_StbM_20023	The StbM configuration shall allow the StbM to support different roles for a Time Base	SWS_StbM_00195, SWS_StbM_00213, SWS_StbM_00223, SWS_StbM_00233, SWS_StbM_00244, SWS_StbM_00408, SWS_StbM_91001, SWS_StbM_91002
SRS_StbM_20024	The StbM shall always maintain the Time Base	SWS_StbM_00178, SWS_StbM_00180, SWS_StbM_00342, SWS_StbM_00413
SRS_StbM_20025	The StbM 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_00194, SWS_StbM_00239, SWS_StbM_00305, SWS_StbM_00393, SWS_StbM_00399, SWS_StbM_00419, SWS_StbM_00420, SWS_StbM_00425

SRS_StbM_20026	The StbM shall allow customer on master side to set the global time	SWS_StbM_00213, SWS_StbM_00240, SWS_StbM_00244, SWS_StbM_00342, SWS_StbM_00385
SRS_StbM_20027	The StbM shall allow Timesync modules to read the offset value of an Offset Time Base	SWS_StbM_00191, SWS_StbM_00193, SWS_StbM_00228
SRS_StbM_20028	The StbM shall allow customers and Timesync modules to set the offset value of an Offset Time Base	SWS_StbM_00177, SWS_StbM_00190, SWS_StbM_00191, SWS_StbM_00192, SWS_StbM_00193, SWS_StbM_00223, SWS_StbM_00240, SWS_StbM_00244, SWS_StbM_00304
SRS_StbM_20029	The StbM shall allow customers to read User Data propagated via the Time Synchronization protocol	SWS_StbM_00173, SWS_StbM_00192, SWS_StbM_00195, SWS_StbM_00200, SWS_StbM_00243, SWS_StbM_00247, SWS_StbM_00248
SRS_StbM_20030	The StbM shall allow customers to set User Data propagated via the Time Synchronization protocol	SWS_StbM_00190, SWS_StbM_00218, SWS_StbM_00240, SWS_StbM_00243, SWS_StbM_00244, SWS_StbM_00381, SWS_StbM_00398
SRS_StbM_20054	The StbM 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
SRS_StbM_20056	The StbM shall notify customers about a set time	SWS_StbM_00247, SWS_StbM_00257, 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_00300, SWS_StbM_00301, SWS_StbM_00335, SWS_StbM_00336, SWS_StbM_00337, SWS_StbM_00409, SWS_StbM_00421, SWS_StbM_91004
SRS_StbM_20057	The StbM 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
SRS_StbM_20064	The StbM shall allow customers on master side to trigger time transmission by	SWS_StbM_00240, SWS_StbM_00344, SWS_StbM_00346, SWS_StbM_00347, SWS_StbM_00350, SWS_StbM_00351,

	the Time Providers	SWS_StbM_00414
SRS_StbM_20065	The StbM shall support rate correction	SWS_StbM_00352, SWS_StbM_00353, SWS_StbM_00354, SWS_StbM_00355, SWS_StbM_00356, SWS_StbM_00359, SWS_StbM_00360, SWS_StbM_00361, SWS_StbM_00362, SWS_StbM_00363, SWS_StbM_00364, SWS_StbM_00365, SWS_StbM_00366, SWS_StbM_00367, SWS_StbM_00368, SWS_StbM_00369, 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_00423, SWS_StbM_00424
SRS_StbM_20067	The StbM shall support smooth offset correction	SWS_StbM_00354, SWS_StbM_00356

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.

7.1.2 Initialization

[SWS_StbM_00170]⌈
On invocation of `StbM_Init()` each configured Time Base (refer to `StbMSynchronizedTimeBase`, **ECUC_StbM_00003 :**) shall be initialized with zero and its synchronization status `timeBaseStatus` shall be set to `0x00`.
⌋ (SRS_StbM_20018)

[SWS_StbM_00345]⌈
For each Time Base the StbM shall initialize the corresponding event status `NotificationEvents` with 0.
⌋ (SRS_StbM_20054)

[SWS_StbM_00344]⌈
For each Time Base the StbM shall initialize the corresponding update counter `timeBaseUpdateCounter` with 0.
⌋ (SRS_StbM_20064)

[SWS_StbM_00171]⌈
For each Time Base configured to be stored non-volatile (`StbMStoreTimebaseNonVolatile == STORAGE_AT_SHUTDOWN`), the Time Base value shall be loaded from NvM. In case the restore is not successful, the Time Base shall start with zero.
⌋ (SRS_StbM_20019)

Note: The further details on the NvM handling is 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 via callback or via constant.

[SWS_StbM_00306]⌈
If `StbMTimeRecordingSupport` (**ECUC_StbM_00038 :**) is set to `TRUE`, the StbM shall initialize all Block Elements of the measurement recording table with zero.

⌋ (SRS_StbM_20057)

7.2 Shutdown behavior

[SWS_StbM_00172]⌈
For each Time Base configured to be stored non-volatile (`StbMStoreTimebaseNonVolatile == STORAGE_AT_SHUTDOWN`), the value shall be stored to NvM latest at shutdown.
⌋ (SRS_StbM_20020)

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

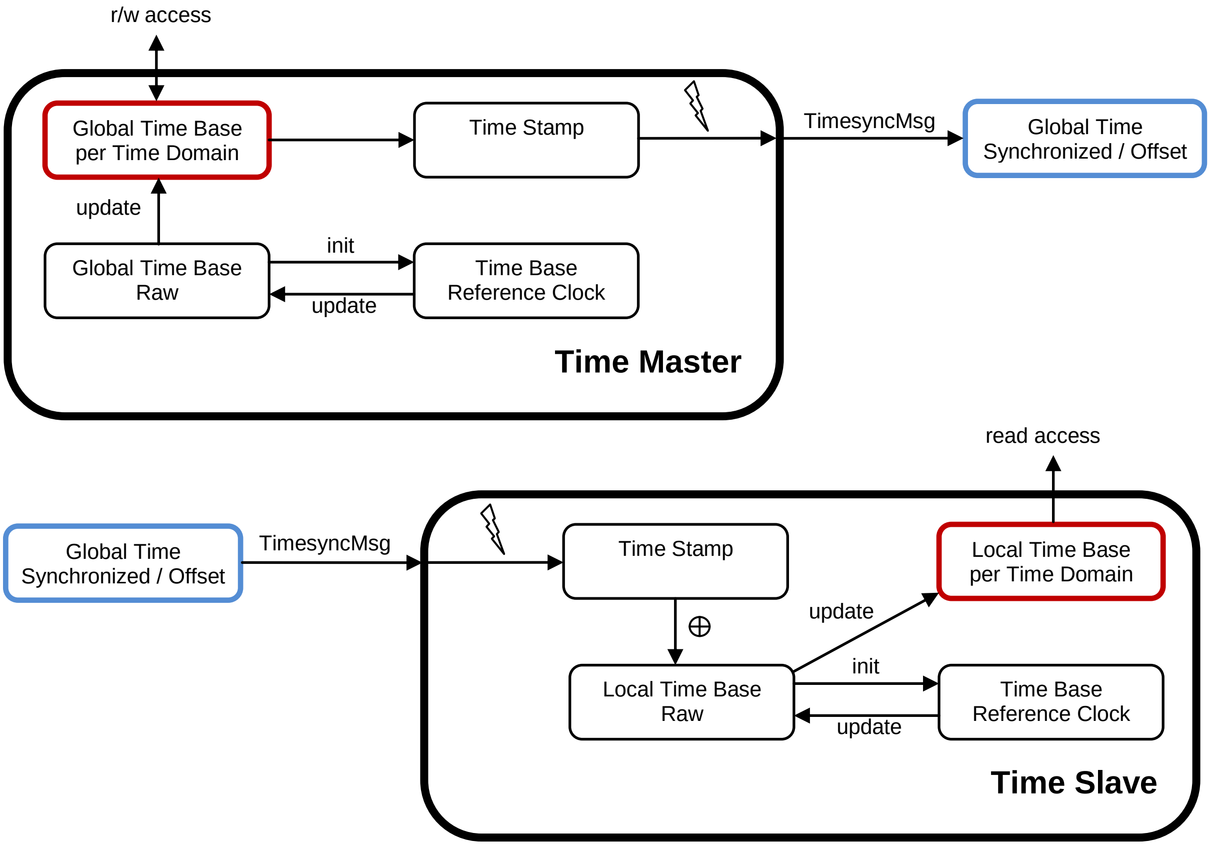

r/w access
Global Time Base per Time Domain
Time Stamp
TimesyncMsg
Global Time Synchronized / Offset
update
Global Time Base Raw
init
update
Time Base Reference Clock
Time Master
read access
Global Time Synchronized / Offset
TimesyncMsg
Time Stamp
Local Time Base per Time Domain
update
⊕
init
Local Time Base Raw
update
Time Base Reference Clock
Time Slave

Figure 10: Global Time Base Distribution

7.3.1.1 Types of Time Bases

7.3.1.1.1 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.

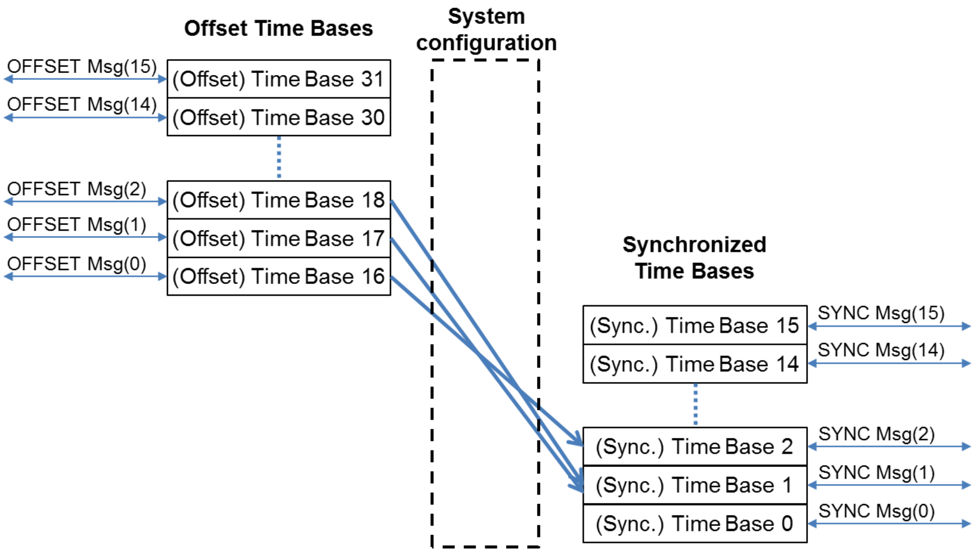


Figure 11: Offset Time Base to Synchronized Time Base relationship

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 (Note that the OFS Sub-TLVs within the AUTOSAR TLV on Ethernet always contain 17 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), is also based on the underlying Synchronized Time Base 0. An Offset Time Base might have leaps in time, e.g. after GPS time becomes available.

7.3.1.1.2 Pure Local Time Bases

For details of the Pure Local Time Bases refer to 7.3.4.

7.3.1.2 Roles of the StbM

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

- Global Time Master
- Time Slave
- Time Gateway

In each role specific functionality is supported or not supported.

7.3.1.2.1 Global Time Master

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

[SWS_StbM_00408]⌈

StbM_GetMasterConfig() shall return the value of the configuration parameter StbMIsSystemWideGlobalTimeMaster (**ECUC_StbM_00036 :**) for the Time Base timeBaseId. This is to check, if the StbM is configured as system wide Global Time Master for a specific Time Base.
⌋ (SRS_StbM_20023)

7.3.1.2.2 Time Slave

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

7.3.1.2.3 Time Gateway

A Time Gateway in the StbM is a Time Base which is referenced by one Time Slave and one or more Time Masters. The Time Slave, which references a StbM Time Gateway receives Timesync messages on the corresponding bus and passes the received Time Base values to the StbM (refer to 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 can not automatically trigger the transmission on the master side.

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

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).
⌋ (SRS_StbM_20016, SRS_StbM_20024)

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).

[SWS_StbM_00178]⌈
If EthIfGlobalTimeSupport (referenced via StbMLocalTimeHardware **ECUC_StbM_00053 : ,** if set to EthTSynGlobalTimeDomain) is set to TRUE for a Synchronized Time Base, the StbM shall retrieve the Local Time from the corresponding Ethernet Controller via EthIf_GetCurrentTime().
⌋ (SRS_StbM_20016, SRS_StbM_20024)

[SWS_StbM_00173]⌈

For Time Domains 0 to 15 `StbM_GetCurrentTime()` and `StbM_GetCurrentTimeExtended()` shall return for the requested Time Domain the time of the Time Base, the related Status and the User Data. The current time of the Time Base shall be derived from the related Virtual Local Time, which is derived from either the referenced OS counter, a GPT or a referenced Ethernet controller (refer to `StbMLocalTimeHardware`).
⌋ (SRS_StbM_20003, SRS_StbM_20013, SRS_StbM_20029)

Note: Calling `StbM_GetCurrentTime()` shall not worsen the precision of the requested Time Base.

[SWS_StbM_00352]⌈
In the scope of `StbM_GetCurrentTime()` and `StbM_GetCurrentTimeExtended()`, StbM shall use the factor (`StbMClockPrescaler /StbMClockFrequency`) to convert the time of its local hardware reference clock to the actual time of the Virtual Local Time.
⌋ (SRS_StbM_20065)

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_00174]⌈
`StbM_GetCurrentTimeRaw()` shall return the nanoseconds part of the Virtual Local Time of the associated Time Base (refer **[SWS_StbM_00173]**).
⌋ (SRS_StbM_20013,SRS_StbM_20016, SRS_StbM_20021)

[SWS_StbM_00175]⌈
`StbM_GetCurrentTimeDiff()` shall return the time difference of the nanoseconds part of the Virtual Local Time of the associated Time Base (refer to **[SWS_StbM_00173]**) minus the time given by the paramerter `givenTimeStamp` in raw format.
⌋ (SRS_StbM_20016, SRS_StbM_20021, SRS_StbM_20013)

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 Local Time of the corresponding Time Base.
⌋ (SRS_StbM_20026, SRS_StbM_20024)

7.3.2.2 Time Slave

[SWS_StbM_00179]⌈

For Time Domains 0 to 15 each invocation of `StbM_BusSetGlobalTime()` shall update the corresponding Synchronized Time Base and set the User Data and the Time Base Status accordingly.
⌋ (SRS_StbM_20014, SRS_StbM_20025)

7.3.3 Offset Time Bases

[SWS_StbM_00191]⌈
`StbM_SetOffset()` and `StbM_GetOffset()` shall only accept Offset Time Bases with a timeBaseId 16 to 31.
⌋ (SRS_StbM_20027, SRS_StbM_20028)
[SWS_StbM_00177]⌈
For Time Domains 16 to 31 the `StbM_GetCurrentTime()` and `StbM_GetCurrentTimeExtended()` shall return for the requested Time Domain an absolute time value calculated by adding the given offset to the current Time Base of the referenced Time Domain via `StbMOffsetTimeBase` (**ECUC_StbM_00030 :**).
⌋ (SRS_StbM_20028)

[SWS_StbM_00193]⌈
Configuration Constraint: The parameter `StbMOffsetTimeBase` shall only be valid for `StbMSynchronizedTimeBaseIdentifier` 16 to 31.
⌋ (SRS_StbM_20027, SRS_StbM_20028)

7.3.3.1 Global Time Master

[SWS_StbM_00190]⌈
Each invocation of `StbM_SetOffset()` shall update the Offset Time and the User Data of the corresponding Time Base.
⌋ (SRS_StbM_20028, SRS_StbM_20030)

[SWS_StbM_00192]⌈
Each invocation of `StbM_GetOffset()` shall return the Offset Time and the User Data of the corresponding Offset Time Base.
⌋ (SRS_StbM_20028, SRS_StbM_20029)

[SWS_StbM_00304]⌈
On invocation of `StbM_SetGlobalTime()` or `StbM_UpdateGlobalTime()` for Time Domains 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 is not set.

If the `GLOBAL_TIME_BASE` bit is set, 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()

- 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().

⌋ (SRS_StbM_20028)

7.3.3.2 Time Slave

[SWS_StbM_00393]⌈
For Time Domains 16 to 31 each invocation of StbM_BusSetGlobalTime() shall update the corresponding Offset Time Base and set the User Data and the Time Base Status accordingly.
⌋ (SRS_StbM_20014, SRS_StbM_20025)

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 StbMLocalTimeClock).
⌋ (SRS_StbM_20016, SRS_StbM_20024)

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() and StbM_GetCurrentTimeExtended() shall return the User Data as set by StbM_SetGlobalTime(), StbM_UpdateGlobalTime() or StbM_SetUserData() by the local Pure Local Time Master.
⌋ (SRS_StbM_20030)

[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().
⌋ (SRS_StbM_20025)

7.3.5 Synchronization State

[SWS_StbM_00261]⌈
For Offset Time Bases StbM_GetCurrentTime() and StbM_GetCurrentTimeExtended() 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 (**ECUC_StbM_00030 :**):

Bit Name	Bit Position	Description
TIMEOUT	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	Bit 1: Always 0 (reserved for future usage)
SYNC_TO_GATEWAY	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
GLOBAL_TIME_BASE	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
TIMELEAP_FUTURE	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
TIMELEAP_PAST	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

⌋ (SRS_StbM_20003)

[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 ECUC_StbM_00030 :) via `syncTimeBaseStatus`.

⌋ (SRS_StbM_20003)

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.
⌋ (SRS_StbM_20025)

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 each Time Domain where a Time Slave or a Time Gateway Slave Port belongs to, an invocation of `StbM_BusSetGlobalTime()` shall check, if the time difference between the updated and the current Time Base value exceeds the configured threshold of `StbMTimeLeapFutureThreshold` (**ECUC_StbM_00041 :**), i.e. TG - TL_{Sync} > `StbMTimeLeapFutureThreshold`, if at least one Time Base value has been successfully received before.

With:

- TL_{Sync} = Value of the local instance of the Time Base before the new value of the Global Time is applied
- TG = Received value of the Global Time

In case the threshold is exceeded the StbM shall set the `TIMELEAP_FUTURE` bit within `timeBaseStatus` of the Time Base.

If the next StbMClearTimeleapCount updates are within the threshold of StbMTimeLeapFutureThreshold the StbM shall clear the TIMELEAP_FUTURE bit within timeBaseStatus of the Time Base.

A threshold of 0 shall deactivate this check.
⌋ (SRS_StbM_20025)

[SWS_StbM_00305]⌈
For each Time Domain where a Time Slave or a Time Gateway Slave Port belongs to, an invocation of StbM_BusSetGlobalTime() shall check, if the time difference between the current and the updated Time Base value exceeds the configured threshold of StbMTimeLeapPastThreshold (**ECUC_StbM_00042 :**), i.e. TL_{Sync} - TG > StbMTimeLeapPastThreshold, if at least one Time Base value has been successfully received before.

With:

- TL_{Sync} = Value of the local instance of the Time Base before the new value of the Global Time is applied
- TG = Received value of the Global Time

In case the threshold is exceeded the StbM shall set the TIMELEAP_PAST bit within timeBaseStatus of the Time Base.

If the next StbMClearTimeleapCount updates are within the threshold of StbMTimeLeapPastThreshold the StbM shall clear the TIMELEAP_PAST bit within timeBaseStatus of the Time Base.

A threshold of 0 shall deactivate this check.
⌋ (SRS_StbM_20025)

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/StbMTimeLeapPastThreshold have not yet happened.

[SWS_StbM_00425]⌈
For Time Slaves and Time Gateways of Synchronized Time Bases StbM_GetTimeLeap() shall return the time difference between the newly received and the current Time Base value, i.e. TG - TL, which is calculated upon each, except

the very first, valid invocation of StbM_BusSetGlobalTime() for the corresponding Time Base.

With

- TL = Current value of the local instance of the Time Base (before newly received Time Base value is applied)
- TG = Newly received Time Base value

For Time Slaves and Time Gateways of Offset Time Bases StbM_GetTimeLeap() shall return the time difference between the newly received and the current Time Base offset value, i.e. TOG - TOL, which is calculated upon each, except the very first, valid invocation of StbM_BusSetGlobalTime() for the corresponding Time Base.

With

- TOL = Current offset value of the local instance of the Time Base (before newly received Time Base offset value is applied)
- TOG = Newly received Time Base offset value

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.
⌋ (SRS_StbM_20025)

[SWS_StbM_00183]⌈
For each Time Domain where a Time Slave belongs to, the StbM shall observe a timeout. The timeout StbMSyncLossTimeout (**ECUC_StbM_00028 :**) shall be measured based on the Virtual Local Time 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.
⌋ (SRS_StbM_20007, SRS_StbM_20025)

[SWS_StbM_00187]⌈
For each Time Domain where a Time Gateway Slave Port belongs to, the StbM shall observe a timeout. The timeout StbMSyncLossTimeout (**ECUC_StbM_00028 :**) shall be measured based on the Virtual Local Time 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.

If the timeout occurs, the StbM shall set the SYNC_TO_GATEWAY bit within timeBaseStatus of the Time Base.
⌋ (SRS_StbM_20007, SRS_StbM_20025)

[SWS_StbM_00420]⌈
The StbM shall check for a timeout condition of a Time Base within StbM_MainFunction() and all API functions, which return the Time Base Status (e.g. StbM_GetTimeBaseStatus() or StbM_GetCurrentTime())
⌋ (SRS_StbM_20007, SRS_StbM_20025)

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_00419]⌈
The StbM shall check for a timeout condition of a Time Base within StbM_MainFunction() and all API functions, which return the Time Base Status (e.g. StbM_GetTimeBaseStatus() or StbM_GetCurrentTime())
⌋ (SRS_StbM_20007, SRS_StbM_20025)

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_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 timeStampPtr argument passed to StbM_BusSetGlobalTime().
⌋ (SRS_StbM_20025)

[SWS_StbM_00185]⌈
For each Time Domain where a Time Slave or a Time Gateway Slave Port belongs to 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.
⌋ (SRS_StbM_20025)

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.
⌋ (SRS_StbM_20064)

[SWS_StbM_00351]⌈
For Synchronized and Offset Time Bases, the `timeBaseUpdateCounter` of a Time Base shall have the value range 0 to 255.
⌋ (SRS_StbM_20064)

[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.
⌋ (SRS_StbM_20064)

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.
⌋ (SRS_StbM_20030)

7.3.8 Time Correction

The Synchronized Time-Base Manager provides the ability for Time Slaves to perform Rate and Offset Correction of a Synchronized Time Base and Rate Correction of an Offset Time Base.

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 (for Time Slaves)

Rate Correction detects- and eliminates rate deviations of local instances of Time Bases and of Offset 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 read (E.g. in the scope of `StbM_GetCurrentTime()`).

[SWS_StbM_00377]⌈
The StbM shall not perform Rate Correction when the measurement duration `StbMRateCorrectionMeasurementDuration` (**ECUC_StbM_00054 :**) is set to zero.
⌋ (SRS_StbM_20065)

[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 the flags equals „Set".
⌋ (SRS_StbM_20065)

[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.
⌋ (SRS_StbM_20065)

[SWS_StbM_00374]⌈
For Rate Correction measurements, the StbM shall evaluate the TIMEOUT flag. The StbM shall discard the measurement, if the flag equals „Set“.
⌋ (SRS_StbM_20065)

[SWS_StbM_00373]⌈
For Rate Correction, the StbM shall evaluate the TIMELEAP_FUTURE/ TIMELEAP_PAST flags during the start of a measurement. The StbM shall not start a Rate Correction measurement when the state of any of the flags equals „Set“.
⌋ (SRS_StbM_20065)

[SWS_StbM_00372]⌈
The StbM shall perform Rate Correction measurements to determine the rate deviation of each configured Time Base.
⌋ (SRS_StbM_20065)

[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 is always triggered by and aligned to the reception of time values for Synchronized or Offset Time Bases.
⌋ (SRS_StbM_20065)

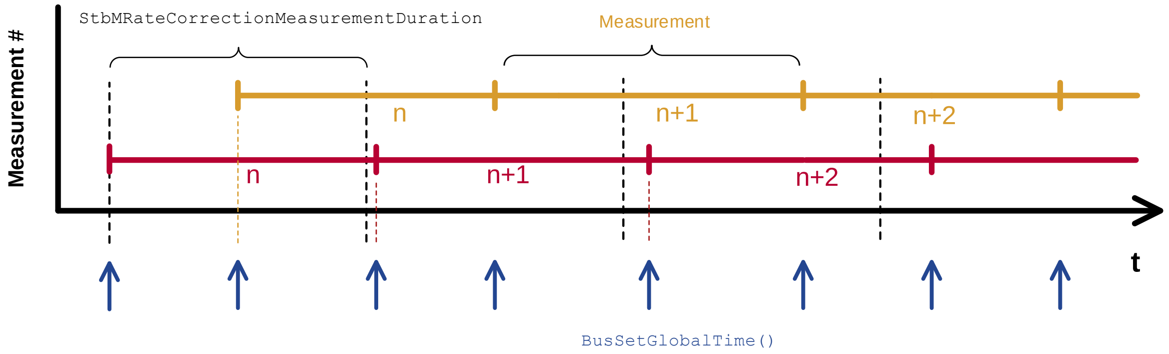


Figure 12: Visualization of two parallel measurements

[SWS_StbM_00370]⌈
During runtime, and for Synchronized Time Bases the StbM shall determine the timespan of a Rate Correction measurement on the basis of the Virtual Local Time.
⌋ (SRS_StbM_20065)

Note: Determination of the measurement duration by simply counting StbM_BusSetGlobalTime() calls (caused by incoming Timesync messages) and deriving the timespan which has passed from the cycle time may not lead to correct results since the Timesync cycle time is allowed to vary.

[SWS_StbM_00369]⌈
During runtime, and for Offset Time Bases the StbM shall determine the timespan of a Rate Correction measurement on basis of the Virtual Local Time.
⌋ (SRS_StbM_20065)

[SWS_StbM_00368]⌈
The StbM shall perform as many simultaneous Rate Correction measurements as configured by parameter: `StbMRateCorrectionsPerMeasurementDuration` (**ECUC_StbM_00055 :**).
⌋ (SRS_StbM_20065)

[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`) (where 'n' is the zero-based index of the current measurement)
⌋ (SRS_StbM_20065)

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 of Synchronized Time Bases, the StbM shall take the following time-snapshots in the scope of the function `StbM_BusSetGlobalTime()`:

- TG_{Start} – Current time of the global Time Base Time Master
- TV_{Start} – Current time of the Virtual Local Time of the associated Time Base

⌋ (SRS_StbM_20065)

[SWS_StbM_00365]⌈
At the start of a Rate correction measurement of Offset Time Bases, the StbM shall take the following time-snapshots in the scope of the function `StbM_BusSetGlobalTime()`:

- TV_{Start} – Value of the Virtual Local Time Base of the related Synchronized Time Base at the start of the measurement

- TO_{Start} – Current Offset value of the Offset Time Base given as function parameter

⌋ (SRS_StbM_20065)

[SWS_StbM_00364]⌈
At the end of the Rate Correction measurement of Synchronized Time Bases, the StbM shall take the following time-snapshots in the scope of the function `StbM_BusSetGlobalTime()`:

- TG_{Stop} – Current time of the Global Time Base Time Master
- TV_{Stop} – Current time of the Virtual Local Time of the associated Time Base

⌋ (SRS_StbM_20065)

[SWS_StbM_00363]⌈
At the end of the Rate Correction measurement of Offset Time Bases, the StbM shall take the following time-snapshots in the scope of the function `StbM_BusSetGlobalTime()`:

- TV_{Stop} – Current time of the Virtual Local Time of the related Synchronized Time Base
- TO_{Stop} – Current Offset value of the Offset Time Base given as function parameter

⌋ (SRS_StbM_20065)

[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:

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

⌋ (SRS_StbM_20065)

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} until a new value has been calculated.
⌋ (SRS_StbM_20065)

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

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

$$r_{orc} = (TO_{Stop} - TO_{Start})) / (TV_{Stop} - TV_{Start})$$

With:

- r_{orc} = Rate correction value of the Offset Time Base

- TV_{Stop} – Current time of the Virtual Local Time of the related Synchronized Time Base
- TO_{Stop} – Current Offset value of the Offset Time Base
- TV_{Start} – Value of the Virtual Local Time Base of the related Synchronized Time Base at the start of the measurement
- TO_{Start} – Offset value of the Offset Time Base at the start of the measurement

⌋ (SRS_StbM_20065)

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

[SWS_StbM_00423]⌈
For Offset Time Bases the StbM shall calculate the rate-corrected offset value of the local instance of the Time Base as:

$$TOL = TOG + (TV – TV_{Sync}) * r_{orc}$$

With:

- r_{orc} = Rate correction value of the Offset Time Base
- TOL = Current rate corrected offset value of the local instance of the Offset Time Base.
- TOG = newly received Offset Time Base value
- TV = Current value of the Virtual Local Time
- TV_{Sync} = Value of the Virtual Local Time when Offset Time Base value is newly received from master

This correction shall be done whenever the time is read in the scope of these functions:

- `StbM_GetCurrentTime()`
- `StbM_GetCurrentTimeExtended()`

This correction shall also be done when the StbM needs to determine the time of the local instance of the Time Base.
⌋ (SRS_StbM_20065)

[SWS_StbM_00397]⌈
For Time Bases with `StbMSynchronizedTimeBaseIdentifier` 0 to 31 (**ECUC_StbM_00021 :**) and `StbMIsSystemWideGlobalTimeMaster` = `False` (**ECUC_StbM_00036 :**), the StbM shall return on invocation of `StbM_GetRateDeviation()` the rate deviation, which has been calculated for that Time Base (i.e., r_{rc} -1 for Synchronized Time Bases or r_{orc} - 1 for Offset Time Bases).

If no rate deviation has been calculated, `StbM_GetRateDeviation()` shall return `E_NOT_OK`.
⌋ (SRS_StbM_20065)

[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 (refer **[SWS_StbM_00355]** and **[SWS_StbM_00354]**).

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.
⌋ (SRS_StbM_20065)

7.3.8.2 Offset Correction (for Time Slaves)

Offset Correction eliminates time offsets of local instances of Synchronized Time Bases. This correction takes place whenever the current time is read (e.g. in the scope of `StbM_GetCurrentTime()`). The offset is measured by the StbM when the local instance of the Time Base is synchronized in the scope of `StbM_BusSetGlobalTime()`.

[SWS_StbM_00359]⌈
For Synchronized Time Bases, the StbM shall measure the offset between its local instance of the Time Base and the Global Time Base whenever the Time Base is synchronized in the scope of the function `StbM_BusSetGlobalTime()` by taking a snapshot of the following values:

- TL_{Sync} = Value of the local instance of the Time Base before the new value of the Global Time is applied
- TV_{Sync} = Value of the Virtual Local Time

⌋ (SRS_StbM_20065)

[SWS_StbM_00355]⌈
If the absolute value of the time offset between Global Time Base and local instance of the Time Base (abs(TG - TL_{Sync})) is equal or greater than `StbMOffsetCorrectionJumpThreshold` (**ECUC_StbM_00056 :** `)`, the StbM shall calculate the corrected time (TL) of its local instance of the Time Base as shown:

$TL = TG + (TV - TV_{Sync}) * r_{rc}$

With:
- TV = Current value of the Virtual Local Time
- TV_{Sync} = Value of the Virtual Local Time as defined in **[SWS_StbM_00359]**
- TG = Received value of the Global Time
- r_{rc} = Most current rate for correcting the local instance of the Time Base

This correction shall be done whenever the time is read in the scope of these functions:
- `StbM_GetCurrentTime()`
- `StbM_GetCurrentTimeExtended()`

This correction shall also be done when the StbM needs to determine the time of the local instance of the Time Base.
⌋ (SRS_StbM_20065)

[SWS_StbM_00356]⌈
The StbM shall correct absolute time offsets between the Global Time Base and the local instance of the Time Base (abs(TG - TL_{Sync})), which are smaller than the value given by `StbMOffsetCorrectionJumpThreshold` (**ECUC_StbM_00056 :**) by temporarily applying an additional rate (r_{oc}) to r_{rc}. This rate shall be used for the duration defined by parameter `StbMOffsetCorrectionAdaptionInterval` (**ECUC_StbM_00057 :**). r_{oc} is calculated as shown:

$r_{oc} = (TG – TL_{Sync}) / (T_{CorrInt})$

With:
- $T_{CorrInt}$ = `StbMOffsetCorrectionAdaptionInterval`
- TL_{Sync} = Value of the local instance of the Time Base before the new value of the Global Time is applied
- TG = Received value of the Global Time

⌋ (SRS_StbM_20065, SRS_StbM_20067)

[SWS_StbM_00354]⌈
If the absolute time offset between Global Time Base and local instance of the Time Base (abs(TG - TL_{Sync})) is smaller than `StbMOffsetCorrectionJumpThreshold` (**ECUC_StbM_00056 :**), the StbM shall calculate the corrected time (TL) of its local instance of the Time Base **within** the period of `StbMOffsetCorrectionAdaptionInterval` (**ECUC_StbM_00057 :**) as shown:

$TL = TL_{Sync} + (TV - TV_{Sync}) * (r_{rc} + r_{oc})$

With:
- TL_{Sync} = Value of the local instance of the Time Base before the new value of the Global Time is applied
- TV = Current value of the Virtual Local Time of the Time Base
- TV_{Sync} = Value of the Virtual Local Time as defined in **[SWS_StbM_00359]**
- r_{rc} = Actual rate for correcting the local instance of the Time Base
- r_{oc} = Rate for time offset elimination via Rate Adaption

This correction shall be done whenever the time is read in the scope of these functions:
- `StbM_GetCurrentTime()`
- `StbM_GetCurrentTimeExtended()`

This correction shall also be done when the StbM needs to determine the time of the local instance of the Time Base.
⌋ (SRS_StbM_20065, SRS_StbM_20067)

[SWS_StbM_00353]⌈
If the absolute time offset between the Global Time Base and the local instance of the Time Base (abs(TG - TL_{Sync})) is smaller than

StbMOffsetCorrectionJumpThreshold (**ECUC_StbM_00056 :**), the StbM shall calculate the corrected time (TL) of its local instance of the Time Base after the period of StbMOffsetCorrectionAdaptionInterval (**ECUC_StbM_00057 :**) as specified in **[SWS_StbM_00355]**.
⌋ (SRS_StbM_20065)

[SWS_StbM_00400]⌈
If StbMOffsetCorrectionJumpThreshold (**ECUC_StbM_00056 :**) is set to 0, Offset Correction shall be performed by Jump Correction only.
⌋ (SRS_StbM_20065)

7.3.8.3 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().)

[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.
⌋ (SRS_StbM_20065)

[SWS_StbM_00411]⌈
The StbM shall apply rate correction to a Time Base, if StbMAllowMasterRateCorrection (**ECUC_StbM_00043 :**) equals TRUE and a valid rate correction value has been set by StbM_SetRateCorrection().
⌋ (SRS_StbM_20065)

[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).
⌋ (SRS_StbM_20065)

Note: The actual applied resulting rate will be the passed deviation value + 1.
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:

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

With:

- TV = Current value of the Virtual Local Time
- TV_{Sync} = Value of the Virtual Local Time at the synchronization event
- TG_{Sync} = Value of the Global Time at the synchronization event
- r_{rc} = Rate for correcting the Time Base

⌋ (SRS_StbM_20065)

Note: Synchronization events for determining TV_{Sync} and TG_{Sync} are invocations of `StbM_SetRateCorrection()`, `StbM_SetGlobalTime()`, `StbM_UpdateGlobalTime()`, and the initialization of the StbM. `StbM_SetOffset()` is an additional synchronization event for Offset Time Bases. In case of `StbM_SetRateCorrection()` TG_{Sync} is calculated as TL based on the previous TG_{Sync} value. Additional events might need to be considered for synchronization (e.g., overflow of underlying HW timers). Those should however occur not too often to avoid worsening the precision, e.g., by rounding effects.

[SWS_StbM_00422]⌈

- For Time Bases with `StbMSynchronizedTimeBaseIdentifier` 32 to 127 (**ECUC_StbM_00021 :**) and
- for Time Bases with `StbMSynchronizedTimeBaseIdentifier` 0 to 31 and `StbMIsSystemWideGlobalTimeMaster` equals `True` (**ECUC_StbM_00036 :**),

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`.
⌋ (SRS_StbM_20065)

7.3.9 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.

7.3.9.1 Time Notifications

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

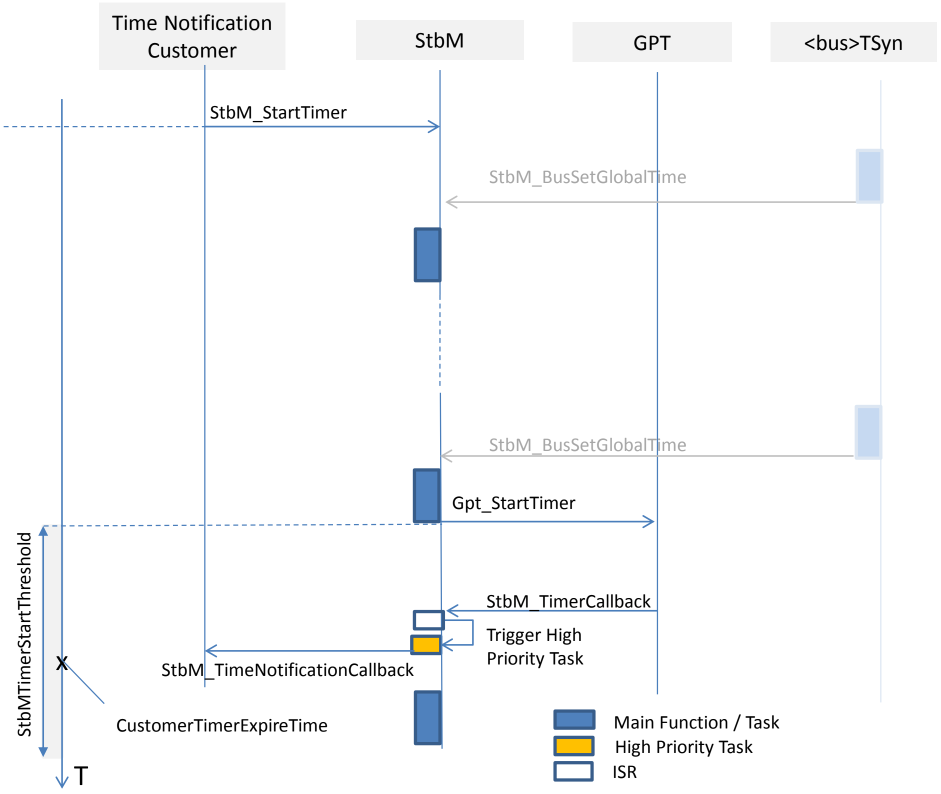


Figure 13: Basic mechanism of Time Notification

[SWS_StbM_00421][
If any `StbM_TimeNotificationCallback` is configured, the StbM shall use one additional GPT source (referenced by **ECUC_StbM_00039 :** `StbMGptTimerRef`), which is not used for other purposes.
] (SRS_StbM_20056)

[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.
] (SRS_StbM_20056)

[SWS_StbM_00335][
The StbM shall cyclically monitor the Time Bases and, if needed, re-adjust in its `StbM_MainFunction` the expiration time `CustomerTimerExpireTime` for the currently active Time Notification Customers.
] (SRS_StbM_20056)

Note: Re-adjustment is necessary, because the Time Base value could be updated asynchronously e.g. by `StbM_BusSetGlobalTime`.

[SWS_StbM_00336][

A time interval StbMTimerStartThreshold (**ECUC_StbM_00063 :**) 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 (**ECUC_StbM_00039 :**) to be notified, when the time difference has elapsed.
⌋ (SRS_StbM_20056)

Note: StbMTimerStartThreshold should be set to a value greater than StbMMainFunctionPeriod 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.

The StbM shall then call StbM_TimeNotificationCallback() to inform the corresponding Time Notification Customer and return the value of the calculated time difference by parameter deviationTime.
⌋ (SRS_StbM_20056)

Note: StbM_TimerCallback() is called in interrupt context. StbM_TimeNotificationCallback may however only be called from task context, if any of the Time Notification Customers is a SW Component. So, the StbM has to decouple the interrupt context from the task context (e.g. by triggering an ExternalTriggerOccurredEvent inside StbM_TimeNotificationCallback). The details are considered to be implementation specific.

[SWS_StbM_00337]⌈
To support N (N > 1) Customer Timers to 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
⌋ (SRS_StbM_20056)

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)

7.3.9.2 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 occurence 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 the StbM shall notify Status Notification Customers of a Time Base about status related events by the callback `StbM_StatusNotificationCallback`, which shall to be set via configuration parameter `StbMStatusNotificationCallback` (**ECUC_StbM_00046 :**).
⌋ (SRS_StbM_20001, SRS_StbM_20054)

[SWS_StbM_00279]⌈
For each Time Base the StbM has a configurable mask `StbMStatusNotificationMask` (**ECUC_StbM_00045 :**), which allows to mask individually status event notifications.
⌋ (SRS_StbM_20001, SRS_StbM_20054)

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

Status Event Name	Status Event Set Condition
`EV_GLOBAL_TIME_BASE`	1: `GLOBAL_TIME_BASE` in `timeBaseStatus` has changed from 0 to 1 0: otherwise
`EV_TIMEOUT_OCCURED`	1: `TIMEOUT` bit in `timeBaseStatus` has changed from 0 to 1 0: otherwise
`EV_TIMEOUT_REMOVED`	1: `TIMEOUT` bit in `timeBaseStatus` has changed from 1 to 0 0: otherwise
`EV_TIMELEAP_FUTURE`	1: `TIMELEAP_FUTURE` bit in `timeBaseStatus` has changed from 0 to 1 0: otherwise
`EV_TIMELEAP_FUTURE_REMOVED`	1: `TIMELEAP_FUTURE` bit in `timeBaseStatus` has changed from 1 to 0 0: otherwise
`EV_TIMELEAP_PAST`	1: `TIMELEAP_PAST` bit in `timeBaseStatus` has changed from 0 to 1 0: otherwise
`EV_TIMELEAP_PAST_REMOVED`	1: `TIMELEAP_PAST` bit in `timeBaseStatus` has changed from 1 to 0 0: otherwise
`EV_SYNC_TO_SUBDOMAIN`	1: `SYNC_TO_GATEWAY` bit in `timeBaseStatus` has changed from 0 to 1

	0: otherwise
EV_SYNC_TO_GLOBAL_MASTER	1: SYNC_TO_GATEWAY bit in timeBaseStatus 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 has been calculated (not beyond limits) 0: otherwise

⌋ (SRS_StbM_20054)

[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 NotificationMask mask is set, the corresponding bit in the NotificationEvents variable is set , i.e., NotificationEvents contains the bits for the events, which are enabled within the NotificationMask mask (refer to **[SWS_StbM_00284]**).
⌋ (SRS_StbM_20001)

[SWS_StbM_00282]⌈
If any status event (refer to **[SWS_StbM_00284]**) occurs and the corresponding bit in the NotificationMask mask is set, the StbM shall the callback function StbM_StatusNotificationCallback.

If multiple status events occur simultaneously for the same Time Base, the StbM shall call the callback function StbM_StatusNotificationCallback only once.

The StbM shall set the eventNotifications parameter of StbM_StatusNotificationCallback to the value of the NotificationEvents variable.
⌋ (SRS_StbM_20001)

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_BASE, 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 returning from the StbM_StatusNotificationCallback the StbM shall reset NotificationEvents to 0.
⌋ (SRS_StbM_20054)

7.3.10 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 Custome*r*. The module calls the OS API for synchronizing OS ScheduleTables.
⌋ (SRS_BSW_00429, SRS_StbM_20001, SRS_StbM_20002)

[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 **ECUC_StbM_00004 :** `StbMTriggeredCustomer`)
⌋ (SRS_StbM_20001, SRS_StbM_20002)

The schedule table to be synchronized is given by `StbMOSScheduleTableRef` (refer to **ECUC_StbM_00007 :**) 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.
⌋ (SRS_StbM_20002)

[SWS_StbM_00093]⌈
The triggering period `StbMTriggeredCustomerPeriod` (refer to **ECUC_StbM_00020 :**) shall be configurable for each Triggered Customer
⌋ (SRS_StbM_20001, SRS_StbM_20002)

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()`.
⌋ (SRS_StbM_20002)

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` (see `OsScheduleTable` container referenced by **ECUC_StbM_00007 :**).
⌋ (SRS_StbM_20001, SRS_StbM_20002)

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.
⌋ (SRS_StbM_20002)

[SWS_StbM_00092]⌈
The Synchronized Time-Base Manager shall synchronize only OS ScheduleTables that are in one of the states `WAITING`, `RUNNING` or `RUNNING_SYNCHRONOUS`.

This implies that the Synchronized Time-Base Manager shall check the OS for the status of the OS ScheduleTable before performing the synchronization.
⌋ (SRS_BSW_00429, SRS_StbM_20002)

Note: The Synchronized Time-Base Manager should ignore possible errors caused by the sequential execution of a) getting OS ScheduleTable status and b) performing the synchronization (e.g. someone else might have called a service to stop the OS ScheduleTable in the meantime).

7.3.11 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.

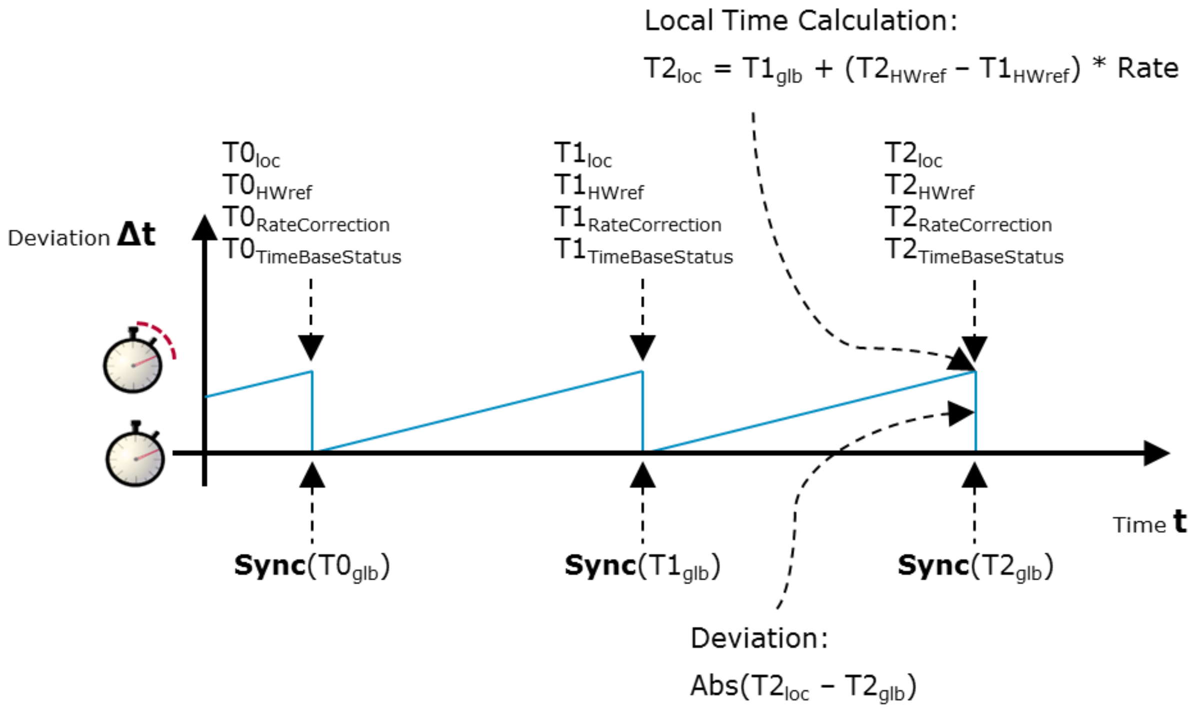


Figure 14: Simplified view how the recorded Time Base related snapshot data are taken

[SWS_StbM_00307]⌈
The StbM shall support the Global Time precision measurement, if `StbMTimeRecordingSupport` (**ECUC_StbM_00038 :**) is set to `TRUE`.
⌋ (SRS_StbM_20057)

7.3.11.1 Synchronized Time Base Record Table

[SWS_StbM_00308]⌈
If `StbMTimeRecordingSupport` (**ECUC_StbM_00038 :**) is set to `TRUE`, the StbM establishes a table to record values depending on the Synchronized Time Base with the following structure:

	Record Table Element	Multi-plicity	Range	Bytes	Type / Unit
Header		1		9	
	SynchronizedTimeDomain	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
	GlbNanoSeconds	1	0..999999999	4	StbM_TimeStampType. nanoseconds
	TimeBaseStatus	1	0..255	1	StbM_TimeStampType. StbM_TimeBaseStatusType

	HWcounter	1	0..4294967295	4	uint32 / nanoseconds
	RateDeviation	1	0..+-32000	2	StbM_RateDeviationType / ppm
	LocSeconds	1	0..4294967295	4	StbM_TimeStampType. seconds
	LocNanoSeconds	1	0..999999999	4	StbM_TimeStampType. nanoseconds
	PathDelay	1	0..4294967295	4	uint32 / nanoseconds
Block 1	…				
…					
Block (Block-Count-1)	…				

⌋ (SRS_StbM_20057)

[SWS_StbM_00309]⌈
If StbMTimeRecordingSupport (**ECUC_StbM_00038 :**) is set to TRUE, StbMClockfrequency (**ECUC_StbM_00051 :**) shall be mapped to the Header Element HWfrequency of the table belonging to the Synchronized Time Base.
⌋ (SRS_StbM_20057)

[SWS_StbM_00310]⌈
If StbMTimeRecordingSupport (**ECUC_StbM_00038 :**) is set to TRUE, StbMClockprescaler (**ECUC_StbM_00052 :**) shall be mapped to the Header Element HWprescaler of the table belonging to the Synchronized Time Base.
⌋ (SRS_StbM_20057)

[SWS_StbM_00382]⌈
If StbMTimeRecordingSupport (**ECUC_StbM_00038 :**) is set to TRUE, StbMSyncTimeRecordTableBlockCount (**ECUC_StbM_00058 :**) shall be used to increase the number of blocks of the Synchronized Time Base Record Table.
⌋ (SRS_StbM_20057)

7.3.11.2 Offset Time Base Record Table

[SWS_StbM_00311]⌈
If StbMTimeRecordingSupport (**ECUC_StbM_00038 :**) is set to TRUE, the StbM establishes a table to record values depending on the Offset Time Base with the following structure:

	Record Table Element	**Multi-plicity**	**Range**	**Bytes**	**Type / Unit**
Header		1		1	
	OffsetTimeDomain	1	16..31	1	uint8
Block 0		1		9	

	GlbSeconds	1	0..4294967295	4	StbM_TimeStampType. seconds
	GlbNanoSeconds	1	0..999999999	4	StbM_TimeStampType. nanoseconds
	TimeBaseStatus	1	0..255	1	StbM_TimeStampType. StbM_TimeBaseStatusType
Block 1	…				
…					
Block (Block-Count-1)	…				

⌋ (SRS_StbM_20057)

[SWS_StbM_00383]⌈
If StbMTimeRecordingSupport (**ECUC_StbM_00038 :**) is set to TRUE, StbMOffsetTimeRecordTableBlockCount (**ECUC_StbM_00059 :**) shall be used to increase the number of blocks of the Offset Time Base Record Table.
⌋ (SRS_StbM_20057)

7.3.11.3 Record Table Snapshot Conditions

[SWS_StbM_00312]⌈
If StbMTimeRecordingSupport (**ECUC_StbM_00038 :**) is set to TRUE, 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 StbM_SyncTimeRecordBlockCallback() or StbM_OffsetTimeRecordBlockCallback() did release all blocks with their elements, the StbM shall again overwrite the Block index 0 (zero) with the incoming measurement data.
⌋ (SRS_StbM_20057)

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. Old data are kept to give the Tester a chance to recognize a jump in time backwards, which implies a potential misconfiguration of StbMSyncTimeRecordTableBlockCount (**ECUC_StbM_00058 :**) or StbMOffsetTimeRecordTableBlockCount (**ECUC_StbM_00059 :**).

[SWS_StbM_00313]⌈
For Synchronized Time Bases, if StbMTimeRecordingSupport (**ECUC_StbM_00038 :**) is set to TRUE, on an invocation of StbM_BusSetGlobalTime() the StbM shall write the block elements LocSeconds and LocNanoSeconds of the related measurement recording table before updating the Local Time Base by the Global Time Base.
⌋ (SRS_StbM_20057)

[SWS_StbM_00314]⌈
For Synchronized Time Bases, if StbMTimeRecordingSupport (**ECUC_StbM_00038 :**) is set to TRUE, on an invocation of StbM_BusSetGlobalTime() the StbM shall write the block elements GlbSeconds, GlbNanoSeconds, HWcounter, RateDeviation, TimeBaseStatus and PathDelay to the related measurement recording table after updating the Local Time Base by the Global Time Base.
⌋ (SRS_StbM_20057)

Note: PathDelay will be retrieved from the <Bus>TSyn module via PathDelay member of parameter measureData of StbM_BusSetGlobalTime().

[SWS_StbM_00388]⌈
For Offset Time Bases, if StbMTimeRecordingSupport (**ECUC_StbM_00038 :**) is set to TRUE, on an invocation of StbM_BusSetGlobalTime() the StbM shall write the block elements GlbSeconds, GlbNanoSeconds and TimeBaseStatus to the related measurement recording table.
⌋ (SRS_StbM_20057)

[SWS_StbM_00315]⌈
If StbMTimeRecordingSupport (**ECUC_StbM_00038 :**) is set to TRUE, the application collects the contents of the header of the Synchronized Time Base Record Table by calling StbM_GetSyncTimeRecordHead().
⌋ (SRS_StbM_20057)

[SWS_StbM_00316]⌈
If StbMTimeRecordingSupport (**ECUC_StbM_00038 :**) is set to TRUE, the application collects the contents of the header of the Offset Time Base Record Table by calling StbM_GetOffsetTimeRecordHead().
⌋ (SRS_StbM_20057)

[SWS_StbM_00317]⌈
If StbMTimeRecordingSupport (**ECUC_StbM_00038 :**) is set to TRUE, the StbM notifies the Application by calling StbM_SyncTimeRecordBlockCallback() in the next StbM_MainFunction() call cycle block by block for all available blocks, starting with block index 0 (zero), if all elements of each block belonging to the Synchronized Time Base Record Table have been updated.
⌋ (SRS_StbM_20057)

[SWS_StbM_00318]⌈
If StbMTimeRecordingSupport (**ECUC_StbM_00038 :**) is set to TRUE, the StbM notifies the Application by calling StbM_OffsetTimeRecordBlockCallback() in the next StbM_MainFunction() call cycle block by block for all available blocks, starting with block index 0 (zero), if all elements of the block belonging to the Offset Time Base Record Table have been updated.
⌋ (SRS_StbM_20057)

7.3.12 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 [16].

7.4 Error Handling

[SWS_StbM_00031]⌈
If a triggered customer is configured (refer to **ECUC_StbM_00004 :** `StbMTriggeredCustomer`), the Synchronized Time-Base Manager shall monitor the cyclic execution of the `StbM_MainFunction()` (see section 8.1.3.22). This is to guarantee cyclic synchronization of OS schedule tables.
⌋ (SRS_StbM_20007)

[SWS_StbM_00198]⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) is set to `TRUE`, all StbM API services other then StbM_Init() and `StbM_GetVersion()` shall

- not execute their normal operation
- report to DET the development error `STBM_E_NOT_INITIALIZED` and
- return `E_NOT_OK`

unless the StbM has been initialized with a preceding call of `StbM_Init()`.
⌋ (SRS_BSW_00337, SRS_BSW_00386, SRS_BSW_00327)

[SWS_StbM_00199]⌈
For any StbM API service other then `StbM_Init()` and `StbM_GetVersion()` all out parameters shall remain untouched, if an error occurs during execution of that API service.
⌋ (SRS_StbM_20007)

For further details refer to the chapter 7.2 "Error Handling" in *SWS_BSWGeneral* and chapter 8 for API specific error handling.

7.5 Error Classification

7.5.1 Development Errors

[SWS_StbM_00041]⌈
The following errors and exceptions shall be detectable by the Synchronized Time-Base Manager depending on its build version (development/production mode).

Type or error	***Related error code***	***Value [hex]***
StbM_Init called with an invalid	`STBM_E_INIT_FAILED`	0x11

configuration pointer		
API called while StbM is not initialized	`STBM_E_NOT_INITIALIZED`	0x0B
API called with wrong parameter	`STBM_E_PARAM`	0x0A
API called with invalid pointer in parameter list	`STBM_E_PARAM_POINTER`	0x10
API disabled by configuration	`STBM_E_SERVICE_DISABLED`	0x12

⌋ (SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_00327, SRS_BSW_00323)

Note:
There exist errors, which are of interest for the user of the Synchronized Time-Base Manager, but the source of failure is somewhere else (e.g. the FlexRay Time Base is not synchronized). Thus, they do not appear in the above-mentioned error list and the Synchronized Time-Base Manager does not perform an error handling for those kinds of errors.

7.5.2 Runtime Errors

No Runtime Errors defined.

7.5.3 Transient Faults

No Transient Faults defined.

7.5.4 Production Errors

No Production Errors defined.

7.5.5 Extended Production Errors

No Extended Production Errors defined.

7.6 Version Check

For details refer to the chapter 5.1.8 “Version Check” in *SWS_BSWGeneral.*

8 API specification

8.1 API

8.1.1 Imported types

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

[SWS_StbM_00051] ⌈

Module	***Imported Type***
Eth_GeneralTypes	Eth_TimeStampQualType
	Eth_TimeStampType
Os	CounterType
	ScheduleTableStatusRefType
	ScheduleTableType
	StatusType
	TickRefType
	TickType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋ (SRS_BSW_00301)

8.1.2 Type definitions

8.1.2.1 StbM_ConfigType

[SWS_StbM_00249] ⌈

Name:	`StbM_ConfigType`	
Type:	`Structure`	
Range:	`implementation specific`	--
Description:	Configuration data structure of the StbM module.	

⌋ (SRS_BSW_00414)

8.1.2.2 StbM_TimeStampRawType

[SWS_StbM_00194] ⌈

Name:	`StbM_TimeStampRawType`		
Type:	`uint32`		
Range:	`0..4294967295`	`--`	nanoseconds (0x000 00000 .. 0xFFFF FFFF)
Description:	Variables of this type are used for expressing time stamps in raw format in nanoseconds only.		

⌋ (SRS_StbM_20025)

8.1.2.3 StbM_MeasurementType

[SWS_StbM_00384] ⌈

Name:	`StbM_MeasurementType`		
Type:	`Structure`		
Element:	`uint32`	`pathDelay`	Propagation delay in nanoseconds
Description:	Structure which contains additional measurement data		

⌋ (SRS_StbM_20057)

8.1.3 Function definitions

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

8.1.3.1 StbM_GetVersionInfo

[SWS_StbM_00066] ⌈

Service name:	StbM_GetVersionInfo	

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

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.	

⌋ (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)

8.1.3.2 StbM_Init

[SWS_StbM_00052] ⌈

Service name:	StbM_Init	

Syntax:	`void StbM_Init(`
	`    const StbM_ConfigType* ConfigPtr`
	`)`	

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	

⌋ (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)

8.1.3.3 StbM_GetCurrentTime

[SWS_StbM_00195] ⌈

Service name:	StbM_GetCurrentTime	

Syntax:	`Std_ReturnType StbM_GetCurrentTime(`
	`    StbM_SynchronizedTimeBaseType timeBaseId,`
	`    StbM_TimeStampType* timeStamp,`
	`    StbM_UserDataType* userData`
	`)`	

Service ID[hex]:	0x07	
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 standard format.	

⌋ (SRS_StbM_20003, SRS_StbM_20013, SRS_StbM_20023, SRS_StbM_20029)

[SWS_StbM_00196]⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) 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` (**ECUC_StbM_00012 :**) 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 `timeStamp` or `userData`.

⌋ (SRS_BSW_00386, SRS_BSW_00323)

8.1.3.4 StbM_GetCurrentTimeExtended

[SWS_StbM_00200] ⌈

Service name:	StbM_GetCurrentTimeExtended	

Syntax:	`Std_ReturnType StbM_GetCurrentTimeExtended(`
	`    StbM_SynchronizedTimeBaseType timeBaseId,`
	`    StbM_TimeStampExtendedType* timeStamp,`
	`    StbM_UserDataType* userData`
	`)`	

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.	

⌋ (SRS_StbM_20003, SRS_StbM_20029)

[SWS_StbM_00201]⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) is set to `TRUE`, `StbM_GetCurrentTimeExtended()` 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_00202]⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) is set to `TRUE`, `StbM_GetCurrentTimeExtended()` 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)

8.1.3.5 StbM_GetCurrentTimeRaw

[SWS_StbM_00205] ⌈

Service name:	StbM_GetCurrentTimeRaw

Syntax:	`Std_ReturnType StbM_GetCurrentTimeRaw(`
	`    StbM_SynchronizedTimeBaseType timeBaseId,`
	`    StbM_TimeStampRawType* timeStampRawPtr`
	`)`

Service ID[hex]:	0x09
Sync/Async:	Synchronous

Reentrancy:	Non Reentrant	
Parameters (in):	timeBaseId	Time Base reference
Parameters (inout):	None	
Parameters (out):	timeStampRawPtr	Current time stamp that is valid at this time
Return value:	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description:	Returns nanosecond part of the Virtual Local Time of the referenced Time Base.	

⌋ (SRS_StbM_20013, SRS_StbM_20016)

[SWS_StbM_00206]⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) is set to `TRUE`, `StbM_GetCurrentTimeRaw()` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with a NULL pointer for parameter `timeStampRawPtr`.
⌋ (SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00417]⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) is set to `TRUE`, `StbM_GetCurrentTimeRaw()` shall report to DET the development error `STBM_E_PARAM`, if called with a parameter `timeBaseId`, which

- is referring to Offset time base
- is not configured or
- is within the reserved value range.

⌋ (SRS_BSW_00386, SRS_BSW_00323)

8.1.3.6 StbM_GetCurrentTimeDiff

[SWS_StbM_00209] ⌈

Service name:	StbM_GetCurrentTimeDiff	

Syntax:	`Std_ReturnType StbM_GetCurrentTimeDiff(`
	`StbM_SynchronizedTimeBaseType timeBaseId,`
	`StbM_TimeStampRawType givenTimeStamp,`
	`StbM_TimeStampRawType* timeStampDiffPtr`
	`)`	

Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	timeBaseId	Time Base reference
	givenTimeStamp	Given time stamp as difference calculation basis
Parameters (inout):	None	
Parameters (out):	timeStampDiffPtr	Time difference of current time stamp that is valid at this time minus given time stamp
Return value:	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description:	Returns time difference of the nanoseconds part of the Virtual Local Time of the referenced Time Base minus the time given by the parameter givenTimeStamp.	

⌋ (SRS_StbM_20013, SRS_StbM_20016)

[SWS_StbM_00210]⌈

If the switch StbMDevErrorDetect (ECUC_StbM_00012 :) is set to TRUE, StbM_GetCurrentTimeDiff() shall report to DET the development error STBM_E_PARAM_POINTER, if called with a NULL pointer for parameter timeStampDiffPtr.
⌋ (SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00418]⌈
If the switch StbMDevErrorDetect (**ECUC_StbM_00012 :**) is set to TRUE, StbM_GetCurrentTimeDiff() shall report to DET the development error STBM_E_PARAM, if called with a parameter timeBaseId, which

- is referring to Offset time base
- is not configured or
- is within the reserved value range.

⌋ (SRS_BSW_00386, SRS_BSW_00323)

8.1.3.7 StbM_SetGlobalTime

[SWS_StbM_00213] ⌈

Service name:	StbM_SetGlobalTime	

Syntax:	Std_ReturnType StbM_SetGlobalTime(
	    StbM_SynchronizedTimeBaseType timeBaseId,
	    const StbM_TimeStampType* timeStamp,
	    const StbM_UserDataType* userData
	)	

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.	

⌋ (SRS_StbM_20023, SRS_StbM_20026)

[SWS_StbM_00214]⌈
If the switch StbMDevErrorDetect (**ECUC_StbM_00012 :**) 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` (ECUC_StbM_00012 :) 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)

8.1.3.8 StbM_UpdateGlobalTime

[SWS_StbM_00385] ⌈

Service name:	StbM_UpdateGlobalTime	

Syntax:	`Std_ReturnType StbM_UpdateGlobalTime(`
	`    StbM_SynchronizedTimeBaseType timeBaseId,`
	`    const StbM_TimeStampType* timeStamp,`
	`    const StbM_UserDataType* userData`
	`)`	

Service ID[hex]:	0x10	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	timeBaseId	time base reference
	timeStamp	New time stamp
	userData	New user data (if not NULL)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description:	Allows the Customers to set the Global Time that will be sent to the buses. This function will be used if a Time Master is present in this ECU. Using UpdateGlobalTime will not lead to an immediate transmission of the Global Time.	

⌋ (SRS_StbM_20026)

[SWS_StbM_00340]⌈
If the switch `StbMDevErrorDetect` (ECUC_StbM_00012 :) 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` (ECUC_StbM_00012 :) 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)

8.1.3.9 StbM_SetUserData

[SWS_StbM_00218] ⌈

Service name:	StbM_SetUserData

Syntax:	`Std_ReturnType StbM_SetUserData(`
	`StbM_SynchronizedTimeBaseType timeBaseId,`
	`const StbM_UserDataType* userData`
	`)`	

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.	

⌋ (SRS_StbM_20030)

[SWS_StbM_00219]⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) 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` (ECUC_StbM_00012 :) 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)

8.1.3.10 StbM_SetOffset

[SWS_StbM_00223] ⌈

Service name:	StbM_SetOffset	

Syntax:	`Std_ReturnType StbM_SetOffset(`
	`StbM_SynchronizedTimeBaseType timeBaseId,`
	`const StbM_TimeStampType* timeStamp,`
	`const StbM_UserDataType* userData`
	`)`	

Service ID[hex]:	0x0d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	timeBaseId	time base reference
	timeStamp	New offset time stamp
	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 and the Timesync Modules to set the Offset Time and the User Data.

⌋ (SRS_StbM_20023, SRS_StbM_20028)

[SWS_StbM_00224]⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) 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` (ECUC_StbM_00012 :) 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` or `userData`.
⌋ (SRS_BSW_00386, SRS_BSW_00323)

8.1.3.11 StbM_GetOffset

[SWS_StbM_00228] ⌈

Service name:	StbM_GetOffset	

Syntax:	`Std_ReturnType StbM_GetOffset(`
	`    StbM_SynchronizedTimeBaseType timeBaseId,`
	`    StbM_TimeStampType* timeStamp,`
	`    StbM_UserDataType* userData`
	`)`	

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.	

⌋ (SRS_StbM_20027)

[SWS_StbM_00229]⌈
If the switch `StbMDevErrorDetect` (ECUC_StbM_00012 :) 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` (ECUC_StbM_00012 :) 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)

8.1.3.12 StbM_BusSetGlobalTime

[SWS_StbM_00233] ⌈

Service name:	StbM_BusSetGlobalTime	

Syntax:	`Std_ReturnType StbM_BusSetGlobalTime(`
	`    StbM_SynchronizedTimeBaseType timeBaseId,`
	`    const StbM_TimeStampType* timeStampPtr,`
	`    const StbM_UserDataType* userDataPtr,`
	`    const StbM_MeasurementType* measureDataPtr`
	`)`	

Service ID[hex]:	0x0f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	timeBaseId	Time Base reference
	timeStampPtr	New time stamp
	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 a new Global Time value to the StbM, which has been received from a bus.	

⌋ (SRS_StbM_20014, SRS_StbM_20023, SRS_StbM_20057)

[SWS_StbM_00234]⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) is set to `TRUE`, `StbM_BusSetGlobalTime()` 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

⌋ (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 (ECUC_StbM_00012 :) 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

- timeStampPtr.
- measureDataPtr

⌋ (SRS_BSW_00386, SRS_BSW_00323)

8.1.3.13 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	

⌋ (SRS_StbM_20065)

[SWS_StbM_00379]⌈
If the switch StbMDevErrorDetect (ECUC_StbM_00012 :) 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 (ECUC_StbM_00012 :) 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)

8.1.3.14 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:	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).	

⌋ (SRS_StbM_20065)

[SWS_StbM_00391]⌈
If the switch `StbMDevErrorDetect` (ECUC_StbM_00012 :) 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` (ECUC_StbM_00012 :) 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)

8.1.3.15 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.

⌋ (SRS_StbM_20003)

[SWS_StbM_00268]⌈
If the switch `StbMDevErrorDetect` (ECUC_StbM_00012 :) 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` (ECUC_StbM_00012 :) 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)

8.1.3.16 StbM_GetTimeBaseStatus

[SWS_StbM_00263] ⌈

Service name:	StbM_GetTimeBaseStatus	

Syntax:	`Std_ReturnType StbM_GetTimeBaseStatus(`
	`    StbM_SynchronizedTimeBaseType timeBaseId,`
	`    StbM_TimeBaseStatusType* syncTimeBaseStatus,`
	`    StbM_TimeBaseStatusType* offsetTimeBaseStatus`
	`)`	

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 Time Base
	offsetTimeBaseStatus	Status of the Offset Time Base
Return value:	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description:	Returns the detailed status of the Time Base. For Offset Time Bases the status of the Offset Time Base itself and the status of the underlying Synchronized Time Base is returned.	

⌋ (SRS_StbM_20003)

[SWS_StbM_00264]⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) 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` (ECUC_StbM_00012 :) 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)

8.1.3.17 StbM_StartTimer

[SWS_StbM_00272] ⌈

Service name:	StbM_StartTimer	

Syntax:	`Std_ReturnType StbM_StartTimer(`
	`    StbM_SynchronizedTimeBaseType timeBaseId,`
	`    StbM_CustomerIdType customerId,`
	`    const StbM_TimeStampType* expireTime`
	`)`	

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	

⌋ (SRS_StbM_20056)

[SWS_StbM_00296]⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) 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` (**ECUC_StbM_00012 :**) 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` (ECUC_StbM_00012 :) 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)

8.1.3.18 StbM_GetSyncTimeRecordHead

[SWS_StbM_00319] ⌈

Service name:	StbM_GetSyncTimeRecordHead	

Syntax:	`Std_ReturnType StbM_GetSyncTimeRecordHead(`
	`    StbM_SynchronizedTimeBaseType timeBaseId,`
	`    StbM_SyncRecordTableHeadType* syncRecordTableHead`
	`)`	

Service ID[hex]:	0x16	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	timeBaseId	Time Base reference
Parameters (inout):	None	
Parameters (out):	syncRecordTableHead	Header of the table
Return value:	Std_ReturnType	E_OK: Table access done E_NOT_OK: Table contains no data or access invalid
Description:	Accesses to the recorded snapshot data Header of the table belonging to the Synchronized Time Base.	

⌋ (SRS_StbM_20057)

[SWS_StbM_00320]⌈
The function StbM_GetSyncTimeRecordHead() shall be pre compile time configurable ON/OFF by the configuration parameter: StbMTimeRecordingSupport (**ECUC_StbM_00038 :**).
⌋ (SRS_StbM_20057)

[SWS_StbM_00394]⌈
If the switch StbMDevErrorDetect (**ECUC_StbM_00012 :**) 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 a Offset Time Base or
- is within the reserved value range.

⌋ (SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00405]⌈
If the switch StbMDevErrorDetect (ECUC_StbM_00012 :) is set to TRUE, 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)

8.1.3.19 StbM_GetOffsetTimeRecordHead

[SWS_StbM_00325] ⌈

Service name:	StbM_GetOffsetTimeRecordHead	

Syntax:	`Std_ReturnType StbM_GetOffsetTimeRecordHead(`
	`StbM_SynchronizedTimeBaseType timeBaseId,`
	`StbM_OffsetRecordTableHeadType* offsetRecordTableHead`
	`)`	

Service ID[hex]:	0x17	
Sync/Async:	Synchronous	
Reentrancy:	Non 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.	

⌋ (SRS_StbM_20057)

[SWS_StbM_00326]⌈
The function `StbM_GetOffsetTimeRecordHead()` shall be pre compile time configurable `ON`/`OFF` by the configuration parameter: `StbMTimeRecordingSupport` **(ECUC_StbM_00038 :)** .
⌋ (SRS_StbM_20057)

[SWS_StbM_00327]⌈
If the switch `StbMDevErrorDetect` (ECUC_StbM_00012 :) 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` (ECUC_StbM_00012 :) is set to `TRUE`, `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)

8.1.3.20 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]	

⌋ (SRS_StbM_20064)

[SWS_StbM_00349]⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) 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)

8.1.3.21 StbM_GetTimeBaseUpdateCounter

[SWS_StbM_00347] ⌈

Service name:	StbM_GetTimeBaseUpdateCounter	

Syntax:	`uint8 StbM_GetTimeBaseUpdateCounter(`
	`    StbM_SynchronizedTimeBaseType timeBaseId`
	`)`	

Service ID[hex]:	0x1b	
Sync/Async:	Synchronous	
Reentrancy:	Non 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.	

⌋ (SRS_StbM_20064)

[SWS_StbM_00348]⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) 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)

8.1.3.22 StbM_GetMasterConfig

[SWS_StbM_91002] ⌈

Service name:	StbM_GetMasterConfig	

Syntax:	`Std_ReturnType StbM_GetMasterConfig(`
	`    StbM_SynchronizedTimeBaseType timeBaseId,`
	`    StbM_MasterConfigType* masterConfig`
	`)`	

Service ID[hex]:	0x1d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	timeBaseId	Time Base reference
Parameters (inout):	None	
Parameters (out):	masterConfig	Indicates, if system wide master functionality is supported
Return value:	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description:	Indicates if the functionality for a system wide master (e.g. StbM_SetGlobalTime) for a given Time Base is available or not.	

⌋ (SRS_StbM_20023)

[SWS_StbM_00415]⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) 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` (**ECUC_StbM_00012 :**) 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)

8.1.4 Scheduled functions

8.1.4.1 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.

⌋ (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.
⌋ (SRS_StbM_20002, SRS_BSW_00333)

8.1.5 Callback Functions

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

8.1.5.1 StbM_TimerCallback

[SWS_StbM_00257] ⌈

Service name:	StbM_TimerCallback

Syntax:	`void StbM_TimerCallback(`
	`    void`
	`)`

Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Notifies the StbM, that the GPT timer, which is used to trigger the StbM_TimeNotificationCallback has expired.

⌋ (SRS_StbM_20056)

8.1.6 Expected Interfaces

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

8.1.6.1 Mandatory Interfaces

This chapter defines all interfaces which are required to fulfill the core functionality of the Synchronized Time-Base Manager.

[SWS_StbM_00058] ⌈

API function	*Description*
Det_ReportError	Service to report development errors.

⌋ (SRS_BSW_00301, SRS_BSW_00339)

8.1.6.2 Optional Interfaces

This chapter defines all interfaces which are required to fulfill an optional functionality of the Synchronized Time-Base Manager.

[SWS_StbM_00059] ⌈

API function	*Description*
EthIf_GetCurrentTime	Returns a time value out of the HW registers according to the capability of the HW. Is the HW resolution is lower than the Eth_TimeStampType resolution resp. range, the remaining bits will be filled with 0.
GetCounterValue	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	This service gets the number of ticks between the current tick value and a previously read tick value.
GetScheduleTableStatus	This service queries the state of a schedule table (also with respect to synchronization).
NextScheduleTable	This service switches the processing from one schedule table to another schedule table.
SetScheduletableAsync	This service stops synchronization of a schedule table.
StartScheduleTableAbs	This service starts the processing of a schedule table at an absolute value "Start" on the underlying counter.
StartScheduleTableRel	This service starts the processing of a schedule table at "Offset" relative to the "Now" value on the underlying counter.
StartScheduleTableSynchron	This service starts an explicitly synchronized schedule table synchronously.
StopScheduleTable	This service cancels the processing of a schedule table immediately at any point while the schedule table is running.
SyncScheduleTable	This service provides the schedule table with a synchronization count and start synchronization.

⌋ (SRS_BSW_00301, SRS_BSW_00339)

8.1.6.3 Configurable Interfaces

8.1.6.3.1 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.	

⌋ (SRS_StbM_20057)

[SWS_StbM_00323]⌈
The function `SyncTimeRecordBlockCallback<timeBaseId>()` shall be set by the parameter `StbMSyncTimeRecordBlockCallback` (**ECUC_StbM_00060 :**).
⌋ (SRS_StbM_20057)

[SWS_StbM_00403]{OBSOLETE}⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) is set to `TRUE`, `SyncTimeRecordBlockCallback` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with an invalid pointer of parameter `syncRecordTableBlock`.
⌋ (SRS_BSW_00386, SRS_BSW_00323)

8.1.6.3.2 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.	

⌋ (SRS_StbM_20057)

[SWS_StbM_00329]⌈
The function `OffsetTimeRecordBlockCallback<timeBaseId>` shall set by the parameter `StbMOffsetTimeRecordBlockCallback` (**ECUC_StbM_00061 :**).
⌋ (SRS_StbM_20057)

[SWS_StbM_00402]{OBSOLETE}⌈
If the switch `StbMDevErrorDetect` (**ECUC_StbM_00012 :**) is set to `TRUE`, `OffsetTimeRecordBlockCallback` shall report to DET the development error

STBM_E_PARAM_POINTER, if called with an invalid pointer of parameter offsetRecordTableBlock.
⌋ (SRS_BSW_00386, SRS_BSW_00323)

8.1.6.3.3 StatusNotificationCallback

[SWS_StbM_00285] ⌈

Service name:	StatusNotificationCallback<TimeBase>	

Syntax:	`Std_ReturnType StatusNotificationCallback<TimeBase>(`
	`    StbM_TimeBaseNotificationType eventNotification`
	`)`	

Service ID[hex]:	0x19	
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	

⌋ (SRS_StbM_20001, SRS_StbM_20054, SRS_BSW_00457, SRS_BSW_00360, SRS_BSW_00333)

[SWS_StbM_00299]⌈
The status notification callback function shall be set by the parameter StbMStatusNotificationCallback (**ECUC_StbM_00046 :**).
⌋ (SRS_StbM_20054)

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.

8.1.6.3.4 <Customer>_TimeNotificationCallback

[SWS_StbM_00273] ⌈

Service name:	<Customer>_TimeNotificationCallback<TimeBase>	

Syntax:	`Std_ReturnType`
	`<Customer>_TimeNotificationCallback<TimeBase>(`
	`    StbM_TimeDiffType deviationTime`
	`)`	

Service ID[hex]:	0x18	
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>	

⌋ (SRS_StbM_20056, SRS_BSW_00457 , SRS_BSW_00360, SRS_BSW_00333)

[SWS_StbM_00274]⌈
The event notification callback function shall be set by the parameter `StbMTimeNotificationCallback` (**ECUC_StbM_00064 :** .)
⌋ (SRS_StbM_20056)

8.2 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.

8.2.1 Provided Ports

8.2.1.1 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)}		

⌋ (SRS_StbM_20003, SRS_StbM_20010, SRS_StbM_20023, SRS_StbM_20026, SRS_StbM_20028, SRS_StbM_20030)

8.2.1.2 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/ StbMSynchronizedTimeBaseIdentifier.value)}
Variation	Name = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}	

⌋ (SRS_StbM_20003, SRS_StbM_20010, SRS_StbM_20029)

8.2.1.3 GlobalTime_StatusEvent

[SWS_StbM_00290] ⌈

Name	GlobalTime_StatusEvent_{TBName}		
Kind	ProvidedPort	Interface	StatusNotification
Description	--		
Variation	({ecuc(StbM/StbMSynchronizedTimeBase/StbMStatusNotificationCallback)} != NULL) TBName = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		

⌋ (SRS_StbM_20010, SRS_StbM_20054)

8.2.1.4 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/ StbMSynchronizedTimeBaseIdentifier.value)}	
	Type	StbM_CustomerIdType	
	Value	{ecuc(StbM/StbMSynchronizedTimeBase/ StbMNotificationCustomer/StbMNotificationCustomerId.value)}	

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

⌋ (SRS_StbM_20056)

8.2.2 Required Ports

8.2.2.1 GlobalTime_TimeEvent

[SWS_StbM_00276] ⌈

Name	GlobalTime_TimeEvent_{TBName}_{CName}		
Kind	RequiredPort	Interface	TimeNotification
Description	--		

Variation	({ecuc(StbM/StbMSynchronizedTimeBase/StbMNotificationCustomer/StbMTimeNotificationCallback)}!=NULL)
	TBName={ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}
	CName={ecuc(StbM/StbMSynchronizedTimeBase/StbMNotificationCustomer.SHORT-NAME)}		

⌋ (SRS_StbM_20010, SRS_StbM_20056)

8.2.2.2 GlobalTime_Measurement

[SWS_StbM_00387] ⌈

Name	MeasurementNotification_{TBName}		
Kind	RequiredPort	Interface	MeasurementNotification_{TB_Name}
Description	--		
Variation	({ecuc(StbM/StbMGeneral/StbMTimeRecordingSupport)} == True) &&({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 32) TBName={ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		

⌋ (SRS_StbM_20057)

8.2.3 Sender-Receiver Interfaces

8.2.3.1 StatusNotification

[SWS_StbM_00286] ⌈

Name	StatusNotification	
Comment	Notification about a Time Base related status change	
IsService	false	
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 128	
Data Elements	eventNotification	
	Type	StbM_TimeBaseNotificationType

	Variation	--

⌋ (SRS_StbM_20010, SRS_StbM_20054)

8.2.4 Client-Server-Interfaces

8.2.4.1 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
	1	E_NOT_OK

Operations

GetMasterConfig			
Comments	Indicates in postbuild use case, if the StbM is actually configured as system wide master		
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMAllowSystemWideGlobalTimeMaster)} != NULL		
Parameters	masterConfig	Comment	--
		Type	StbM_MasterConfigType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
SetGlobalTime			
Comments	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 SetGlobalTimecan lead to an immediate		

	transmission of the Global Time.		
Variation	--		
Parameters	timeStamp	Comment	--
		Type	StbM_TimeStampType
		Variation	--
		Direction	IN
	userData	Comment	--
		Type	StbM_UserDataType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
SetOffset			
Comments	Allows the Customers and the Timesync Modules to set the Offset Time.		
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} > 15 &&{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 32		
Parameters	timeStamp	Comment	--
		Type	StbM_TimeStampType
		Variation	--
		Direction	IN
	userData	Comment	--
		Type	StbM_UserDataType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
SetRateCorrection			
Comments	Allows to set the rate of a Synchronized Time Base (being either a Pure Local Time Base or not).		

Variation	--		
Parameters	rateDeviation	Comment	Value of the applied rate deviation
		Type	StbM_RateDeviationType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	

SetUserData			
Comments	Allows the Customers to set the User Data that will be sent to the buses.		
Variation	--		
Parameters	userData	Comment	New user data
		Type	StbM_UserDataType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	

TriggerTimeTransmission		
Comments	Allows the Customers to force the Timesync Modules to transmit the current Time Base due to an incremented timeBaseUpdateCounter	
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 32	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	Operation failed

UpdateGlobalTime			
Comments	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.		
Variation	--		
Parameters	timeStamp	Comment	--

		Type	StbM_TimeStampType
		Variation	--
		Direction	IN
	userData	Comment	--
		Type	StbM_UserDataType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	

⌋ (SRS_StbM_20003, SRS_StbM_20010, SRS_StbM_20026, SRS_StbM_20028, SRS_StbM_20030, SRS_StbM_20064)

8.2.4.2 GlobalTime_Slave

[SWS_StbM_00247] ⌈

Name	GlobalTime_Slave_{Name}	
Comment	--	
IsService	true	
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 128 Name = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK

Operations

GetCurrentTime			
Comments	Returns a time value (Local Time Base derived from Global Time Base) in standard format.		
Variation	--		
Parameters	timeStamp	Comment	--
		Type	StbM_TimeStampType
		Variation	--
		Direction	OUT
	userData	Comment	--

		Type	StbM_UserDataType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
GetCurrentTimeExtended			
Comments	Returns a time value (Local Time Base derived from Global Time Base) in extended format.		
Variation	{ecuc(StbM/StbMGeneral/StbMGetCurrentTimeExtendedAvailable)}		
Parameters	timeStamp	Comment	--
		Type	StbM_TimeStampExtendedType
		Variation	--
		Direction	OUT
	userData	Comment	--
		Type	StbM_UserDataType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
GetOffsetTimeRecordHead			
Comments	Reads the header of the table with recorded measurement data belonging to the Offset Time Base		
Variation	{ecuc(StbM/StbMGeneral/StbMTimeRecordingSupport)} == True &&{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} > 15 &&{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 32		
Parameters	offsetRecordTableHead	Comment	Header of the table
		Type	StbM_OffsetRecordTableHeadType
		Variation	--
		Direction	OUT
Possible	E_OK	Operation successful	

Errors	E_NOT_OK	Operation failed	
GetRateDeviation			
Comments	Returns value of the current rate deviation of a Time Base		
Variation	--		
Parameters	rateDeviation	Comment	Value of the current rate deviation of a Time Base
		Type	StbM_RateDeviationType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
GetSyncTimeRecordHead			
Comments	Reads the header of the table with recorded measurement data belonging to the Synchronized Time Base		
Variation	({ecuc(StbM/StbMGeneral/StbMTimeRecordingSupport)} == True) &&({ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 16)		
Parameters	syncRecordTableHead	Comment	Header of the table
		Type	StbM_SyncRecordTableHeadType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Record head read successfully.	
	E_NOT_OK	Read access to record head failed.	
GetTimeBaseStatus			
Comments	Returns detailed status information for a Synchronized Time Base and, if called for an Offset Time Base, for the Offset Time Base and the underlying Synchronized Time Base		
Variation	--		
Parameters	syncTimeBaseStatus	Comment	Status of the Synchronized Time Base
		Type	StbM_TimeBaseStatusType
		Variation	--

		Direction	OUT
	offsetTimeBaseStatus	Comment	Status of the Offset Time Base.
		Type	StbM_TimeBaseStatusType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
GetTimeLeap			
Comments	Returns value of time leap.		
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 32		
Parameters	timeJump	Comment	Time leap value
		Type	StbM_TimeDiffType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	

⌋ (SRS_StbM_20003, SRS_StbM_20010, SRS_StbM_20029, SRS_StbM_20056, SRS_StbM_20057)

8.2.4.3 StartTimer

[SWS_StbM_00409] ⌈

Name	StartTimer	
Comment	Interface, which starts a timer for a Time Base	
IsService	true	
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 128	
Possible Errors	0	E_OK
	1	E_NOT_OK

Operations

StartTimer

Comments	Starts a StbM internal timer, which expires at the given expireTime and which triggers a time notification callback.		
Variation	--		
Parameters	expireTime	Comment	--
		Type	StbM_TimeStampType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	

⌋ (SRS_StbM_20056)

8.2.4.4 TimeNotification

[SWS_StbM_00275] ⌈

Name	TimeNotification	
Comment	Notification, which indicates, that the timer has expired, which has been set by StartTimer	
IsService	true	
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 128	
Possible Errors	0	E_OK
	1	E_NOT_OK

Operations

NotifyTime			
Comments	Notification, which indicates, that the timer has expired, which has been set by StbM_StartTimer		
Variation	--		
Parameters	deviationTime	Comment	--
		Type	StbM_TimeDiffType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	

⌋ (SRS_StbM_20010, SRS_StbM_20056)

8.2.4.5 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
	1	E_NOT_OK

Operations

SetOffsetTimeRecordTable			
Comments	Provides to the recorded snapshot data Block of the table belonging to the Offset Time Base.		
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} > 15 &&{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 32		
Parameters	offsetRecordTableBlock	Comment	Header of the table
		Type	StbM_OffsetRecordTableBlockType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Measurement data access completed successfully	
	E_NOT_OK	Measurement data access failed	
SetSyncTimeRecordTable			
Comments	Provides the recorded snapshot data Block of the table belonging to the Synchronized Time Base.		
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMSynchronizedTimeBaseIdentifier)} < 16		
Parameters	syncRecordTableBlock	Comment	Block of the table
		Type	StbM_SyncRecordTableBlockType

		Variation	--
		Direction	IN
Possible Errors	E_OK	Measurement data access completed successfully	
	E_NOT_OK	Measurement data access failed	

⌋ (SRS_StbM_20057)

8.2.5 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.

These data types are included via the application types header `Rte_StbM_Type.h` into the implementation header `StbM.h`. The implementation header defines additionally those data types, which are listed in chapter 8.1.2, if not included by the application types header.

8.2.5.1 StbM_SynchronizedTimeBaseType

[SWS_StbM_00142] ⌈

Name	StbM_SynchronizedTimeBaseType		
Kind	Type		
Derived from	uint16		
Description	Variables of this type are used to represent the kind of synchronized time-base.		
Range	0..2^16-1		--
Variation	--		

⌋ (SRS_BSW_00305, SRS_StbM_20003, SRS_StbM_20002, SRS_StbM_20010)

8.2.5.2 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	Reserved	0x02	Bit 1: Always 0 (reserved for future usage)
	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 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.			

⌋ (SRS_StbM_20025)

8.2.5.3 StbM_TimeStampType

[SWS_StbM_00241] ⌈

Name	StbM_TimeStampType		
Kind	Structure		
Elements	timeBaseStatus	StbM_TimeBaseStatusType	Status of the Time Base
	nanoseconds	uint32	Nanoseconds part of the time
	seconds	uint32	32 bit LSB of the 48 bits

			Seconds part of the time
	secondsHi	uint16	16 bit MSB of the 48 bits Seconds part of the time
Description	Variables of this type are used for expressing time stamps including relative time and absolute calendar time. The absolute time starts from 1970-01-01. 0 to 281474976710655s == 3257812230d [0xFFFF FFFF FFFF] 0 to 999999999ns [0x3B9A C9FF] invalid value in nanoseconds: [0x3B9A CA00] to [0x3FFF FFFF] Bit 30 and 31 reserved, default: 0		
Variation	--		

⌋ (SRS_StbM_20012)

Note: Start of absolute time (1970-01-01) is according to [17], Annex C/C1 (refer to parameter "approximate epoch" for PTP)

8.2.5.4 StbM_TimeStampExtendedType

[SWS_StbM_00242] ⌈

Name	StbM_TimeStampExtendedType		
Kind	Structure		
Elements	timeBaseStatus	StbM_TimeBaseStatusType	Status of the Time Base
	nanoseconds	uint32	Nanoseconds part of the time
	seconds	uint64	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.		
Variation	--		

⌋ (SRS_StbM_20012)

Note: Start of absolute time (1970-01-01) is according to [17], Annex C/C1 (refer to parameter "approximate epoch" for PTP)

8.2.5.5 StbM_TimeDiffType

[SWS_StbM_00300] ⌈

Name	StbM_TimeDiffType
Kind	Type
Derived	sint32

from			
Description	Variables of this type are used to express time differences / offsets as signed values in in nanoseconds		
Range	-2147483647..2147483647		nanoseconds (-2147483647 .. 2147483647)
Variation	--		

⌋ (SRS_StbM_20056)

8.2.5.6 StbM_RateDeviationType

[SWS_StbM_00301] ⌈

Name	StbM_RateDeviationType		
Kind	Type		
Derived from	sint16		
Description	Variables of this type are used to express a rate deviation in ppm.		
Range	-32000..32000		parts per million (-32000..32000)
Variation	--		

⌋ (SRS_StbM_20056)

8.2.5.7 StbM_UserDataType

[SWS_StbM_00243] ⌈

Name	StbM_UserDataType		
Kind	Structure		
Elements	userDataLength	uint8	User Data Length in bytes
	userByte0	uint8	User Byte 0
	userByte1	uint8	User Byte 1
	userByte2	uint8	User Byte 2
Description	Current user data of the Time Base		
Variation	--		

⌋ (SRS_StbM_20029, SRS_StbM_20030)

8.2.5.8 StbM_CustomerIdType

[SWS_StbM_00288] ⌈

Name	StbM_CustomerIdType		
Kind	Type		
Derived from	uint16		
Description	unique identifier of a notification customer		
Range	0..255		(0x00..0xFF)
Variation	--		

⌋ (SRS_StbM_20010, SRS_StbM_20054, SRS_StbM_20056)

8.2.5.9 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 timeBaseStatus has changed from 0 to 1 0: otherwise
	bit	EV_TIMEOUT_REMOVED	0x04	Bit 2 1: TIMEOUT bit in timeBaseStatus 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 occured 0: No resynchronization event occured
	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.			

⌋ (SRS_StbM_20010, SRS_StbM_20054)

8.2.5.10 StbM_SyncRecordTableHeadType

[SWS_StbM_00331] ⌈

Name	StbM_SyncRecordTableHeadType		
Kind	Structure		
Elements	SynchronizedTimeDomain	uint8	Time Domain 0..15

	HWfrequency	uint32	HW Frequency in Hz
	HWprescaler	uint32	Prescaler value
Description	Synchronized Time Base Record Table Header		
Variation	--		

⌋ (SRS_StbM_20057)

8.2.5.11 StbM_SyncRecordTableBlockType

[SWS_StbM_00332] ⌈

Name	StbM_SyncRecordTableBlockType		
Kind	Structure		
Elements	GlbSeconds	uint32	Seconds of the Local Time Base directly after synchronization with the Global Time Base
	GlbNanoSeconds	uint32	Nanoseconds of the Local Time Base directly after synchronization with the Global Time Base
	TimeBaseStatus	StbM_TimeBaseStatusType	Time Base Status of the Local Time Base directly after synchronization with the Global Time Base
	HWcounter	uint32	HW counter reference value directly after synchronization with the Global Time Base
	RateDeviation	StbM_RateDeviationType	Calculated Rate Deviation directly after rate deviation measurement
	LocSeconds	uint32	Seconds of the Local Time Base directly before synchronization with the Global Time Base
	LocNanoSeconds	uint32	Nanoseconds of the Local Time Base directly before synchronization with the Global Time Base
	PathDelay	uint32	Current propagation delay in nanoseconds
Description	Synchronized Time Base Record Table Block		
Variation	--		

⌋ (SRS_StbM_20057)

8.2.5.12 StbM_OffsetRecordTableHeadType

[SWS_StbM_00333] ⌈

Name	StbM_OffsetRecordTableHeadType		
Kind	Structure		
Elements	OffsetTimeDomain	uint8	Time Domain 16..31
Description	Offset Time Base Record Table Header		
Variation	--		

⌋ (SRS_StbM_20057)

8.2.5.13 StbM_OffsetRecordTableBlockType

[SWS_StbM_00334] ⌈

Name	StbM_OffsetRecordTableBlockType		
Kind	Structure		
Elements	GlbSeconds	uint32	Seconds of the Offset Time Base
	GlbNanoSeconds	uint32	Nanoseconds of the Offset Time Base
	TimeBaseStatus	StbM_TimeBaseStatusType	Time Base Status of the Local Time Base directly after synchronization with the Global Time Base
Description	Offset Time Base Record Table Block		
Variation	--		

⌋ (SRS_StbM_20057)

8.2.5.14 StbM_MasterConfigType

[SWS_StbM_91001] ⌈

Name	StbM_MasterConfigType		
Kind	Type		
Derived from	uint8		
Description	This type indicates if an ECU is configured for a system wide master for a given Time Base is available or not.		
Range	STBM_ SYSTEM_WIDE_MASTER_DISABLED	0x00	not configured as System Wide Master

	STBM_SYSTEM_WIDE_MASTER_ENABLED	0x01	configured as System Wide Master
Variation	--		

⌋ (SRS_StbM_20023)

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_Init

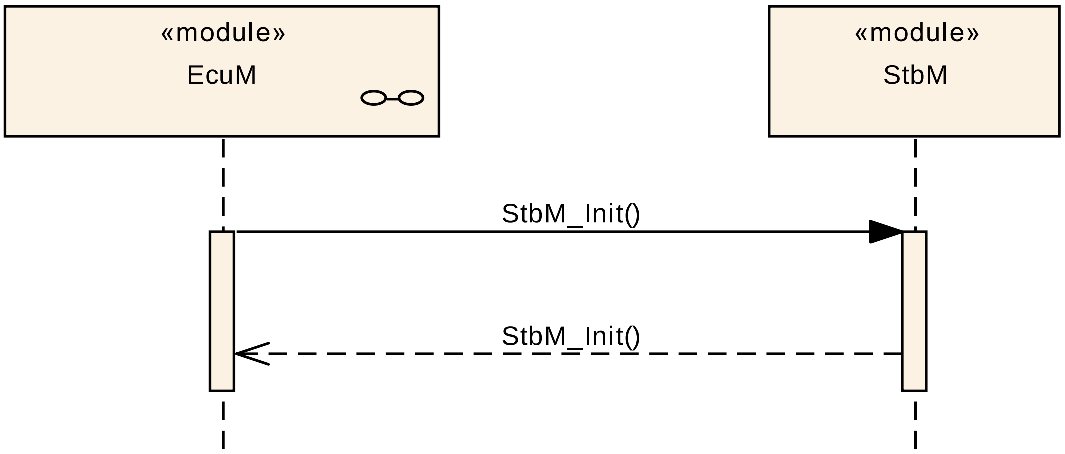


Figure 15: StbM schedule table synchronization sequence

9.2 Immediate Time Synchronisation

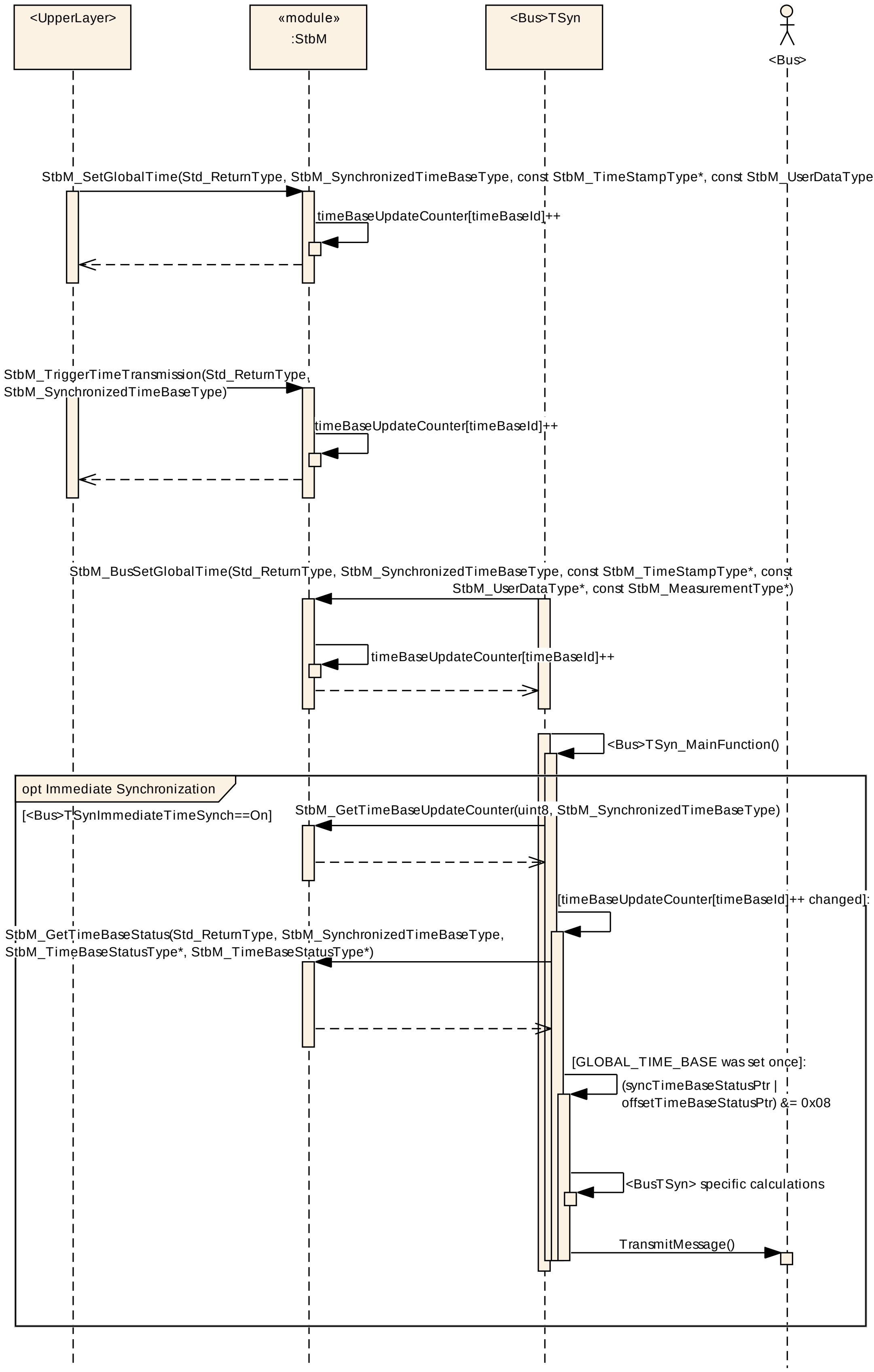

<UpperLayer>
«module»
:StbM
<Bus>TSyn
<Bus>
StbM_SetGlobalTime(Std_ReturnType, StbM_SynchronizedTimeBaseType, const StbM_TimeStampType*, const StbM_UserDataType*)
timeBaseUpdateCounter[timeBaseId]++
StbM_TriggerTimeTransmission(Std_ReturnType, StbM_SynchronizedTimeBaseType)
timeBaseUpdateCounter[timeBaseId]++
StbM_BusSetGlobalTime(Std_ReturnType, StbM_SynchronizedTimeBaseType, const StbM_TimeStampType*, const StbM_UserDataType*, const StbM_MeasurementType*)
timeBaseUpdateCounter[timeBaseId]++
<Bus>TSyn_MainFunction()
opt Immediate Synchronization
[<Bus>TSynImmediateTimeSynch==On]
StbM_GetTimeBaseUpdateCounter(uint8, StbM_SynchronizedTimeBaseType)
[timeBaseUpdateCounter[timeBaseId]++ changed]:
StbM_GetTimeBaseStatus(Std_ReturnType, StbM_SynchronizedTimeBaseType, StbM_TimeBaseStatusType*, StbM_TimeBaseStatusType*)
[GLOBAL_TIME_BASE was set once]:
(syncTimeBaseStatusPtr | offsetTimeBaseStatusPtr) &= 0x08
<BusTSyn> specific calculations
TransmitMessage()

Figure 16: Immediate time synchronization sequence (StbM API)

9.3 Explicit synchronization of OS ScheduleTable

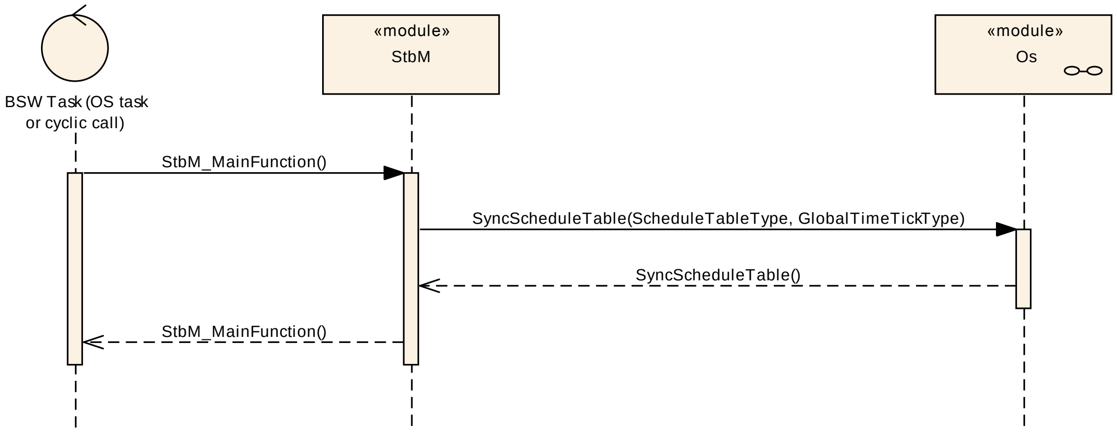

BSW Task (OS task or cyclic call)
«module»
StbM
«module»
Os
StbM_MainFunction()
SyncScheduleTable(ScheduleTableType, GlobalTimeTickType)
SyncScheduleTable()
StbM_MainFunction()

Figure 17: 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 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the Synchronized Time-Base Manager. Chapter 10.3 specifies published information of the module Synchronized Time-Base Manager.

10.1 How to read this chapter

For details refer to the chapter 10.1 "Introduction to configuration specification" in *SWS_BSWGeneral.*

10.2 Containers and configuration parameters

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

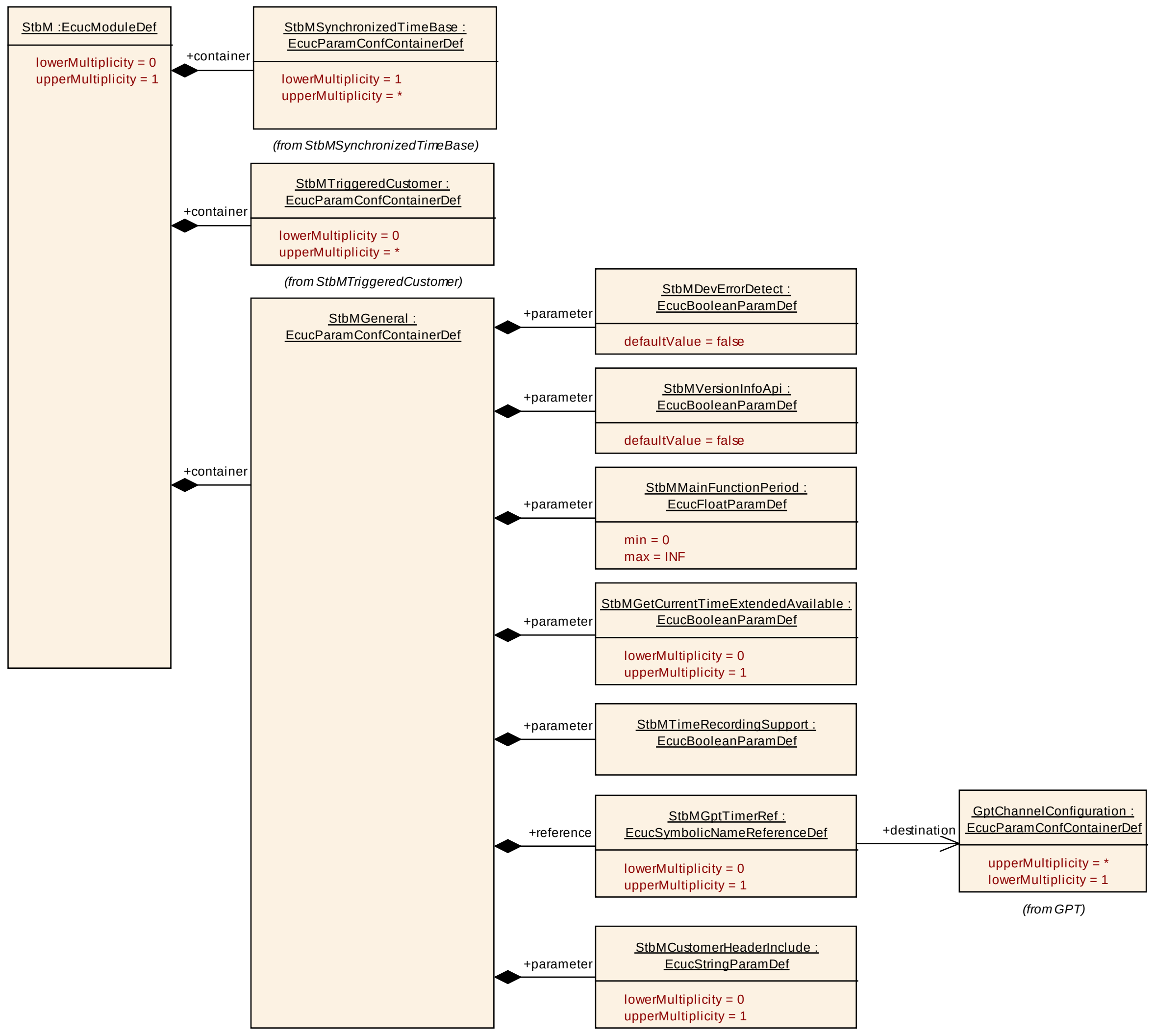
The module supports different post-build variants (previously known as post-build selectable configuration sets), but not post-build loadable configuration.

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*
Module Description	Configuration of the Synchronized Time-base Manager (StbM) module.
Post-Build Variant Support	false
Supported Config Variants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	*Multiplicity*	*Scope / Dependency*
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.


StbM :EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
StbMSynchronizedTimeBase : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
(from StbMSynchronizedTimeBase)
+container
StbMTriggeredCustomer : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
(from StbMTriggeredCustomer)
+container
StbMGeneral : EcucParamConfContainerDef
+parameter
StbMDevErrorDetect : EcucBooleanParamDef
defaultValue = false
+parameter
StbMVersionInfoApi : EcucBooleanParamDef
defaultValue = false
+parameter
StbMMainFunctionPeriod : EcucFloatParamDef
min = 0
max = INF
+parameter
StbMGetCurrentTimeExtendedAvailable : EcucBooleanParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
StbMTimeRecordingSupport : EcucBooleanParamDef
+reference
StbMGptTimerRef : EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
GptChannelConfiguration : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
(from GPT)
+parameter
StbMCustomerHeaderInclude : EcucStringParamDef
lowerMultiplicity = 0
upperMultiplicity = 1

10.2.2 StbMGeneral

SWS Item	**ECUC_StbM_00002 :**
Container Name	StbMGeneral
Description	This container holds the general parameters of the Synchronized Time-base Manager
Configuration Parameters	

SWS Item	**ECUC_StbM_00040 :**
Name	StbMCustomerHeaderInclude
Parent Container	StbMGeneral
Description	Defines the header file, which has the declaration of the the callback function prototype for the notification customer of the reference Time Base.
Multiplicity	0..1
Type	EcucStringParamDef
Default value	--
maxLength	--
minLength	--
regularExpression	--
Post-Build Variant	false

Multiplicity			
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_00012 :**		
Name	StbMDevErrorDetect		
Parent Container	StbMGeneral		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_StbM_00032 :**		
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.		
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 :**	
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 :**		
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_00013 :**		
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_00039 :**		
Name	StbMGptTimerRef		
Parent Container	StbMGeneral		
Description	This represents an optional sub-container in case any Time Notification Customer is configured. The designated GPT timer has to be configured to have a tick duration of one micro second.		
Multiplicity	0..1		
Type	Symbolic name reference to [GptChannelConfiguration]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.3 StbMSynchronizedTimeBase

SWS Item	**ECUC_StbM_00003 :**		
Container Name	StbMSynchronizedTimeBase		
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_00043 :**		
Name	StbMAllowMasterRateCorrection		
Parent Container	StbMSynchronizedTimeBase		
Description	This attribute describes whether the rate correction value of a Time Base can be set by StbM_SetRateCorrection(): • 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_00066 :**
Name	StbMAllowSystemWideGlobalTimeMaster
Parent Container	StbMSynchronizedTimeBase
Description	For postbuild variant of the StbM this parameter has to be set to true for a Global Time Master that may act as a system-wide source of time. Otherwise no corresponding service ports/interfaces is provided. The Global Time Master functionality behind the service ports/interfaces has to be enabled/disabled separately via parameter StbMIsSystemWideGlobalTimeMaster.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	--
Post-Build Variant Multiplicity	false
Post-Build Variant Value	false

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

SWS Item	**ECUC_StbM_00037 :**		
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_00036 :**		
Name	StbMIsSystemWideGlobalTimeMaster		
Parent Container	StbMSynchronizedTimeBase		
Description	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. 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.		
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_00044 :**		
Name	StbMMasterRateDeviationMax		
Parent Container	StbMSynchronizedTimeBase		
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_00046 :**		
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	--		
maxLength	--		
minLength	--		
regularExpression	--		
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_00045 :**		
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 :**		
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_AT_SHUTDOWN	--	
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 :**		
Name	StbMSynchronizedTimeBaseIdentifier		
Parent Container	StbMSynchronizedTimeBase		
Description	Identification of a Synchronized TimeBase via a unique identifier. Range: • 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 :**		
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		

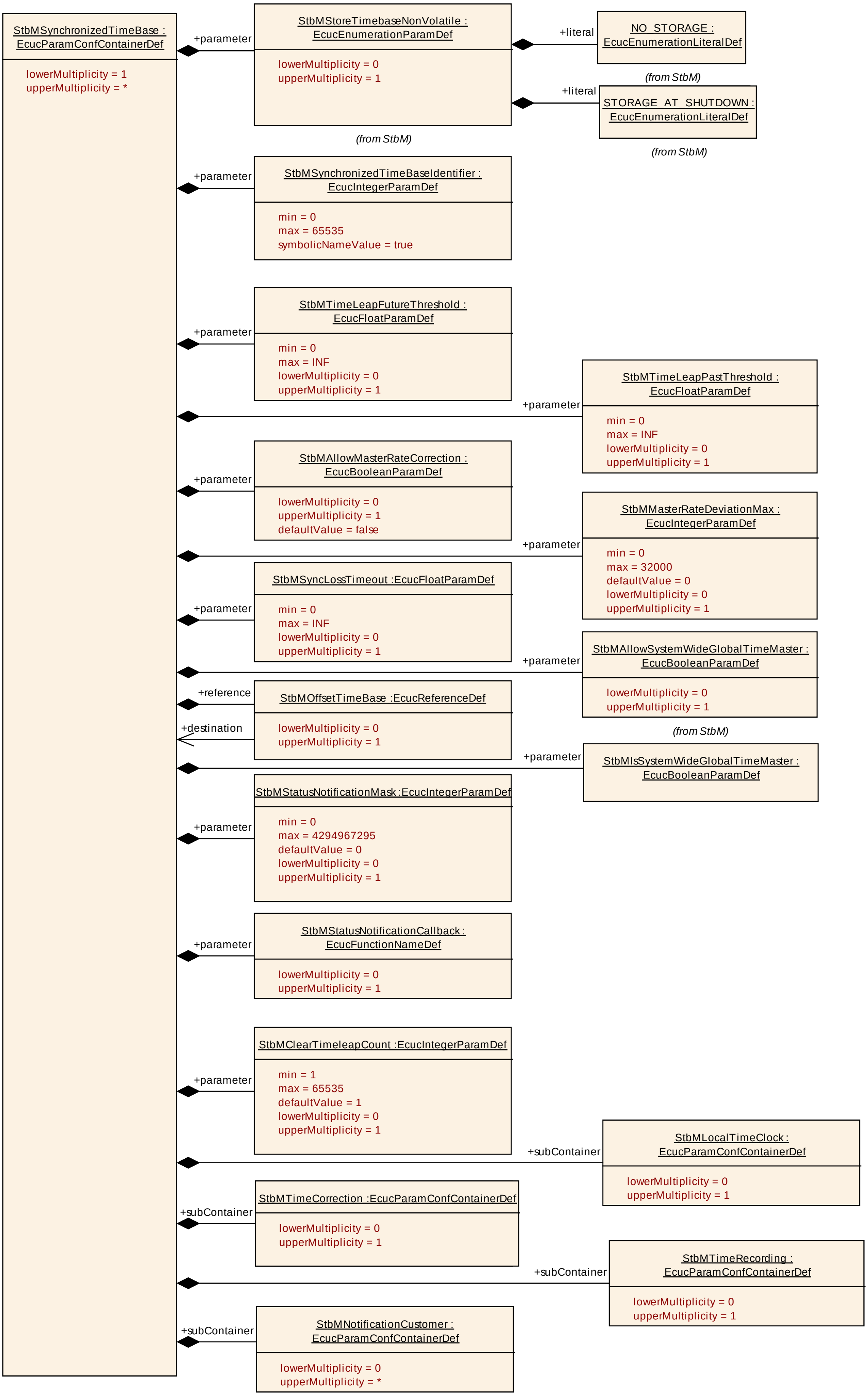
SWS Item	**ECUC_StbM_00041 :**		
Name	StbMTimeLeapFutureThreshold		
Parent Container	StbMSynchronizedTimeBase		
Description	This represents the maximum allowed positive difference between a newly received Global Time Base value and the current Local Time Base value [unit: seconds].		
Multiplicity	0..1		
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 :**		
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 :**
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		

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 occurance 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.


StbMSynchronizedTimeBase : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
StbMStoreTimebaseNonVolatile : EcucEnumerationParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
(from StbM)
+literal
NO_STORAGE : EcucEnumerationLiteralDef
(from StbM)
+literal
STORAGE_AT_SHUTDOWN : EcucEnumerationLiteralDef
(from StbM)
+parameter
StbMSynchronizedTimeBaseIdentifier : EcucIntegerParamDef
min = 0
max = 65535
symbolicNameValue = true
+parameter
StbMTimeLeapFutureThreshold : EcucFloatParamDef
min = 0
max = INF
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
StbMTimeLeapPastThreshold : EcucFloatParamDef
min = 0
max = INF
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
StbMAllowMasterRateCorrection : EcucBooleanParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
defaultValue = false
+parameter
StbMMasterRateDeviationMax : EcucIntegerParamDef
min = 0
max = 32000
defaultValue = 0
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
StbMSyncLossTimeout :EcucFloatParamDef
min = 0
max = INF
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
StbMAllowSystemWideGlobalTimeMaster : EcucBooleanParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
(from StbM)
+reference
StbMOffsetTimeBase :EcucReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
+parameter
StbMIsSystemWideGlobalTimeMaster : EcucBooleanParamDef
StbMStatusNotificationMask :EcucIntegerParamDef
+parameter
min = 0
max = 4294967295
defaultValue = 0
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
StbMStatusNotificationCallback : EcucFunctionNameDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
StbMClearTimeleapCount :EcucIntegerParamDef
min = 1
max = 65535
defaultValue = 1
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
StbMLocalTimeClock : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
StbMTimeCorrection :EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
StbMTimeRecording : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
StbMNotificationCustomer : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *

10.2.4 StbMTimeCorrection

SWS Item	**ECUC_StbM_00048 :**		
Container Name	StbMTimeCorrection		
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_00057 :**		
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		

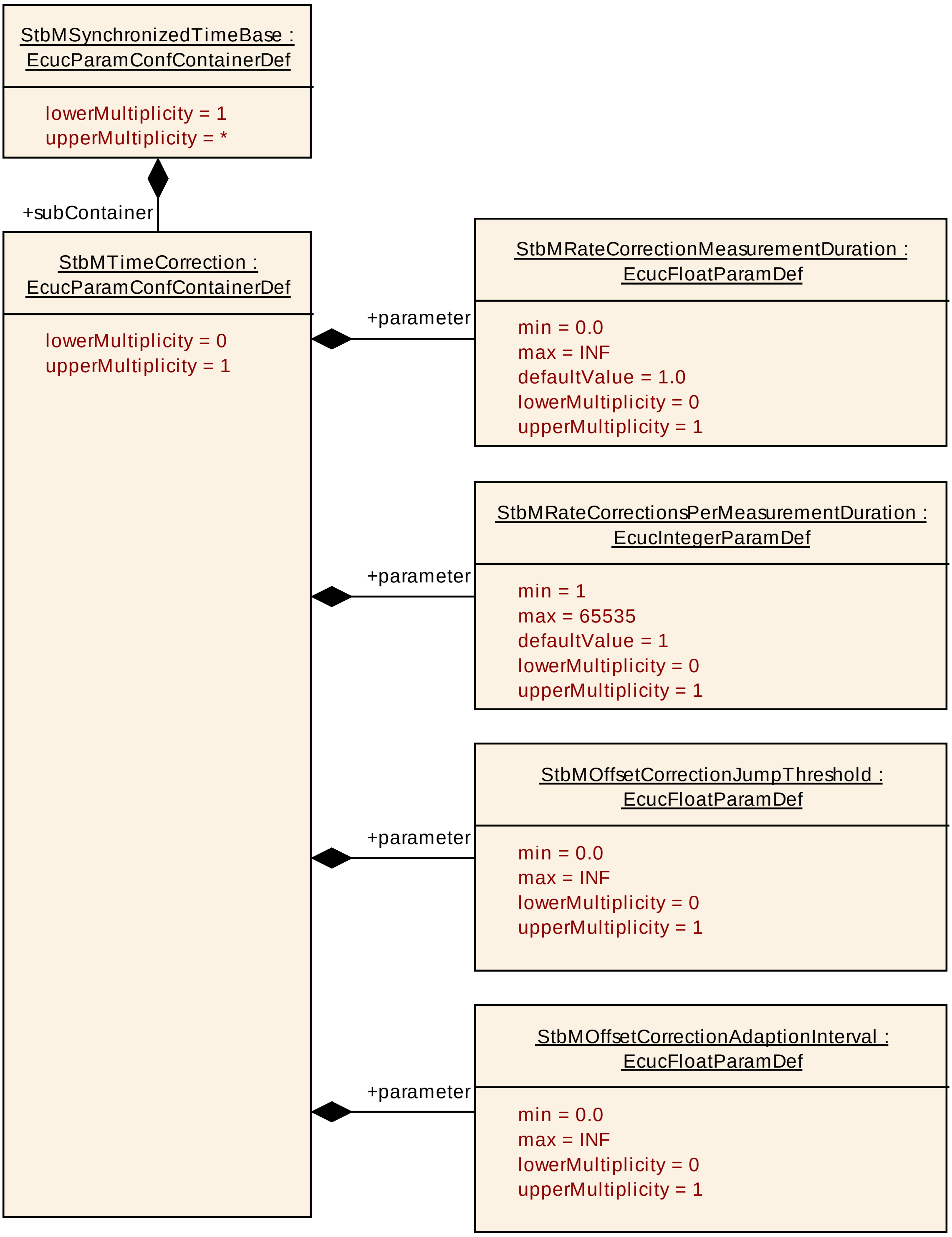
SWS Item	**ECUC_StbM_00056 :**		
Name	StbMOffsetCorrectionJumpThreshold		
Parent Container	StbMTimeCorrection		
Description	Threshold for the correction method. Deviations below this value will be corrected by a linear reduction over a defined timespan. Values equal- and greater than this value will 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 :**		
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 :**		
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



10.2.5 StbMLocalTimeClock

SWS Item	**ECUC_StbM_00047 :**
Container Name	StbMLocalTimeClock
Description	References the hardware reference clock of this Synchronized Time Base.

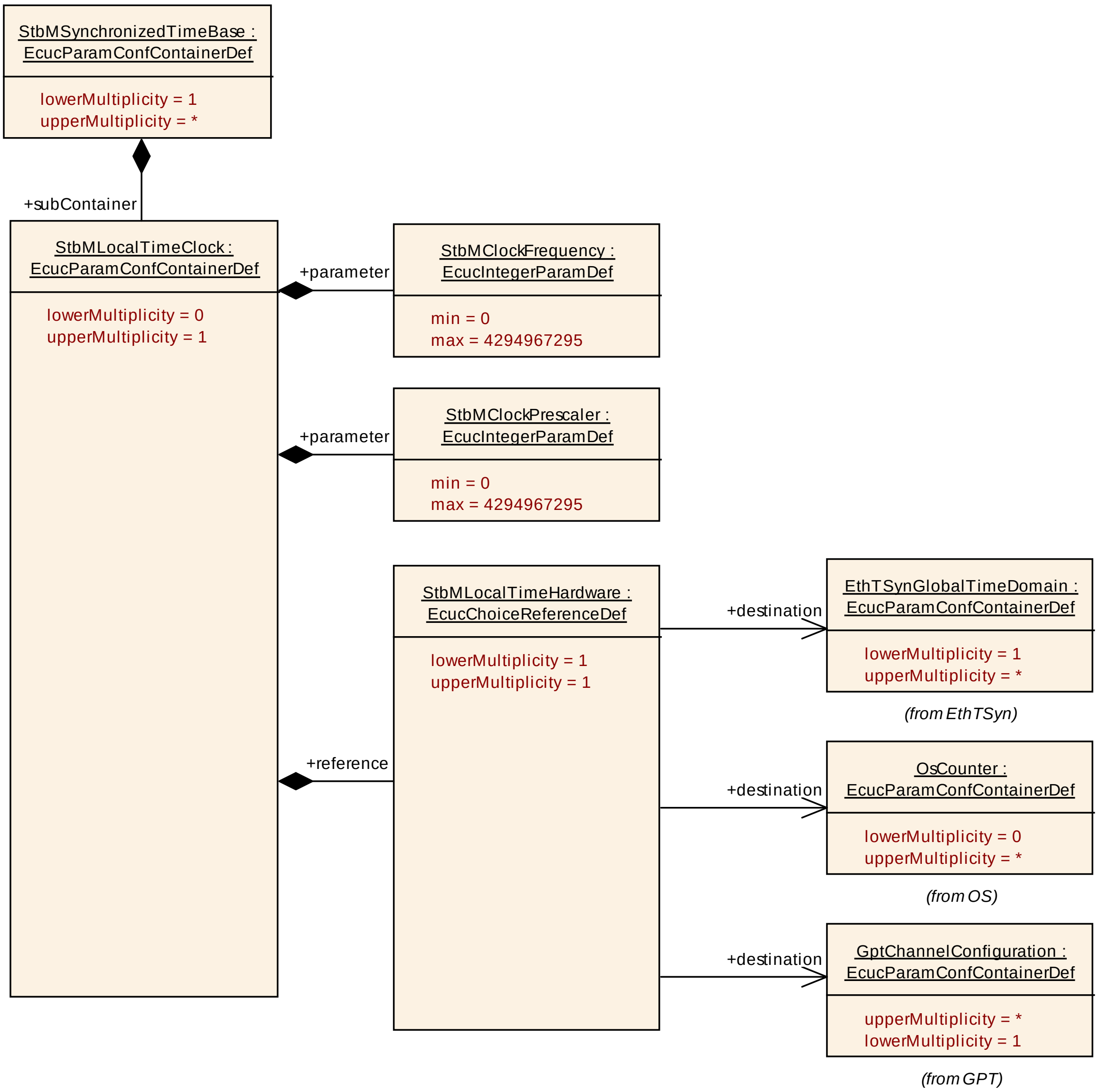
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Configuration Parameters			

SWS Item	**ECUC_StbM_00051 :**		
Name	StbMClockFrequency		
Parent Container	StbMLocalTimeClock		
Description	Represents the frequency [Hz] of the HW reference clock used by the StbM.		
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_00052 :**		
Name	StbMClockPrescaler		
Parent Container	StbMLocalTimeClock		
Description	Represents the prescaler to calculate the resulting frequency of the HW reference clock used by the StbM.		
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_00053 :**		
Name	StbMLocalTimeHardware		
Parent Container	StbMLocalTimeClock		
Description	Reference to the local time hardware.		
Multiplicity	1		
Type	Choice reference to [EthTSynGlobalTimeDomain , GptChannelConfiguration , 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



10.2.6 StbMTimeRecording

SWS Item	**ECUC_StbM_00049 :**		
Container Name	StbMTimeRecording		
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 :**		
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	--		
maxLength	--		
minLength	--		
regularExpression	--		
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		

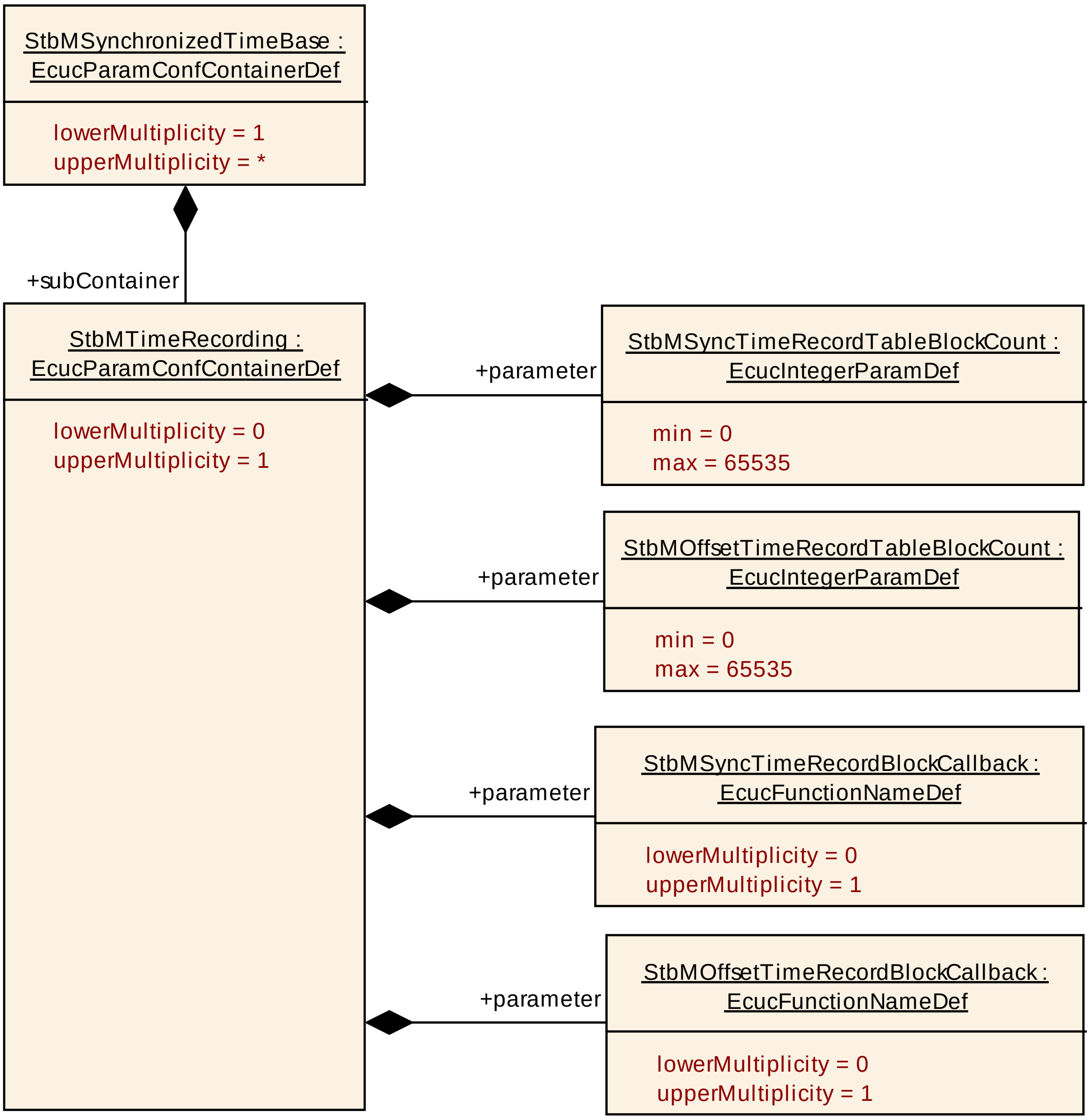
SWS Item	**ECUC_StbM_00059 :**		
Name	StbMOffsetTimeRecordTableBlockCount		
Parent Container	StbMTimeRecording		
Description	Represents the number of Blocks used for queing time measurement events for the Offset Time Base Record Table.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
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 :**		
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	--		
maxLength	--		
minLength	--		
regularExpression	--		
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 :**		
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



10.2.7 StbMNotificationCustomer

SWS Item	**ECUC_StbM_00050 :**		
Container Name	StbMNotificationCustomer		
Description	This container holds the configuration of a notification customer, which is notified is informed about the occurance 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			

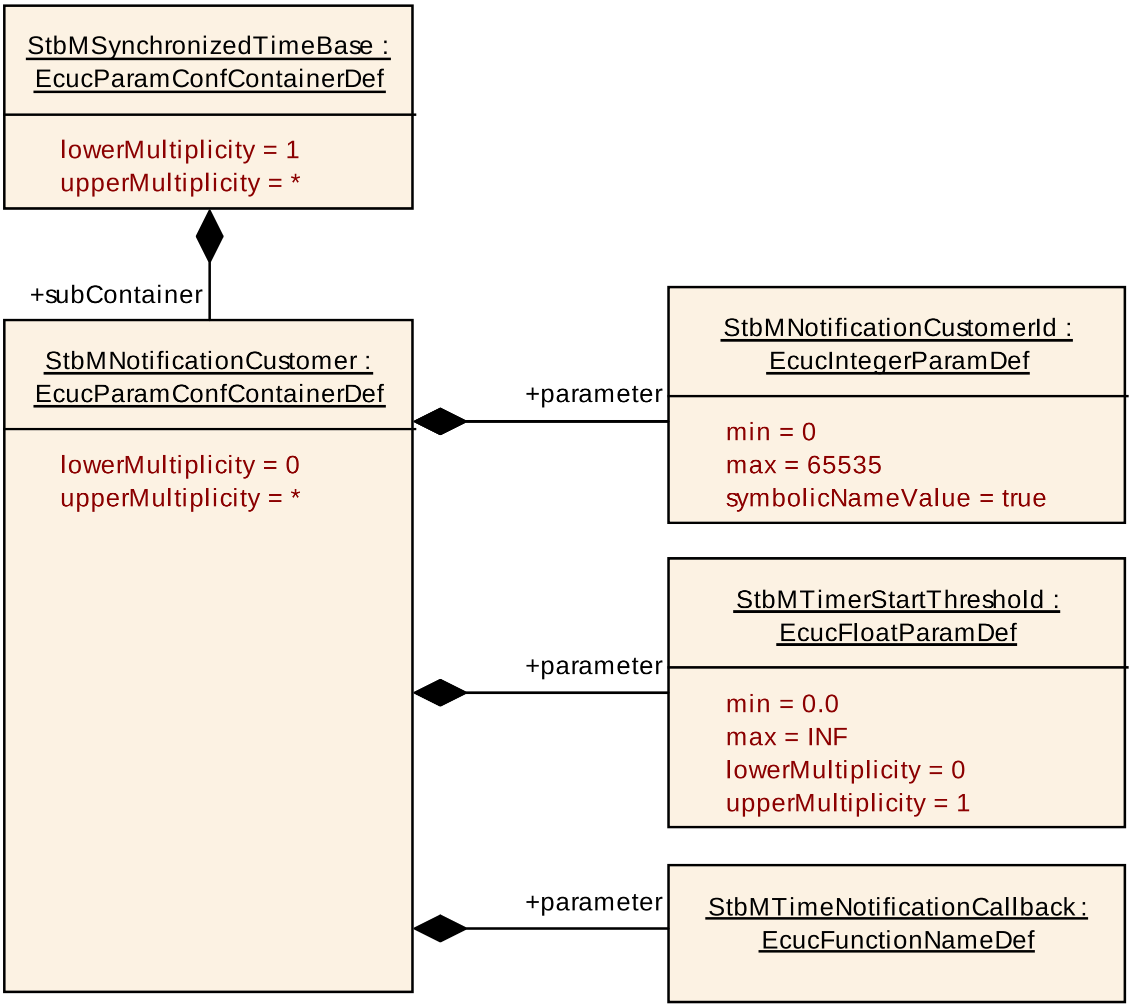
SWS Item	**ECUC_StbM_00062 :**
Name	StbMNotificationCustomerId
Parent Container	StbMNotificationCustomer

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 :**		
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	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_StbM_00063 :**		
Name	StbMTimerStartThreshold		
Parent Container	StbMNotificationCustomer		
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		

No Included Containers


StbMSynchronizedTimeBase : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+subContainer
StbMNotificationCustomer : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
+parameter
StbMNotificationCustomerId : EcucIntegerParamDef
min = 0
max = 65535
symbolicNameValue = true
+parameter
StbMTimerStartThreshold : EcucFloatParamDef
min = 0.0
max = INF
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
StbMTimeNotificationCallback : EcucFunctionNameDef

10.2.8 StbMTriggeredCustomer

SWS Item	**ECUC_StbM_00004 :**		
Container Name	StbMTriggeredCustomer		
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			

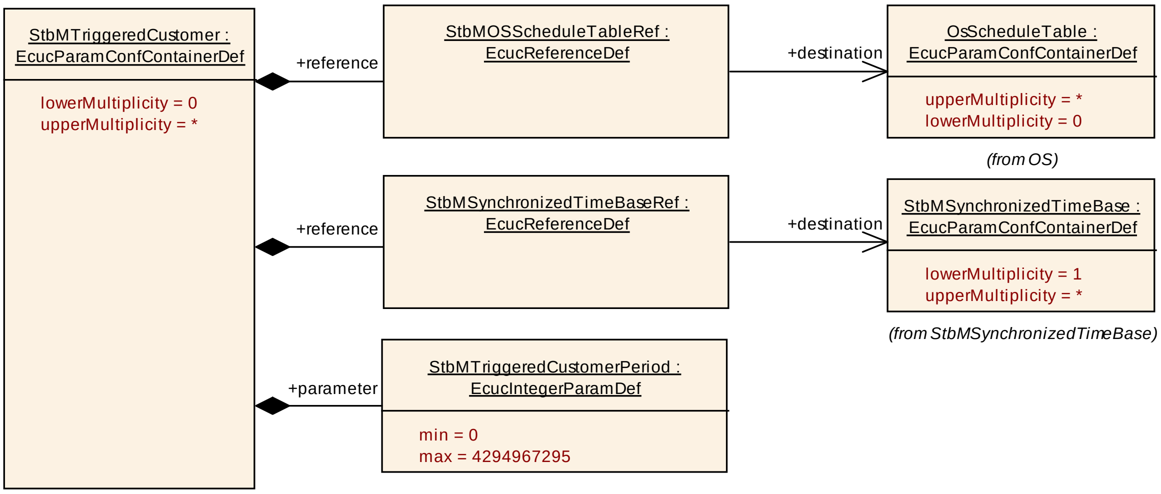
SWS Item	**ECUC_StbM_00020 :**
Name	StbMTriggeredCustomerPeriod
Parent Container	StbMTriggeredCustomer
Description	The triggering period of the triggered customer, called by the StbM_MainFunction. The period is documented in microseconds.
Multiplicity	1

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 :**		
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 :**		
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



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`.

10.4 Published Information

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

11 Not applicable requirements

[SWS_StbM_00140] ⌈These requirements are not applicable to this specification.⌋

(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_00361, SRS_BSW_00371, SRS_BSW_00375, SRS_BSW_00378, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00412, 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)