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

Document Status	Final
Part of AUTOSAR Release	4.2.2

Document Change History

Release	Changed by	Change Description
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
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.
4.1.3	AUTOSAR Release Management	• Clarification on Autonomous Time Maintenance
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

Document Change History

Release	Changed by	Change Description
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
4.0.3	AUTOSAR Administration	• Added functionality for absolute time provision
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
3.1.4	AUTOSAR Administration	• Initial Release

Disclaimer

This specification 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 specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification 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 specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction and Functional Overview 6
1.1 Use Cases....6
1.2 Functional Overview....6
2 Acronyms, Abbreviations, and Definitions 9
2.1 Acronyms and Abbreviations....9
2.2 Definitions 9
2.2.1 Clock....9
2.2.2 Global Time Master....9
2.2.3 Synchronized Time Base 10
2.2.4 Time Base....10
2.2.5 Time Base Provider....11
2.2.6 Time Communication Port....11
2.2.7 Time Communication Service 11
2.2.8 Time Domain....12
2.2.9 Time Gateway....12
2.2.10 Time Hierarchy....12
2.2.11 Time Master 12
2.2.12 Time Slave 12
2.2.13 Time Sub-domain....12
2.2.14 TimeSync ECU....13
2.2.15 TimeSync Module – <Bus>TSyn....13
3 Related documentation....14
3.1 Input documents....14
3.2 Company Reports, Academic Work, etc....15
3.3 Related specification 15
4 Constraints and assumptions 16
4.1 Limitations....16
4.1.1 OS ScheduleTable 16
4.1.2 Mode switches 16
4.1.3 Out of scope....16
4.2 Applicability to car domains....16
4.3 Conflicts 16
5 Dependencies to other modules....17
5.1 Code file structure 17
5.2 Header file structure....17
6 Requirements traceability....19
7 Functional specification 26
7.1 Startup behavior....26
7.1.1 Preconditions 26
7.1.2 Initialization 26
7.2 Shutdown behavior....26
7.3 Normal operation....27

7.3.1 Introduction 27
7.3.2 Synchronized Time Bases 29
7.3.3 Offset Time Bases....32
7.3.4 Time Gateway 32
7.3.5 Customers....32
7.4 Error Handling 34
7.4.1 Error classification....34
7.4.2 Error detection....35
7.4.3 Error notification 35
7.4.4 Production Errors 35
7.4.5 Extended Production Errors 35
7.5 Version checking 35
7.6 AUTOSAR Service Interface 36
7.6.1 Architecture 36
7.6.2 Requirements....36
7.6.3 Use Cases....37
7.6.4 Specification of the Ports and Port Interfaces 37
7.6.5 Definition of the Service 38
7.6.6 Configuration of the Service 39
8 API specification....40
8.1 API 40
8.1.1 Imported types 40
8.1.2 Type definitions 41
8.1.3 Function definitions 42
8.1.4 Scheduled functions....49
8.1.5 Expected Interfaces 50
8.2 Service Interfaces....51
8.2.1 Client-Server-Interfaces 51
8.2.2 Implementation Data Types 55
8.2.3 Ports....58
9 Sequence diagrams 59
9.1 StbM_Init 59
9.2 Explicit synchronization of OS ScheduleTable 59
10 Configuration specification....60
10.1 How to read this chapter 60
10.2 Containers and configuration parameters 60
10.2.1 Variants....60
10.2.2 StbM....60
10.2.3 StbMGeneral 61
10.2.4 StbMSynchronizedTimeBase....62
10.2.5 StbMTriggeredCustomer 66
10.3 Published Information....68
11 Not applicable requirements 69

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

2 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.

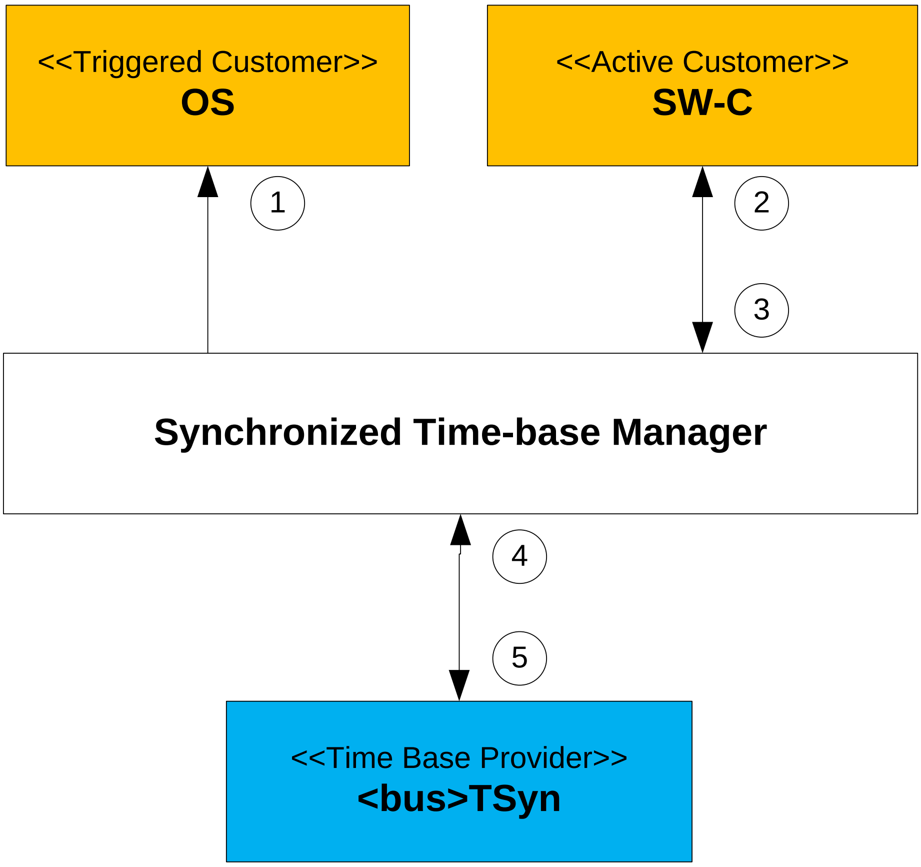


Figure 1: Synchronized Time-Base Manager as broker

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. 2 types of customers may be distingushed:

a) **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).

b) **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 timebase maintained by the Synchronized Time-Base Manager according to application information.

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.

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
DEM	Diagnostic Event Manager
DET	Development 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	Time adjustment message (Follow-Up)
GM(C)	Grand Master (Clock)
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
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 micro controller.

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.

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 2 shows a network topology example and the related terminology.

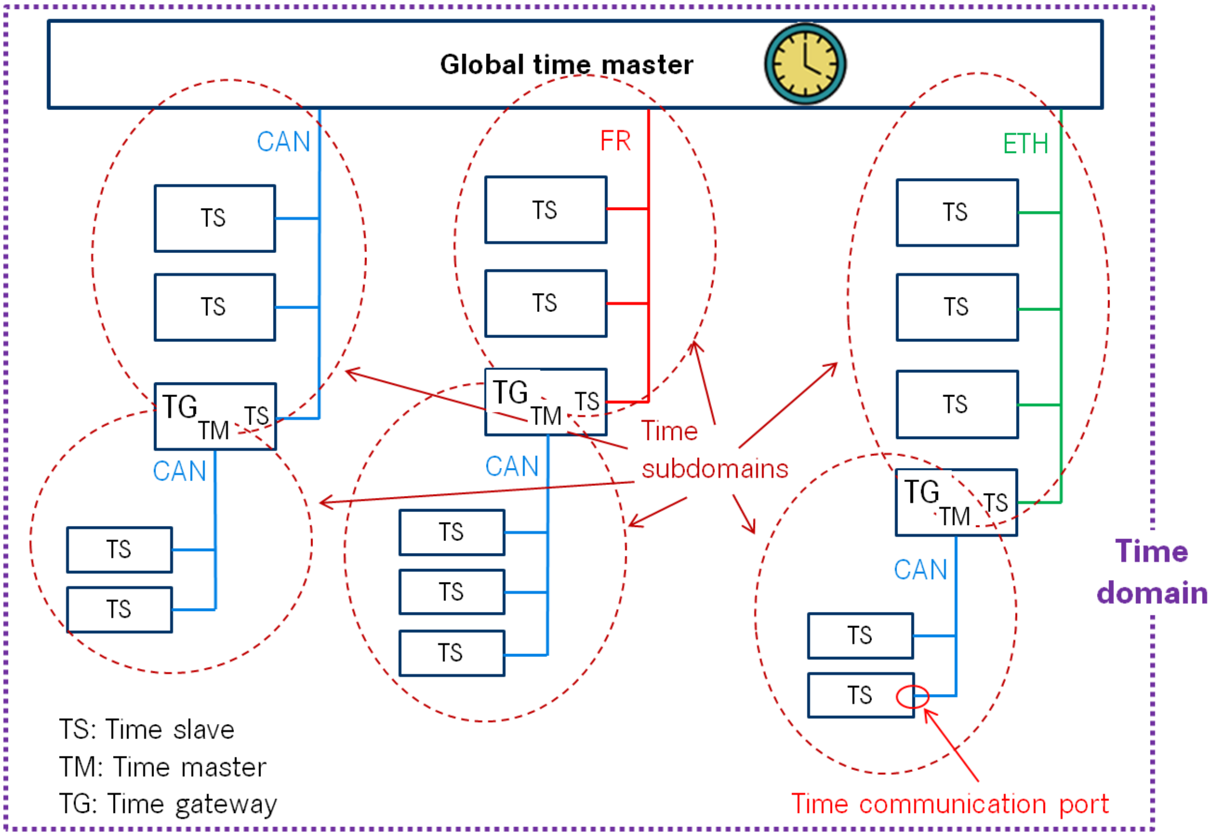


Figure 2: Terminology Example

2.2.8 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.

2.2.9 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 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).

2.2.10 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 gateways (if present) to various Time Slaves.

2.2.11 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.

2.2.12 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.

2.2.13 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.

2.2.14 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.

2.2.15 TimeSync Module – <Bus>TSyn

Definition: TimeSync 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 busses of the same type.

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 TTCAN Interface
AUTOSAR_SWS_TTCANInterface.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

[11] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping.pdf

[12] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.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

3.2 Company Reports, Academic Work, etc.

[16] Real-Time Systems and Software
Publisher: John Wiley & Sons Inc Publication
Date: 2001; Author: Alan C. Shaw

[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 Synchronized Time-Base Manager.

Thus, the specification SWS BSW General shall be considered as additional and required specification for 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 Mode switches

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

4.1.3 Out of scope

- Responsibility for those occurred errors during global time establishment, 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 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 and EthTSyn.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 synchronization via Ethernet shall be supported (refer to **ECUC_StbM_00033 :** *StbMEthGlobalTimeDomainRef*), StbM.c shall include EthTSyn.h to have access to the interface of the EthTSyn module.⌋ (SRS_BSW_00384)

Figure 3 shows the header file structure of the Synchronized Time-base Manager.

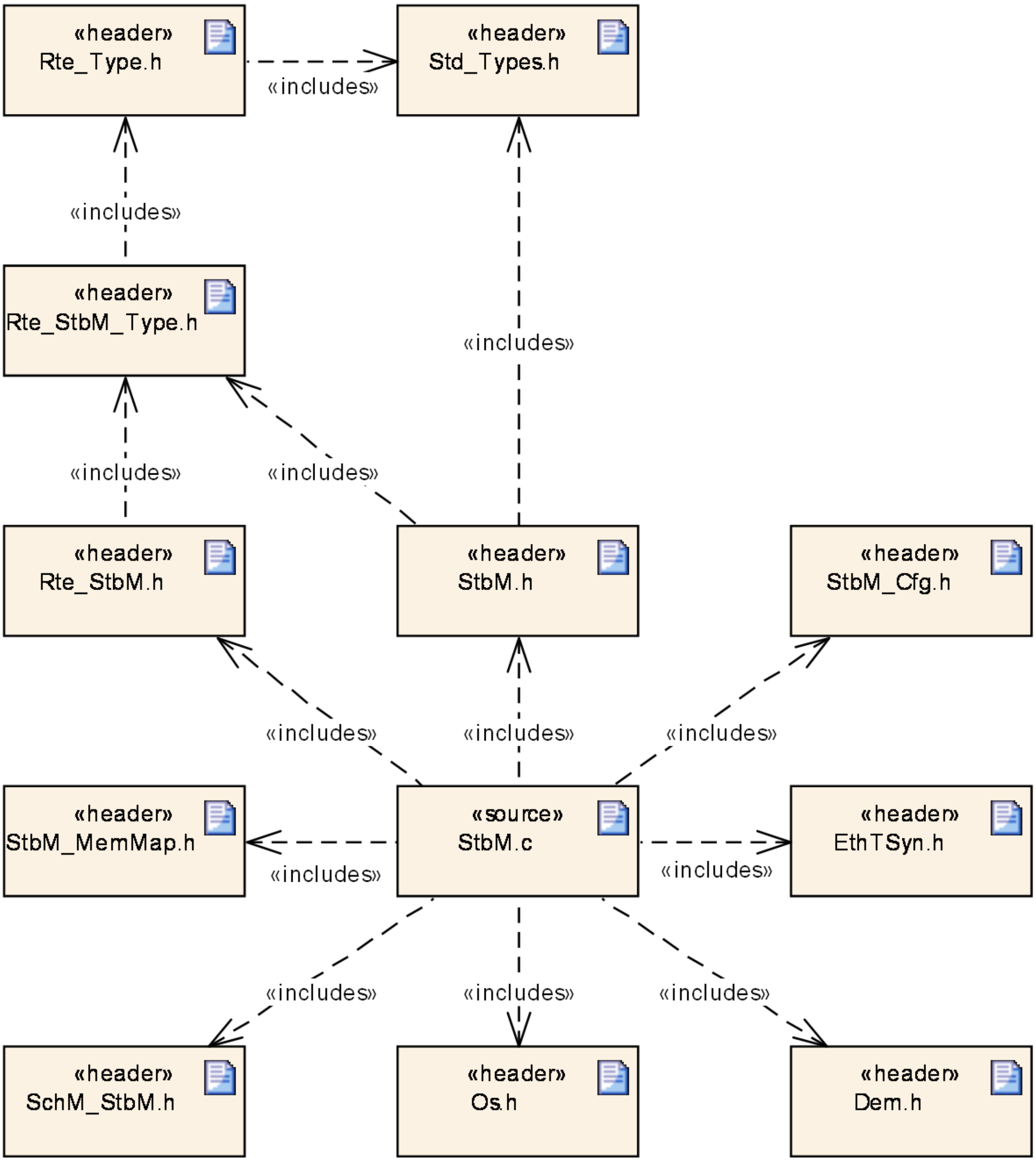


Figure 3: Header File Structure

6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00005	Modules of the ÂµC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_StbM_00140
SRS_BSW_00006	The source code of software modules above the ÂµC 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 2004 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 Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_StbM_00140
SRS_BSW_00172	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	SWS_StbM_00057
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, SWS_StbM_00150
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_StbM_00124
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
SRS_BSW_00325	The runtime of interrupt	SWS_StbM_00140

	service routines and functions that are running in interrupt context shall be kept short	
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
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_00099, 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_00345	BSW Modules shall support pre-compile configuration	SWS_StbM_00245
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_00361	All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler	SWS_StbM_00140

	specific type and keyword header	
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_00099, 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
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	SWS_StbM_00100, SWS_StbM_00121

	with value 0 before any APIs of the BSW module is called	
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, SWS_StbM_00250
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 defined and documented in the BSW module description template	SWS_StbM_00140
SRS_BSW_00428	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	SWS_StbM_00140
SRS_BSW_00429	BSW modules shall be only allowed to use OS objects and/or related OS services	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_00442	{OBSOLETE} The AUTOSAR architecture shall support standardized debugging and tracing features	SWS_StbM_00076
SRS_BSW_00453	BSW Modules shall be harmonized	SWS_StbM_00140
SRS_StbM_20001	The StbM configuration shall allow the interaction with different types of customers	SWS_StbM_00020, SWS_StbM_00022, SWS_StbM_00093
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_00150
SRS_StbM_20003	The StbM shall allow customers to have access to the synchronized time base	SWS_StbM_00142, SWS_StbM_00150, SWS_StbM_00173, SWS_StbM_00195, SWS_StbM_00200, SWS_StbM_00240, SWS_StbM_00244, SWS_StbM_00247, SWS_StbM_00248
SRS_StbM_20007	The StbM shall provide fault detection mechanisms	SWS_StbM_00031, SWS_StbM_00183, SWS_StbM_00187, SWS_StbM_00199
SRS_StbM_20010	The StbM shall provide a system service interface to applications	SWS_StbM_00131, SWS_StbM_00150, SWS_StbM_00240, SWS_StbM_00244, SWS_StbM_00247, SWS_StbM_00248
SRS_StbM_20012	The StbM shall provide a bus independent customer interface	SWS_StbM_00241, SWS_StbM_00242
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

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
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 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_20024	The StbM configuration shall allow the StbM to support different types of time base providers	SWS_StbM_00178, SWS_StbM_00180
SRS_StbM_20025	The StbM shall maintain the synchronization status of a synchronized time base	SWS_StbM_00176, SWS_StbM_00179, SWS_StbM_00181, SWS_StbM_00182, SWS_StbM_00183, SWS_StbM_00184, SWS_StbM_00185, SWS_StbM_00186, SWS_StbM_00187, SWS_StbM_00188, SWS_StbM_00189, SWS_StbM_00194, SWS_StbM_00239
SRS_StbM_20026	The StbM shall allow customer on master side to set the local time	SWS_StbM_00213, SWS_StbM_00240, SWS_StbM_00244
SRS_StbM_20027	The StbM shall allow time base providers to read the offset value of a time base	SWS_StbM_00191, SWS_StbM_00193, SWS_StbM_00228
SRS_StbM_20028	The StbM shall allow customers and time base providers to set the offset value of a 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
SRS_StbM_20029	The StbM shall allow customers to read User Data propagated via the time synchronization protocol	SWS_StbM_00173, 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_00218, SWS_StbM_00240, SWS_StbM_00243, SWS_StbM_00244

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 synchronized time base to the *customers*.

7.1.1 Preconditions

Note:
The C initialization code (also known as start-up code) which initializes global and static variables with the initial values shall be executed before any call of a module function.

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 `StbMSynchronizedTimeBase` (**ECUC_StbM_00003 :**)) shall be initialized with zero and its synchronization status `timeBaseStatus` shall be set set for all Time Domains and all bits to `0x00`.⌋ (SRS_StbM_20018)

[SWS_StbM_00171]⌈
For each Time Base configured to be stored non-volatile (`StbMStoreTimebaseNonVolatile == STORAGE_AT_SHUTDOWN`), the 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.

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 latest received valid Global Time Base and runs autonomously until the next Global Time Base is received.

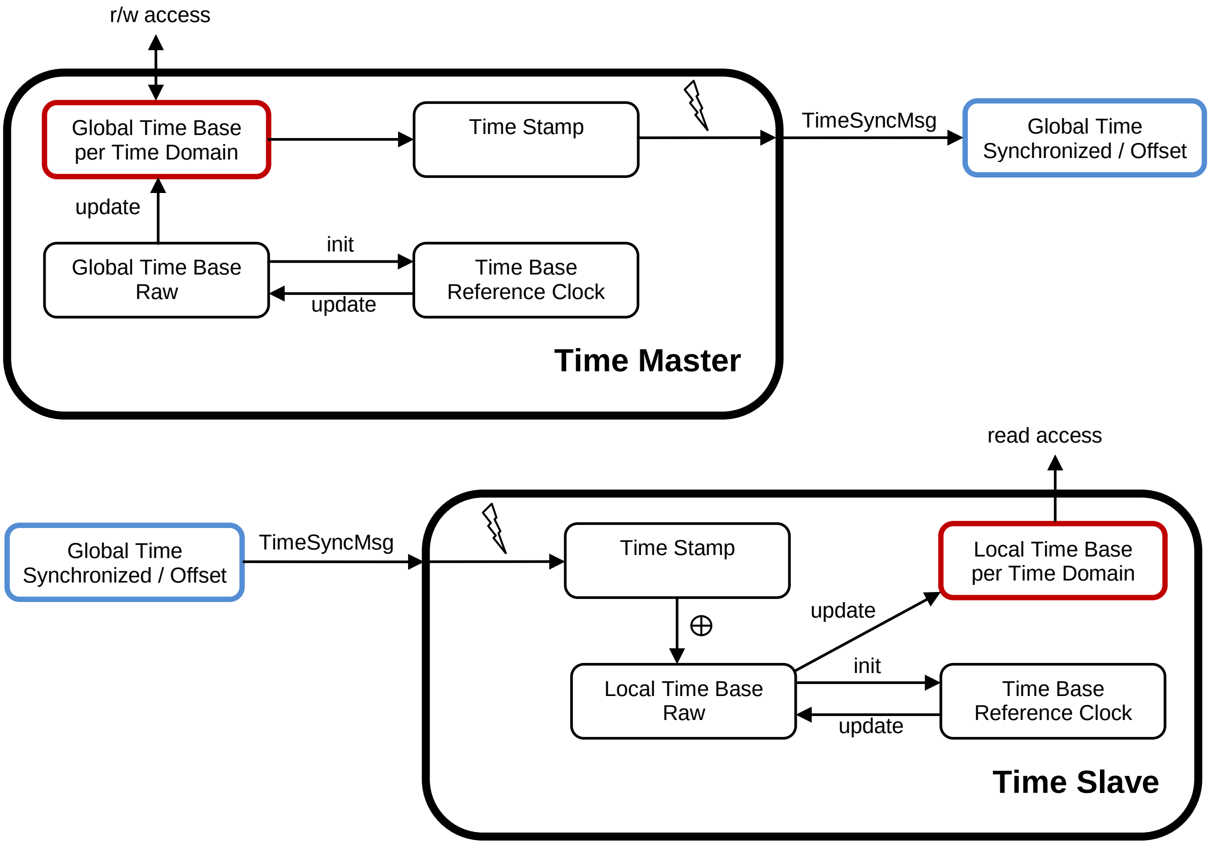

r/w access
Global Time Base per Time Domain
Time Stamp
TimeSyncMsg
Global Time Synchronized / Offset
update
init
Global Time Base Raw
update
Time Base Reference Clock
Time Master
read access
Global Time Synchronized / Offset
TimeSyncMsg
Time Stamp
Local Time Base per Time Domain
update
init
Local Time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

Figure 4: Global Time Base Distribution

Synchronized Time Base
The Time Domains 0 till 15 are Synchronized Time Bases.

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

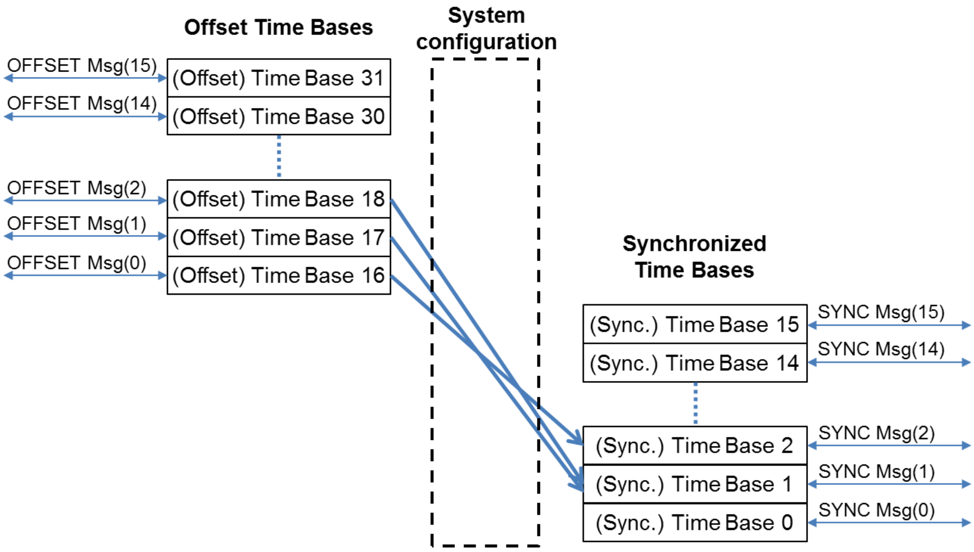


Figure 5: Offset Time Base to Synchronized Time Base relationship

Example:
For an Offset Time Base with Time Domain number 17 the OFFSET TimeSync messages always carries the Time Domain number 17-16 = 1. However the underlying Synchronized Time Base could have Time Domain number 0, i.e., SYNC and FUP TimeSync messages contain Time Domain number 0.
Another Offset Time Base with Time Domain number 18 (Time Domain number 2), is also based on the underlying Synchronized Time Base 0.
An Offset Time Bases might have leaps in time, e.g. after GPS time stamp is available.

[SWS_StbM_00173]⌈
For Time Domains 0 till 15 the `StbM_GetCurrentTime()` shall return for the requested Time Domain the current Time Base, the related Status and the User Data. The current Time Base shall be derived from the either the referenced OS counter (refer `StbMLocalTimeRef`) or the referenced Ethernet controller (refer `StbMEthGlobalTimeDomainRef`) if `EthTSynHardwareTimestamp` is set to TRUE⌋ (SRS_StbM_20003, SRS_StbM_20029)

[SWS_StbM_00076] {OBSOLETE}

⌈The status of a timebase, which is maintained by the Synchronized Time-Base Manager and defined in section 8.2.2.2, shall be available for debugging. ⌋ (SRS_BSW_00442)

[SWS_StbM_00174][⌈
`StbM_GetCurrentTimeRaw()` shall return the nanoseconds portion from the most accurate OS counter (refer `StbMLocalTimeRef`).⌋ (SRS_StbM_20016, SRS_StbM_20021)

[SWS_StbM_00175]⌈

StbM_GetCurrentTimeDiff() shall return the time difference from the nanoseconds portion from the most accurate OS counter (refer StbMLocalTimeRef) minus the given time in raw format.⌋ (SRS_StbM_20016, SRS_StbM_20021)

[SWS_StbM_00176]⌈
In case of EthTSyn_GetCurrentTime() the timeBaseStatus shall be derived as follows from the syncState:
ETHTSYN_SYNC → The GLOBAL_TIME_BASE bit is set
ETHTSYN_UNSYNC → If the GLOBAL_TIME_BASE bit is set the TIMEOUT is set
ETHTSYN_UNCERTAIN → This does not change the current status as long as the ETHTSYN_UNCERTAIN condition does not last for longer as the timeout on receiving valid Synchronisation Messages
ETHTSYN_NEVERSYNC → 0x00 (no bit is set)

Further the return value of StbM_GetCurrentTime() shall be E_NOT_OK if syncState is set to ETHTSYN_UNCERTAIN, otherwise the return value shall be E_OK.⌋ (SRS_StbM_20025)

Note: EthTSyn will never return the state SYNC_TO_GATEWAY due to the fact, that the PTP protocol defines only one Time Domain, which belongs to the Global Time Domain.
[SWS_StbM_00177]⌈
For Time Domains 16 till 31 the StbM_GetCurrentTime() shall return for the requested Time Domain a Time Base calculated by adding the given offset (via StbM_SetOffset()) to the current Time Base of the referenced Time Domain via StbMOffsetTimeBase (**ECUC_StbM_00030 :**).
The timeBaseStatus of the referenced Time Domain shall be returned, except no Offset value is available. In this case (offset value is not available), the returned timestamp shall be of the referenced Time Domain and the timeBaseStatus shall be 0x00.⌋ (SRS_StbM_20028)

[SWS_StbM_00178]⌈
In case the Time Base is maintained by an OS counter (refer StbMLocalTimeRef), the StbM shall retrieve the time value from there, otherwise by the referenced EthTSyn module via EthTSyn_GetCurrentTime() (refer StbMEthGlobalTimeDomainRef), if EthTSynHardwareTimestamp is set to TRUE.⌋ (SRS_StbM_20016, SRS_StbM_20024)

7.3.2 Synchronized Time Bases

[SWS_StbM_00179]⌈
Each invocation of StbM_BusSetGlobalTime() shall update the corresponding Synchronized Time Base and set the Time Base Status accordingly.⌋ (SRS_StbM_20014, SRS_StbM_20025)

[SWS_StbM_00180]⌈
The StbM shall maintain the Local Time Base autonomously either via StbMLocalTimeRef or via StbMEthGlobalTimeDomainRef for each Time Domain after initialization.⌋ (SRS_StbM_20016, SRS_StbM_20024)

7.3.2.1 Synchronization State within Global Time Master

[SWS_StbM_00181]⌈
On a valid invocation of StbM_SetGlobalTime() the StbM shall set the GLOBAL_TIME_BASE bit within timeBaseStatus of the corresponding Time Domain and shall clear all other bits.⌋ (SRS_StbM_20025)

7.3.2.2 Synchronization State Supervision within Time Slaves

[SWS_StbM_00182]⌈
For each Time Domain where a Time Slave belongs to, an invocation of StbM_BusSetGlobalTime() shall check if the time difference between the current and the update timestamp exceeds the configured threshold of StbMSyncLossThreshold (**ECUC_StbM_00029 :**).
In case the threshold is exceeded the StbM shall set the TIMELEAP bit within timeBaseStatus of the Time Base.

A threshold of 0 shall deactivate this check.

If the next update is within the threshold the StbM shall clear the TIMELEAP bit within timeBaseStatus of the 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 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 bit.⌋ (SRS_StbM_20007, SRS_StbM_20025)

[SWS_StbM_00184]⌈
Every invocation of StbM_BusSetGlobalTime() shall clear the SYNC_TO_GATEWAY bit within timeBaseStatus of the Time Base if the parameter syncToTimeBase is set to SYNC to GTM (0) and shall set the SYNC_TO_GATEWAY bit if the parameter syncToTimeBase is set to SYNC to sub-domain (1).⌋ (SRS_StbM_20025)

[SWS_StbM_00185]⌈

Every 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)

7.3.2.3 Synchronization State Supervision within Time Gateways

[SWS_StbM_00186]⌈
For each Time Domain where a Time Gateway Slave Port belongs to, an invocation of StbM_BusSetGlobalTime() shall check if the time difference between the current and the update timestamp exceeds the configured threshold of StbMSyncLossThreshold (**ECUC_StbM_00029 :**).
In case the threshold is exceeded the StbM shall set the TIMELEAP bit within timeBaseStatus of the Time Base.

A threshold of 0 shall deactivate this check.

If the next update is within the threshold the StbM shall clear the TIMELEAP bit within timeBaseStatus of the Time Base.⌋ (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 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_00188]⌈
Every invocation of StbM_BusSetGlobalTime() shall clear the SYNC_TO_GATEWAY bit within timeBaseStatus of the Time Base if the parameter syncToTimeBase is set to SYNC to GTM (0) and shall set the SYNC_TO_GATEWAY bit if the parameter syncToTimeBase is set to SYNC to sub-domain (1).⌋ (SRS_StbM_20025)

[SWS_StbM_00189]⌈
Every 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)

7.3.3 Offset Time Bases

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

[SWS_StbM_00191]⌈
`StbM_SetOffset()` and `StbM_GetOffset()` shall only accept Offset Time Bases with a timeBaseId 16 till 31.⌋ (SRS_StbM_20027, SRS_StbM_20028)

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

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

7.3.4 Time Gateway

A Time Gateway in the StbM is a Time Base which is referenced by a Time Master and a Time Slave. The Time Master retrieves therefore automatically a synchronized time.

7.3.5 Customers

7.3.5.1 Active customers

The Synchronized Time-Base Manager provides to the *active customer* an interface to allow access to the synchronized time base.

The *active customer* calls the API services of the StbM and specifies the required synchronized time base it is interested in.

The Synchronized Time-Base Manager API allows

- to read:
  - The *time base value*.
  - The *sync state* and
  - The *User data*
- To set:
  - the *time base value*
  - the *User data*

of the information for a given time base.

The API for accessing the synchronized time bases is provided to application software components (see [7]) as well as to other BSW modules (see [1]):

- 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.

7.3.5.2 Triggered customers

The Synchronized Time-Base Manager currently only supports the OS as triggered customer. This means, the StbM supports synchronization of OS schedule tables to a given timebase.

The OS provides the API SyncScheduleTable() to synchronize a schedule table to a counter value

[SWS_StbM_00020]⌈
The Synchronized Time-Base Manager must be able to interact with the OS as *triggered customer*. The module calls the OS API for synchronizing OS ScheduleTables. ⌋ (SRS_BSW_00429, 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 timebase, which synchronzises the schedule table, is given by StbMSynchronizedTimeBaseRef.

It is configurable at pre-compile time if an OS ScheduleTable shall be synchronized with a synchronization 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 STBM_TRIGGERED_CUSTOMER_PERIOD (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.

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

[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).

7.4 Error Handling

7.4.1 Error classification

[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	***Relevance***	***Related error code***	***Value [hex]***
API requests called with wrong parameter	Development	STBM_E_PARAM	0x0A
Synchronized Time-Base Manager is not initialized	Development	STBM_E_NOT_INITIALIZED	0x0B
Invalid pointer in parameter list	Development	STBM_E_PARAM_POINTER	0x10
StbM initialization failed	Development	STBM_E_INIT_FAILED	0x011

⌋ (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 Manger does not perform an error handling for those kinds of errors.

7.4.2 Error detection

[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 8.1.4). This is to guarantee cyclic synchronization of OS schedule tables.⌋ (SRS_StbM_20007)

[SWS_StbM_00198]⌈

If the switch `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_GetCurrentTime()` 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.

7.4.3 Error notification

For details refer to the chapter 7.2 “Error Handling” in *SWS_BSWGeneral.*

7.4.4 Production Errors

There are no production errors defined by the Synchronized Time-Base Manager.

7.4.5 Extended Production Errors

There are no extended production errors defined by the Synchronized Time-Base Manager.

7.5 Version checking

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

7.6 AUTOSAR Service Interface

7.6.1 Architecture

In the AUTOSAR ECU Architecture (see [2]) the “Synchronized Time-base Manager” BSW module implements an AUTOSAR Service as indicated in Figure 6. The application can access the AUTOSAR Service “Synchronized Time-Base Manager” via service ports with standardized AUTOSAR Interfaces.

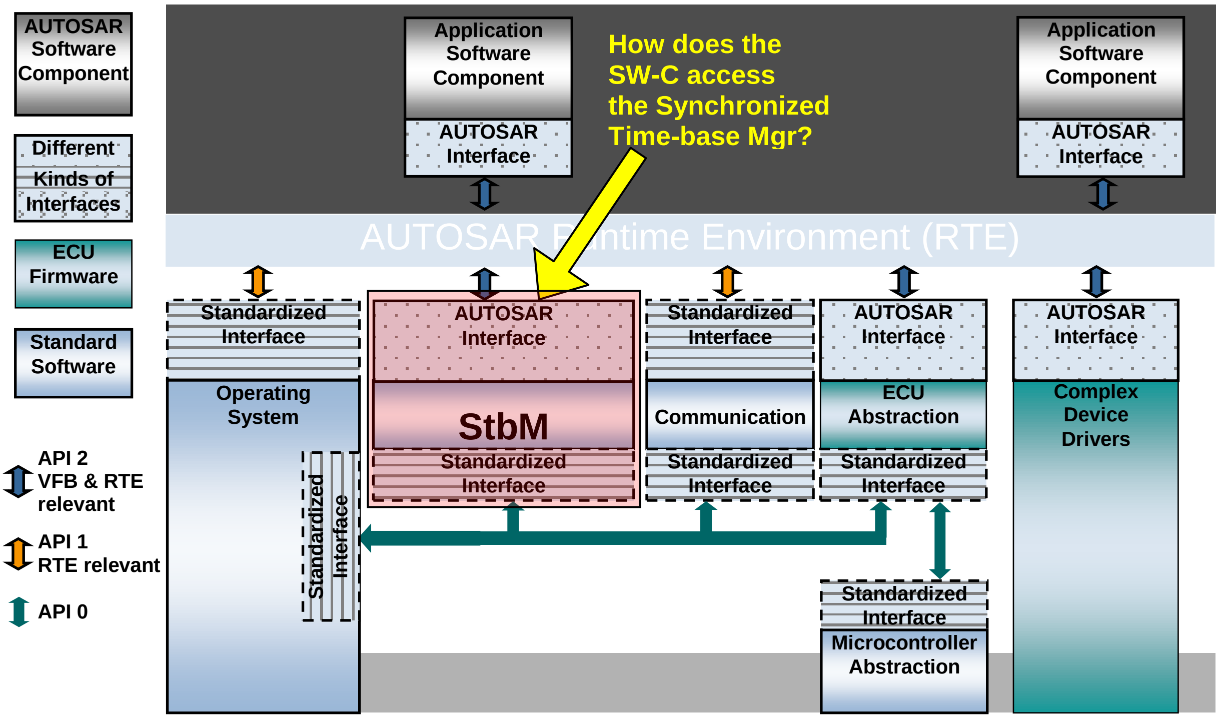


Figure 6: The Synchronized Time-Base Manager in the ECU architecture

This is described in terms of the AUTOSAR meta-model which will contribute to the SW-C Description of the application component as well as to the SW-C Description of the AUTOSAR Service “Synchronized Time-Base Manager”.

7.6.2 Requirements

There are three sources of requirements for the specification of the AUTOSAR Service “Synchronized Time-Base Manager”:

- The requirements for the functionality of the Synchronized Time-Base Manager are specified in [1].
- In order to model the VFB view of the service, the chapter on AUTOSAR Services of the VFB specification [7] has to be considered as an additional requirement.
- For the formal description of the SW-C attributes [8] gives the requirements.

7.6.3 Use Cases

As shown in Figure 1 the Synchronized Time-Base Manager is either interacting with the role *customer* or with the role *provider*. Application software components can only act as customers. Furthermore the Synchronized Time-Base Manager currently only supports the OS as triggered customer. Thus, an application software component as an "active" customer may use the service ports of the "Synchronized Time-Base Manager" service

- to read
    - current time
    - time status
- to set
    - the global time
    - the time offset
    - the user data.

7.6.4 Specification of the Ports and Port Interfaces

This chapter specifies the ports and port interfaces which are needed in order to operate the Synchronized Time-Base Manager functionality via the VFB. Note, that there are ports on both sides of the RTE: The SW-C description of the Synchronized Time-Base Manager defines the ports below the RTE. Each software component which uses the service must contain "service ports" in its own SW-C description which are connected to the ports of the Synchronized Time-Base Manager, so that the RTE can be generated.

The Port Interfaces for "active" customers to access the service are implemented as Client / Server interfaces (refer to 8.2.1). The corresponding ports are described in ch 8.2.3

Each software component must provide exactly one Port for each synchronized time base it wants to access.

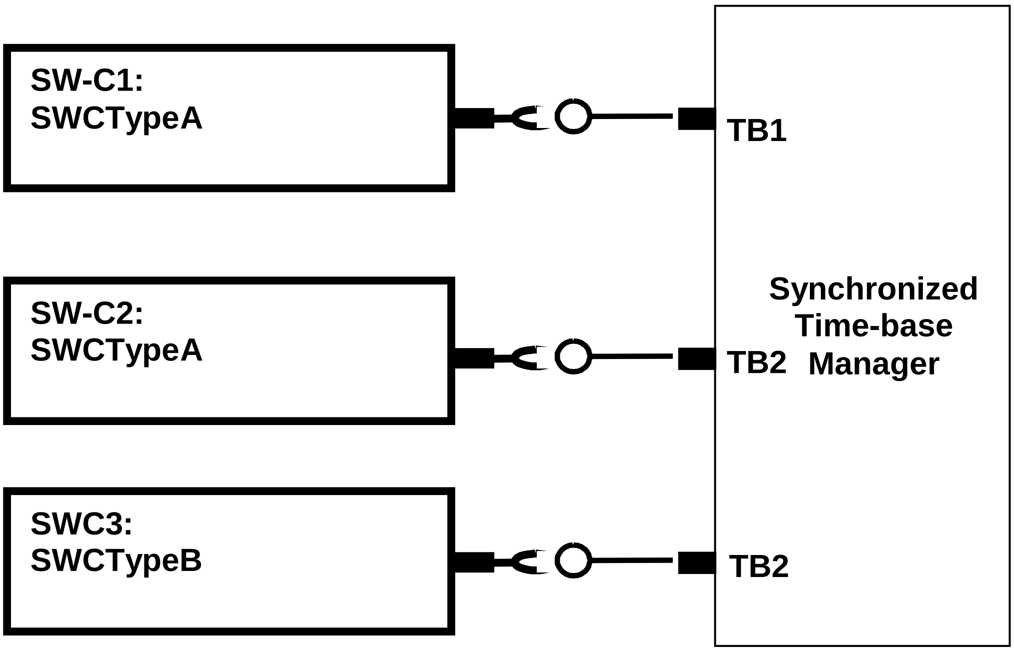


Figure 7: Example of application software components connected to the Synchronized Time-Base Manager via service ports. On the left side, there are two instances of component SWCTypeA and one instance of component SWCTypeB. The Port names on the right side define the requested synchronized time base. No notification ports are configured.

On the software component side, there is one Port for each synchronized time base. For each Port on component side, a respective Port on the service side exists. The name of the Ports on service side defines which synchronized time base is provided by the service. These names (e.g. TB1 for FlexRay time base and TB2 for TTCan time base) are examples and they are not standardized[1].

The mechanism of port-defined arguments (refer to [15] ch. “Port Defined Arguments Values”) is used by the RTE generator to derive from the port name the argument “timeBaseID” of the module API calls (refer to ch. 8.2.1.1 for relevant API calls).

7.6.5 Definition of the Service

The Provide Ports have a relation to the internal behavior of the Synchronized Time-Base Manager: With each call, the time base identifier is passed as an additional argument by the RTE to the C-function which implements the associated runnable entity.

Refer to 7.6.6 for how the required synchronized time base can be documented by augmenting the definition of Ports (feature “port defined argument value”).

[SWS_StbM_00131]⌈

[1] Obviously, it does not make sense to ask for the TTCan global time as synchronized time-base and invoking the operation “GetFrTime()”, getting the FlexRay time representation as clock time format. However, in order to define the Interface between application software component and Synchronized Time-base Manager as simple as possible, the Interface would allow such a invocation.

The InternalBehavior of the Synchronized Time-Base Manager is only seen by the local RTE. Besides the definition of the time base identifiers as port defined arguments, it must specify the operation invoked runnables:

```
InternalBehavior TimeService {

      // definition of associated operation-invoked RTE-events not shown
      // (it is done in the same way as for any SWC type)

      // section "runnable entities":
      RunnableEntity GetCurrentTime
            symbol "StbM_GetCurrentTime"
            canbeInvokedConcurrently = TRUE

      RunnableEntity GetCurrentTimeExtended
            symbol "StbM_GetCurrentTimeExtended"
            canbeInvokedConcurrently = TRUE

      RunnableEntity SetGlobalTime
            symbol "StbM_SetGlobalTime"
            canbeInvokedConcurrently = TRUE

      RunnableEntity SetUserData
            symbol "StbM_SetUserData"
            canbeInvokedConcurrently = TRUE

      RunnableEntity SetOffset
            symbol "StbM_SetOffset"
            canbeInvokedConcurrently = TRUE

      // end of section "runnable entities"

};
```

⌋ (SRS_StbM_20010)

7.6.6 Configuration of the Service

Ports, which are communicating with the Synchronized Time Base Manager are grouped together via the ServiceNeeds (see the Software Component Template specification for more information about ServiceNeeds [8]), when they are asking for the same time base.

The time base identifiers of the StbM service are modeled as "port defined argument values". Thus the configuration of those values is part of the RTE configuration. Pre-compile configuration can be done by changing the XML specification for the ports on the client (SW-C) or service (i.e. StbM) side.

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***
Dem	Dem_EventIdType
	Dem_EventStatusType
Os	AccessType
	ApplicationStateRefType
	ApplicationType
	CounterType
	ISRType
	MemorySizeType
	MemoryStartAddressType
	ObjectAccessType
	ObjectTypeType
	ScheduleTableStatusRefType
	ScheduleTableType
	StatusType
	TaskType
	TickRefType
	TickType
	TrustedFunctionIndexType
	TrustedFunctionParameterRefType
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_SynchronizedTimeBaseType

[SWS_StbM_00142]⌈

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

⌋ (SRS_StbM_20003, SRS_BSW_00305)

8.1.2.3 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)

8.1.3 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.

The StbM module's environment shall make sure that the DEM has been initialized before the StbM is initialized.

[SWS_StbM_00099]

⌈If development error detection is enabled, the StbM module shall report the development error STBM_E_INIT_FAILED when the initialization of the StbM module fails ⌋ (SRS_BSW_00337, SRS_BSW_00386)

[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)

[SWS_StbM_00250]

⌈ The Configuration pointer ConfigPtr shall always have a NULL_PTR value⌋ (SRS_BSW_00414)

Note: The StbM currently only supports Pre-Compile parameters (refer to **[SWS_StbM_00245]**). The Configuration pointer ConfigPtr is therefore currently not used and shall therefore be set to NULL_PTR value.

8.1.3.3 StbM_GetCurrentTime

[SWS_StbM_00195]⌈

Service name:	StbM_GetCurrentTime	

Syntax:	`Std_ReturnType StbM_GetCurrentTime(`
	`    StbM_SynchronizedTimeBaseType timeBaseId,`
	`    StbM_TimeStampType* timeStampPtr,`
	`    StbM_UserDataType* userDataPtr`
	`)`	

Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	timeBaseId	time base reference
Parameters (inout):	None	
Parameters (out):	timeStampPtr	Current time stamp that is valid at this time
	userDataPtr	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_20029)

[SWS_StbM_00196]⌈
If the switch `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_GetCurrentTime()` shall report to DET the development error `STBM_E_PARAM`, if called with an invalid parameter `timeBaseID` (e.g. not configured).⌋ (SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00197]⌈
If the switch `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_GetCurrentTime()` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with an invalid pointer of parameter `timeStampPtr` and `userDataPtr`.⌋ (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* timeStampPtr,`
	`StbM_UserDataType* userDataPtr`
	`)`	

Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	timeBaseId	time base reference
Parameters (inout):	None	
Parameters (out):	timeStampPtr	Current time stamp that is valid at this time
	userDataPtr	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 `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_GetCurrentTimeExtended()` shall report to DET the development error `STBM_E_PARAM`, if called with an invalid parameter `timeBaseID` (e.g. not configured).
⌋ (SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00202]⌈
If the switch `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_GetCurrentTimeExtended()` shall report to DET the development error

STBM_E_PARAM_POINTER, if called with an invalid pointer of parameter timeStampPtr and userDataPtr.⌋ (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_TimeStampRawType* timeStampRawPtr`
	`)`	

Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
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 a time value in raw format from the most accurate time source.	

⌋ (SRS_StbM_20016)

[SWS_StbM_00206]⌈
If the switch STBM_DEV_ERROR_DETECT (**ECUC_StbM_00012 :**) is set to True, StbM_GetCurrentTimeRaw() shall report to DET the development error STBM_E_PARAM_POINTER, if called with an invalid pointer of parameter timeStampRawPtr.⌋ (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_TimeStampRawType givenTimeStamp,`
	`    StbM_TimeStampRawType* timeStampDiffPtr`
	`)`	

Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	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 the time difference of current time raw that is valid at this time minus given time raw by using a most accurate time source.	

⌋ (SRS_StbM_20016)

[SWS_StbM_00210]⌈
If the switch `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_GetCurrentTimeDiff()` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with an invalid pointer of parameter `timeStampDiffPtr`.⌋ (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* timeStampPtr,`
	`const StbM_UserDataType* userDataPtr`
	`)`	

Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	timeBaseId	time base reference
	timeStampPtr	New time stamp
	userDataPtr	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 and modify HW registers behind the providers, if supported. This function will be used if a Time Master is present in this ECU.	

⌋ (SRS_StbM_20026)

[SWS_StbM_00214]⌈
If the switch `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_SetGlobalTime()` shall report to DET the development error `STBM_E_PARAM`, if called with an invalid parameter `timeBaseID` (e.g. not configured).⌋ (SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00215]
⌈ If the switch `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_SetGlobalTime()` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with an invalid pointer of parameter `timeStampPtr`.⌋ (SRS_BSW_00386, SRS_BSW_00323)

8.1.3.8 StbM_SetUserData

[SWS_StbM_00218]⌈

Service name:	StbM_SetUserData	

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

Service ID[hex]:	0x0c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	timeBaseId	time base reference
	userDataPtr	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 `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_SetUserData()` shall report to DET the development error `STBM_E_PARAM`, if called with an invalid parameter `timeBaseID` (e.g. not configured).⌋ (SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00220]⌈
If the switch `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_SetUserData()` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with an invalid pointer of parameter `userDataPtr`.⌋ (SRS_BSW_00386, SRS_BSW_00323)

8.1.3.9 StbM_SetOffset

[SWS_StbM_00223]⌈

Service name:	StbM_SetOffset	

Syntax:	`Std_ReturnType StbM_SetOffset(`
	`    StbM_SynchronizedTimeBaseType timeBaseId,`
	`    const StbM_TimeStampType* timeStampPtr`
	`)`	

Service ID[hex]:	0x0d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	timeBaseId	time base reference
	timeStampPtr	New offset time stamp
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description:	Allows the Customers and the Timebase Provider Modules to set the offset time	

	that has to be valid for the system.

⌋ (SRS_StbM_20028)

[SWS_StbM_00224]⌈
If the switch `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_SetOffset()` shall report to DET the development error `STBM_E_PARAM`, if called with an invalid parameter `timeBaseID` (e.g. not configured).⌋ (SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00225]⌈
If the switch `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_SetOffset()` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with an invalid pointer of parameter `timeStampPtr`.⌋ (SRS_BSW_00386, SRS_BSW_00323)

8.1.3.1 StbM_GetOffset

[SWS_StbM_00228]⌈

Service name:	StbM_GetOffset	

Syntax:	`Std_ReturnType StbM_GetOffset(`
	`    StbM_SynchronizedTimeBaseType timeBaseId,`
	`    StbM_TimeStampType* timeStampPtr`
	`)`	

Service ID[hex]:	0x0e	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	timeBaseId	time base reference
Parameters (inout):	None	
Parameters (out):	timeStampPtr	Current offset time stamp
Return value:	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description:	Allows the Timebase Provider Modules to get the currentoffset time.	

⌋ (SRS_StbM_20027)

[SWS_StbM_00229]⌈
If the switch `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_GetOffset()` shall report to DET the development error `STBM_E_PARAM`, if called with an invalid parameter `timeBaseID` (e.g. not configured).⌋ (SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00230]⌈
If the switch `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_GetOffset()` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with an invalid pointer of parameter `timeStampPtr`.⌋ (SRS_BSW_00386, SRS_BSW_00323)

8.1.3.2 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,`
	`    boolean syncToTimeBase`
	`)`	

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)
	syncToTimeBase	SYNC to GTM (0) clear the SYNC_TO_GATEWAY bit within the status SYNC to sub-domain (1) set the SYNC_TO_GATEWAY bit within the status
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description:	Allows the Timebase Provider Modules to forward a new Global Time to the StbM, which has been received from different busses.	

⌋ (SRS_StbM_20014)

[SWS_StbM_00234]⌈
If the switch `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_BusSetGlobalTime()` shall report to DET the development error `STBM_E_PARAM`, if called with an invalid parameter `timeBaseID` (e.g. not configured).⌋ (SRS_BSW_00386, SRS_BSW_00323)

[SWS_StbM_00235]
⌈ If the switch `STBM_DEV_ERROR_DETECT` (**ECUC_StbM_00012 :**) is set to True, `StbM_BusSetGlobalTime()` shall report to DET the development error `STBM_E_PARAM_POINTER`, if called with an invalid pointer of parameter `timeStampPtr`.⌋ (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_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 Expected Interfaces

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

8.1.5.1 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***
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function. OBD Events Suppression shall be ignored for this computation.
Det_ReportError	Service to report development errors.

⌋ (SRS_BSW_00301, SRS_BSW_00339)

8.1.5.2 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***
AllowAccess	This service sets the own state of an OS-Application from APPLICATION_RESTARTING to APPLICATION_ACCESSIBLE.
CallTrustedFunction	A (trusted or non-trusted) OS-Application uses this service to call a trusted function
CheckISRMemoryAccess	This service checks if a memory region is write/read/execute accessible and also returns information if the memory region is part of the stack space.
CheckObjectAccess	This service determines if the OS-Applications, given by ApplID, is allowed to use the IDs of a Task, Resource, Counter, Alarm or Schedule Table in API calls.
CheckObjectOwnership	This service determines to which OS-Application a given Task, ISR, Counter, Alarm or Schedule Table belongs

CheckTaskMemoryAccess	This service checks if a memory region is write/read/execute accessible and also returns information if the memory region is part of the stack space.
GetApplicationID	This service determines the OS-Application (a unique identifier has to be allocated to each application) where the caller originally belongs to (was configured to).
GetApplicationState	This service returns the current state of an OS-Application.
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.
GetISRID	This service returns the identifier of the currently executing ISR.
GetScheduleTableStatus	This service queries the state of a schedule table (also with respect to synchronization).
IncrementCounter	This service increments a software counter.
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)

8.1.5.3 Configurable Interfaces

None

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 Client-Server-Interfaces

8.2.1.1 StbM_GlobalTime_Master

[SWS_StbM_00240]
⌈

Name	GlobalTime_Master_{Name}	
Comment	--	
IsService	true	
Variation	(({ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(CanTSyn/CanTSynGlobalTimeDomain/CanTSynSynchronizedTimeBaseRef->StbMSynchronizedTimeBase)}) && ({ecuc(CanTSyn/CanTSynGlobalTimeDomain/CanTSynGlobalTimeMaster)}!=NULL)) || (({ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(FrTSyn/FrTSynGlobalTimeDomain/FrTSynSynchronizedTimeBaseRef->StbMSynchronizedTimeBase)}) && ({ecuc(FrTSyn/FrTSynGlobalTimeDomain/FrTSynGlobalTimeMaster)}!=NULL)) || (({ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynSynchronizedTimeBaseRef->StbMSynchronizedTimeBase)}) && ({ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynGlobalTimeMaster)}!=NULL)) Name = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK

Operations

SetGlobalTime		
Comments	Allows the Customers to set the new global time that has to be valid for the system, which will be sent to the busses and modify HW registers behind the providers, if supported. This function will be used if a Time Master is present in this ECU.	
Variation	{ecuc(StbM/StbMSynchronizedTimeBase/StbMIsSystemWideGlobalTimeMaster)} == TRUE	
Parameters	timeStampPtr	
	Comment	--
	Type	StbM_TimeStampType
	Variation	--
	Direction	IN
	userDataPtr	
	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 Timebase Provider Modules to set the offset time that has to be valid for the system.	
Variation	({ecuc(CanTSyn/CanTSynGlobalTimeDomain/CanTSynGlobalTimeDomainId)}>15) || ({ecuc(FrTSyn/FrTSynGlobalTimeDomain/FrTSynGlobalTimeDomainId)}>15) || ({ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynGlobalTimeDomainId)}>15)	
Parameters	timeStampPtr	
	Comment	--
	Type	StbM_TimeStampType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
SetUserData		
Comments	Allows the Customers to set the new user data that has to be valid for the system, which will be sent to the busses.	
Variation	--	
Parameters	userDataPtr	
	Comment	New user data
	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)

8.2.1.2 StbM_GlobalTime_Slave

[SWS_StbM_00247]
⌈

Name	GlobalTime_Slave
Comment	--
IsService	true

Variation	--	
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	timeStampPtr	
	Comment	--
	Type	StbM_TimeStampType
	Variation	--
	Direction	OUT
	userDataPtr	
	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/General/StbMGetCurrentTimeExtendedAvailable)}	
Parameters	timeStampPtr	
	Comment	--
	Type	StbM_TimeStampExtendedType
	Variation	--
	Direction	OUT
	userDataPtr	

	Comment	--
	Type	StbM_UserDataType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	Operation failed

⌋ (SRS_StbM_20003, SRS_StbM_20010, SRS_StbM_20029)

8.2.2 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.

[SWS_StbM_00124]]

⌈The data types `uint8, uint16` and `sint32` used in the interfaces refer to the basic AUTOSAR data types.⌋(SRS_BSW_00306)

8.2.2.1 StbM_SynchronizedTimeBaseType

[SWS_StbM_00150]]

⌈

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)

8.2.2.2 StbM_TimeBaseStatusType

[SWS_StbM_00239]

⌈

Name	StbM_TimeBaseStatusType

Kind	Structure (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	TIMELEAP	0x02	Bit 1: 0x00: No leap within the received time 0x02: Leap within the received time that exceeds a configured threshold
	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
Description	Variables of this type are used to expresse 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 and SYNC_TO_GATEWAY can only be set if the GLOBAL_TIME_BASE bit is set.			

⌋ (SRS_StbM_20025)

8.2.2.3 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 acc. to "[17], Annex C/C1" as specified for PTP. 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)

8.2.2.4 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 acc. to "[17], Annex C/C1" as specified for PTP.		
Variation	--		

⌋ (SRS_StbM_20012)

8.2.2.5 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)

8.2.3 Ports

8.2.3.1 GlobalTime_Master

[SWS_StbM_00244]

⌈

Name	GlobalTime_Master_{Name}		
Kind	ProvidedPort	Interface	GlobalTime_Master_{Name}
Description	--		
Variation	(({ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(CanTSyn/ CanTSynGlobalTimeDomain/CanTSynSynchronizedTimeBaseRef- >StbMSynchronizedTimeBase)}) && ({ecuc(CanTSyn/CanTSynGlobalTimeDomain/ CanTSynGlobalTimeMaster)}!=NULL)) || (({ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(FrTSyn/FrTSynGlobalTimeDomain/FrTSynSynchronizedTimeBaseRef- >StbMSynchronizedTimeBase)}) && ({ecuc(FrTSyn/FrTSynGlobalTimeDomain/ FrTSynGlobalTimeMaster)}!=NULL)) || (({ecuc(StbM/StbMSynchronizedTimeBase)} == {ecuc(EthTSyn/EthTSynGlobalTimeDomain/EthTSynSynchronizedTimeBaseRef- >StbMSynchronizedTimeBase)}) && ({ecuc(EthTSyn/EthTSynGlobalTimeDomain/ EthTSynGlobalTimeMaster)}!=NULL)) Name = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		

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

8.2.3.2 GlobalTime_Slave

[SWS_StbM_00248]

⌈

Name	GlobalTime_Slave_{Name}		
Kind	ProvidedPort	Interface	GlobalTime_Slave
Description	--		
Variation	Name = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)}		

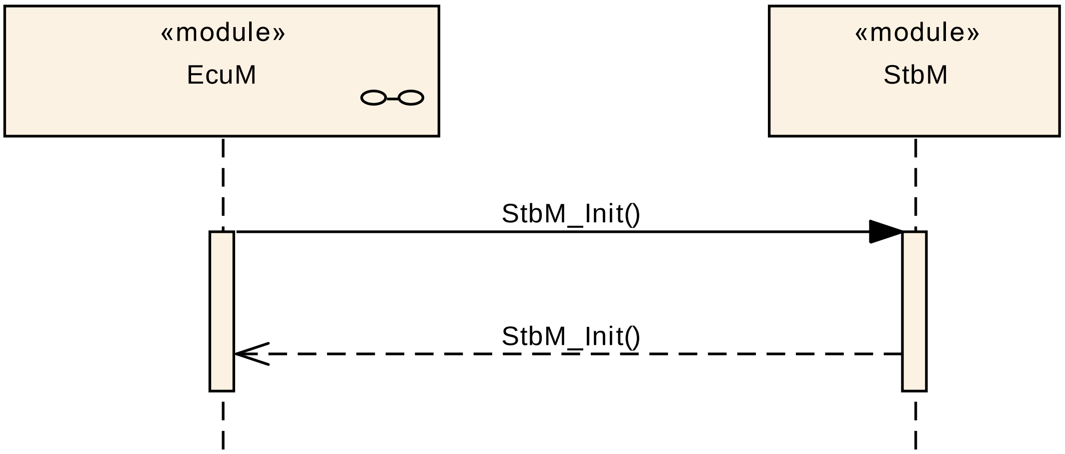
⌋ (SRS_StbM_20003, SRS_StbM_20010, SRS_StbM_20029)

9 Sequence diagrams

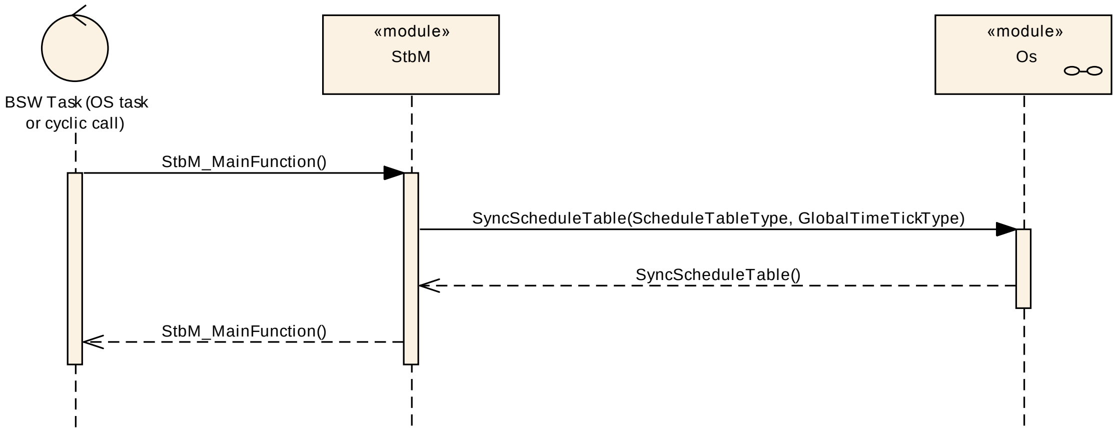
The sequence diagrams in this chapter show the basic operations of the Synchronized Time-Base Manager.

Please note that the sequence diagrams are an extension for illustrational purposes to ease understanding of the specification.

9.1 StbM_Init



9.2 Explicit synchronization of OS ScheduleTable


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

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 configuration tool must check the consistency of the configuration at configuration time.

10.2.1 Variants

The Synchronized Time-base Manager supports only pre-compile parameters.

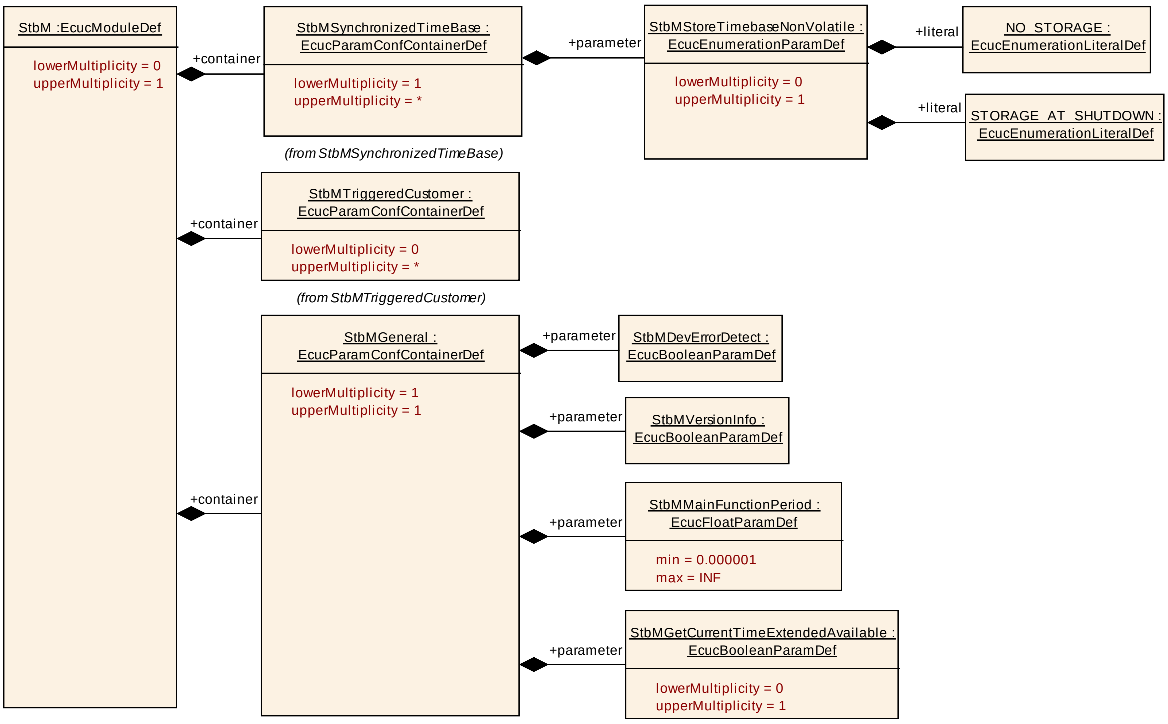
[SWS_StbM_00245]
⌈ The Synchronized Time-base Manager shall support the configuration variant VARIANT-PRE-COMPILE.⌋ (SRS_BSW_00345)

10.2.2 StbM

Module Name	*StbM*
Module Description	Configuration of the Synchronized Time-base Manager (StbM) module.
Post-Build Variant Support	false

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.



10.2.3 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_00012 :**		
Name	StbMDevErrorDetect		
Description	Switches the Default Error Tracer (Det) detection and notification ON or OFF. • true: enabled (ON). • false: disabled (OFF).		
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_00032 :**		
Name	StbMGetCurrentTimeExtendedAvailable		
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		
Description	Schedule period of the main function StbM_MainFunction. Unit: [s].		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	1E-6 .. 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_00013 :**		
Name	StbMVersionInfo		
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	--		
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

10.2.4 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.
Configuration Parameters	

SWS Item	**ECUC_StbM_00036 :**		
Name	StbMIsSystemWideGlobalTimeMaster		
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 start several global time domains on different busses it is connected to.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
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		
Description	This allows for specifying that the timebase 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_00029 :**		
Name	StbMSyncLossThreshold		
Description	This represents the minimum delta between the time value in two sync messages for which the sync loss flag is set. 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_00028 :**		
Name	StbMSyncLossTimeout		
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_00021 :**		
Name	StbMSynchronizedTimeBaseIdentifier		
Description	Identification of a synchronized time-base via a unique identifier.		
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		

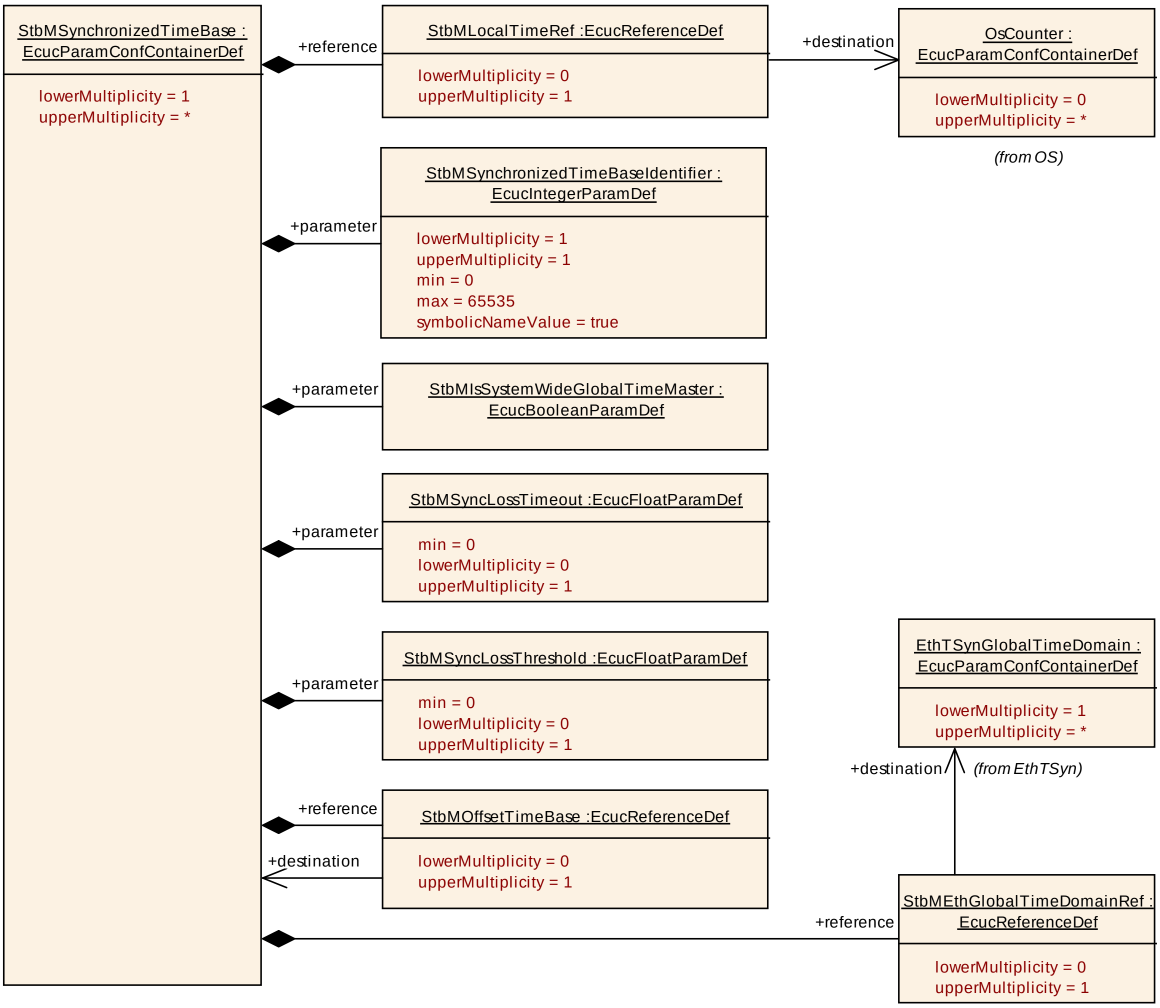
SWS Item	**ECUC_StbM_00033 :**		
Name	StbMEthGlobalTimeDomainRef		
Description	Optional sub container in case a local time shall be accessed on an Ethernet bus.		
Multiplicity	0..1		
Type	Reference to [EthTSynGlobalTimeDomain]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
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_00006 :**
Name	StbMLocalTimeRef
Description	This represents an optional sub-container in case a local time shall be accessed. In this case, the designated OS counter has to be configured properly,

	meaning: • the counter is directly driven by a HW timer. • the counter's OsCounterTicksPerBase is one tick in x nanoseconds (ns).		
Multiplicity	0..1		
Type	Reference to [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		

SWS Item	**ECUC_StbM_00030 :**		
Name	StbMOffsetTimeBase		
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		

No Included Containers



10.2.5 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.
Configuration Parameters	

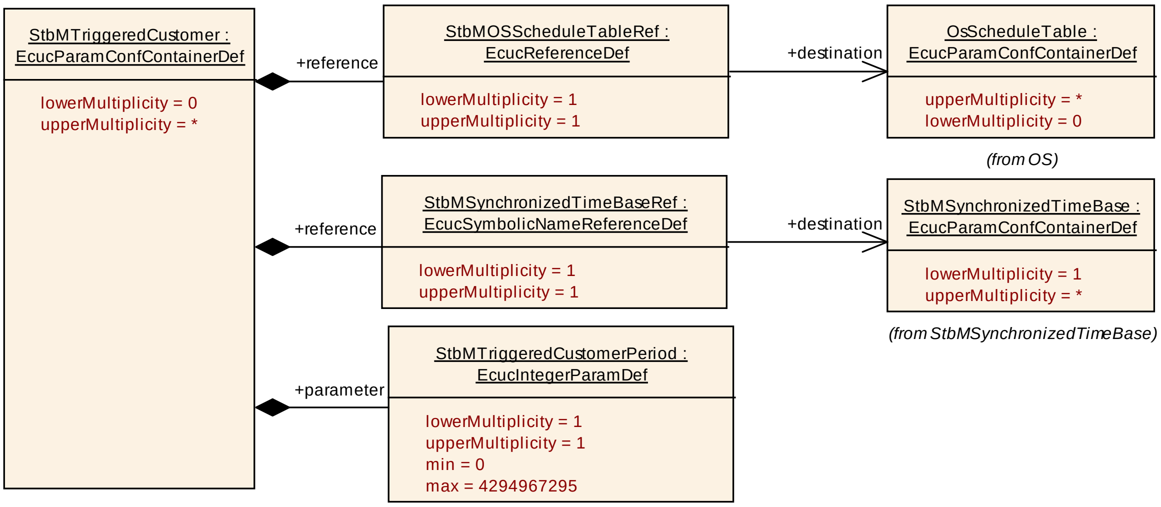
SWS Item	**ECUC_StbM_00020 :**		
Name	StbMTriggeredCustomerPeriod		
Description	The triggering period of the triggered customer, called by the StbM_MainFunction. The period is documented in microseconds.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
Post-Build 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		
Description	Mandatory reference to synchronized OS ScheduleTable, which will be explicitly synchronized by the StbM.		
Multiplicity	1		
Type	Reference to [OsScheduleTable]		
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_00010 :**		
Name	StbMSynchronizedTimeBaseRef		
Description	Mandatory reference to the required synchronized time-base.		
Multiplicity	1		
Type	Symbolic name reference to [StbMSynchronizedTimeBase]		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers



10.3 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)