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

| | |
|---|---|
| **Document Version** | 2.2.1 |
| **Document Status** | Final |
| **Part of Release** | 4.1 |
| **Revision** | 3 |

## Document Change History

| Date | Version | Changed by | Change Description |
|---|---|---|---|
| 31.03.2014 | 2.2.1 | AUTOSAR Release Management | • Clarification on Autonomous Time Maintenance |
| 31.10.2013 | 2.2.0 | AUTOSAR Release Management | • Parameter StbMMainFunctionPeriod added<br>• Requirements StbM_0030 and 00035 removed<br>• Restructuring of and clarification w.r.t. Service Interface related chapters<br>• Parameters StbMFlexRayClusterRef / StbMTtcanClusterRef set to obsolete<br>• Editorial changes<br>• Removed chapter(s) on change documentation |
| 28.02.2013 | 2.1.0 | AUTOSAR Administration | • Added “Known Limitations”<br>• Contradictions in error handling removed<br>• Added chapter service interfaces<br>• Added Subchapter 3.x due to SWS General Rollout<br>• Reworked according to the new SWS_BSWGeneral |
| 03.11.2011 | 2.0.0 | AUTOSAR Administration | • Added functionality for absolute time provision |
| 30.11.2009 | 1.0.0 | AUTOSAR Administration | Initial Release |

# Table of Contents

**Known Limitations**
**Caution:**

**Currently a rework of the Synchronized Time Base Manager according to a new, global time synchronization concept is under discussion.**

**Therefore backward compatibility might be affected after R4.1.2.**

# 1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the Synchronized Time-Base Manager (StbM).

The basic purpose of the Synchronized Time-Base Manager is to provide a “global time” to other BSW modules or to the application. Global time means, that different entities within the system have the same definition of time and passage of time. In the system, multiple “global times” can exist simultaneously (e.g. the FlexRay time definition, the TTCAN time definition …).
A second timebase called “AbsoluteTime” is provided for long term synchronizations realized by counting timer overflows of the global time.

In addition, the Synchronized Time-Base Manager shall provide means for state/error notification (e.g. loss and (re-) establishment of global time).

Main use cases for the application of the Synchronized Time-Base Manager are the following:

- Synchronized execution of several RunnableEntities within one system cluster
- Provision of an accurate definition of the absolute value of time
- Provision of an accurate definition of the passage of time

In section 1.1, a detailed definition of potential use cases are specified.

**Important note**: The proposed solution for the Synchronized Time-Base Manager mainly focuses on the “Synchronization of RunnableEntities”. However, further refinements of the specification shall satisfy as well the other use cases.

## 1.1 Use Cases

### 1.1.1 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. However, as the limitations in section 4.1.1 indicate, only ECUs in the same network cluster are considered for the establishment of a synchronized time-base. Thus, the fulfillment of the requirement described above can only be guaranteed by deploying the related software components within the same cluster (e.g. FlexRay or TTCAN).

A classical application of this use cases is the sensor data read out or synchronous actuator triggering by different RunnableEntities.

### 1.1.2 Time provision

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

A classical application of this use case is the access to synchronized calendar time by the application, e.g. for diagnostic events storage.

Other possible scenarios:

- Measuring the passage of time between two tagged system states (e.g. start and end of a RunnableEntity, deadline monitoring for timing-relevant functions).
- Guaranteeing the accurate triggering of OS alarms every (well-defined) interval

### 1.1.3 Notification mechanism

The application (and other BSW modules) shall have the possibility of getting informed about the current status of the definition of time within the cluster. Thus, some kind of notification mechanism is required, which informs the application (and other BSW modules) upon state changes or error occurrences.

### 1.1.4 Absolute time provision

In some cases an absolute time value (relative to the start of the ECU) is needed.
E.g. in safety related systems it might be necessary to test the system every 48 hours.

## 1.2 Synchronized Time-Base Manager as broker

Basically, the Synchronized Time-Base Manager is interacting with two different roles, as Figure 1 shows:

1) **Provider**
The StbM requires information from other modules (e.g. receiving the global time defined by the FlexRay Interface). So far, the *provider* has the task to deliver a “synchronized time-base”.

2) **Customer**
The StbM collects information about time. This functionality can be used by several *customers*: either other BSW modules or application SW-Cs.

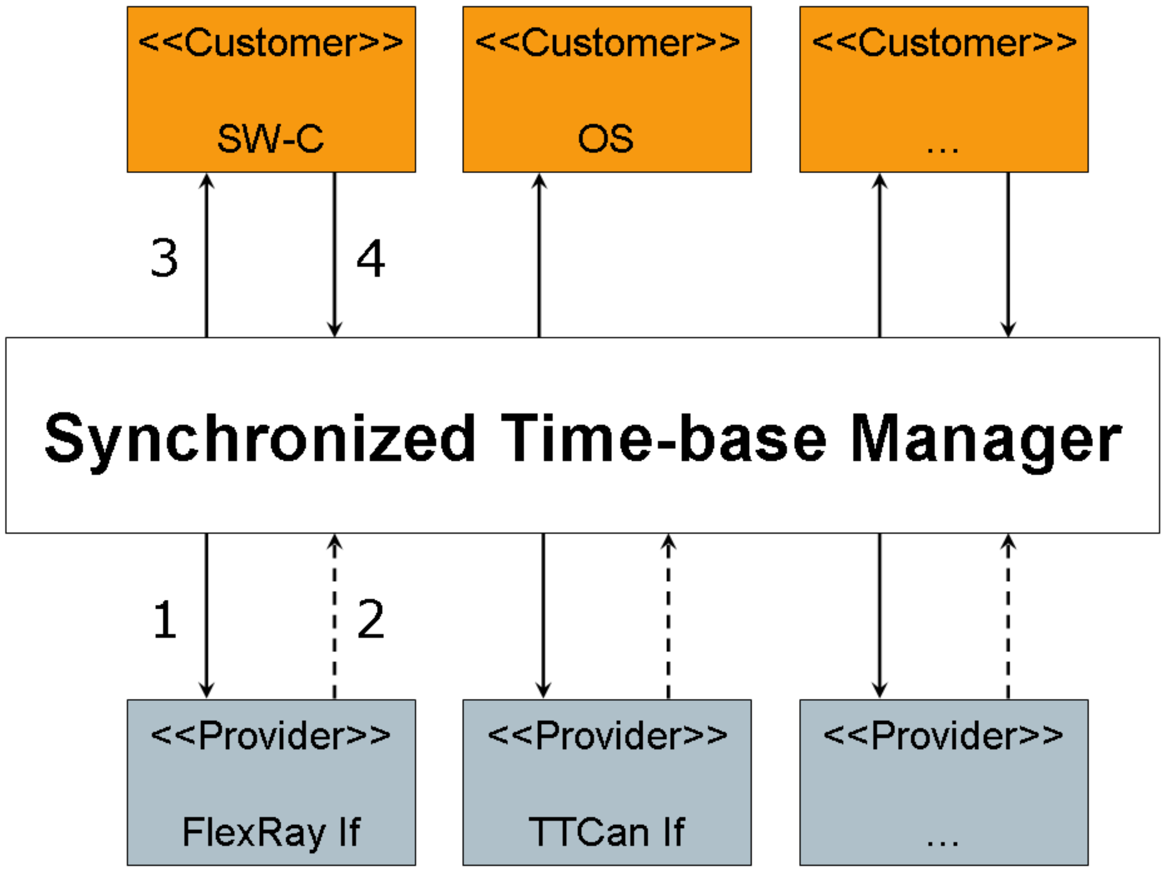


**Figure 1: Synchronized Time-Base Manager as consumer and provider**

So to say, the Synchronized Time-Base Manager acts as **time-base broker** by offering the customers access to synchronized time-bases. Doing this, the Synchronized Time-Base Manager abstracts from the “real” time-base provider.

In the following, the different providers and customers and the meaning of the several arrows in the figure above are described in a more detailed way.

## 1.3 Provider

The Synchronized Time-Base Manager itself does not provide any facility (e.g. protocols) for establishing a synchronized time across multiple nodes. Thus, the service requires other modules within the AUTOSAR BSW stack, which can provide this functionality. The FlexRay Interface and the TTCAN Interface are examples of modules which provide the definition of a synchronized time-base.

Calling the provider (arrow “1” in Figure 1) or getting called by the provider (arrow “2” in Figure 1) are the possible ways how to communicate with the provider.

## 1.4 Customer

The Synchronized Time-Base Manager has the requirement of satisfying the customers need in regard to time and passage of time. This section describes the different classes of customers and how they access the functionality.

**Note**: The classes are not completely disjoint. Thus, one specific customer can potentially be mapped to different classes of customers.

a) **Triggered customer**
This kind of customer is triggered by the Synchronized Time-Base Manager. Thus, the module itself is aware of the required functionality of the customer, and uses the defined interface of the customer to access it (e.g. accessing the Os SyncScheduleTable() API for synchronizing the respective OS ScheduleTable with the global time).

b) **Active customer**
This kind of customer autonomously calls the Synchronized Time-Base Manager, getting knowledge about the global time value (e.g. asking for the actual date and time).

c) **Notification customer**
This kind of customer is interested in the current state of the Synchronized Time-Base Manager and wants to get informed about state changes and/or error occurrences (e.g. loss of global time).

Customers a) and c) are triggered by the Synchronized Time-Base Manager (arrow “3” in Figure 1). Customer b) accesses itself the Synchronized Time-Base Manager (arrow “4” in Figure 1).

# 2 Acronyms and abbreviations

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

| ***Abbreviation / Acronym:*** | ***Description:*** |
|---|---|
| StbM | Synchronized Time-Base Manager |
| | |
| | |
| | |

# 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

## 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 Cluster

For simplicity, only ECUs in the same network cluster are considered. A vehicle-wide definition of time can be achieved by introducing time gateways between clusters but this is left to the implementer.

### 4.1.2 Time agreement protocol

The current solution of the Synchronized Time-Base Manager does not provide its own time agreement protocol. Thus, the Synchronized Time-Base Manager shall use the functionality of time-base *provider* as defined in 1.3[1].

### 4.1.3 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.

### 4.1.4 Mode switches

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

### 4.1.5 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, not an issue of the Synchronized Time-Base Manager).
- Errors occurred during interaction with *customers*. Let's take the OS as example. Calling the explicit OS ScheduleTable synchronization may cause

[1] This limitation is only required for distributed global time needs. Obviously, when only a single ECU is considered (e.g. when the whole functionality is deployed to only one ECU), a global time is not explicitly required.

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 Terminology

So far, referencing a common definition of time has been declared throughout as “global time” in this document. However, with the help of this section a more precise definition of the term “global time” is given.

**Important note**: We avoid the use of the term “global time” for the rest of this specification, as there is no single “global time”. Instead, we use the term “synchronized time-base”.

For the purposes of this specification, we use the terms “physical time-base”, “software time-base” and “synchronized time-base” as follows:

### 4.2.1 Physical Time-Base

**Definition:** A physical time-base is a physical device which allows obtaining knowledge about the passage of time based on its physical properties (e.g. oscillation of a quartz crystal with a known frequency, atomic clock, earth rotation).

### 4.2.2 Software Time-Base

**Definition:** A software time-base is a software-provided measure of time derived from one or more physical time-bases. A time-base can be realized as

- an event source (e.g. a regular interrupt or alarm)
- a clock value data entity (e.g. a hardware or software counter representing time)

A software time-base is a means for *customers* to be triggered based on the passage of time and/or to obtain knowledge about the passage of time.

### 4.2.3 Synchronized Time-Base

**Definition:** A synchronized time-base is a software time-base existing at a processing entity (actor / processor / node of a distributed system) that is synchronized with software 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”, like the so called “FlexRay global time”. We do not use the term “global time” here because it is an important feature of the module specified in this document that it can cope with different synchronized time-bases which may vary in terms of rate and absolute value.

For long term time determination a timer overflow counting mechanism provides an AbsoluteTime.

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.

Note: A synchronized time-base is derived in a defined way (time protocol or time agreement protocol) from a defined set of physical time-bases.

## 4.3 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.

In addition, the cluster limitation as described in 4.1.1 must be considered as restriction for the applicability.

## 4.4 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_BSWGeneral.*

## 5.2 Header file structure

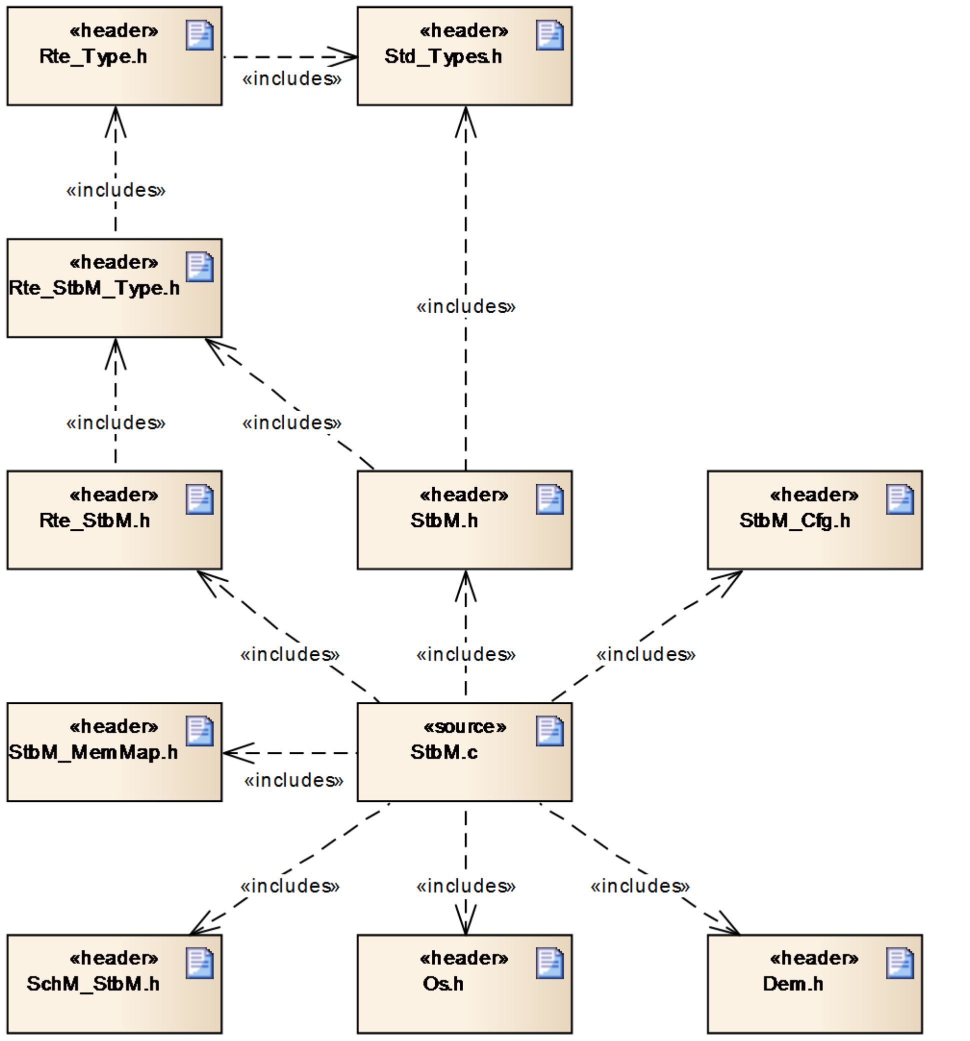
**[SWS_StbM_00002]**⌈
The module shall include the StbM.h file, providing the function prototypes to access the underlying functions of the module. ⌋ ( )

**[SWS_StbM_00065]**⌈
In addition, the header files structure shall contain the following header files:

Os.h: general header file of the Operating System

StbM.c shall include Os.h, Dem.h and MemMap.h.
⌋ (SRS_BSW_00384, SRS_BSW_00339, SRS_BSW_00436)

**[SWS_StbM_00115]**⌈
Only StbM.h shall be included by the users of the Synchronized Time-Base Manager⌋ ( )


«header» Rte_Type.h
«header» Std_Types.h
«includes»
«header» Rte_StbM_Type.h
«header» Rte_StbM.h
«header» StbM.h
«header» StbM_Cfg.h
«source» StbM.c
«header» StbM_MemMap.h
«header» SchM_StbM.h
«header» Os.h
«header» Dem.h

.

# 6 Requirements traceability

| Requirement | Description | Satisfied by |
|---|---|---|
| - | - | SWS_StbM_00002 |
| - | - | SWS_StbM_00006 |
| - | - | SWS_StbM_00012 |
| - | - | SWS_StbM_00013 |
| - | - | SWS_StbM_00019 |
| - | - | SWS_StbM_00051 |
| - | - | SWS_StbM_00059 |
| - | - | SWS_StbM_00078 |
| - | - | SWS_StbM_00079 |
| - | - | SWS_StbM_00086 |
| - | - | SWS_StbM_00087 |
| - | - | SWS_StbM_00088 |
| - | - | SWS_StbM_00089 |
| - | - | SWS_StbM_00090 |
| - | - | SWS_StbM_00091 |
| - | - | SWS_StbM_00093 |
| - | - | SWS_StbM_00094 |
| - | - | SWS_StbM_00097 |
| - | - | SWS_StbM_00098 |
| - | - | SWS_StbM_00099 |
| - | - | SWS_StbM_00100 |
| - | - | SWS_StbM_00101 |
| - | - | SWS_StbM_00102 |
| - | - | SWS_StbM_00103 |
| - | - | SWS_StbM_00104 |
| - | - | SWS_StbM_00105 |
| - | - | SWS_StbM_00106 |
| - | - | SWS_StbM_00107 |
| - | - | SWS_StbM_00108 |
| - | - | SWS_StbM_00109 |
| - | - | SWS_StbM_00110 |
| - | - | SWS_StbM_00111 |
| - | - | SWS_StbM_00112 |
| - | - | SWS_StbM_00113 |
| - | - | SWS_StbM_00114 |
| - | - | SWS_StbM_00115 |

| | | |
|---|---|---|
| - | - | SWS_StbM_00116 |
| - | - | SWS_StbM_00117 |
| - | - | SWS_StbM_00118 |
| - | - | SWS_StbM_00121 |
| - | - | SWS_StbM_00125 |
| - | - | SWS_StbM_00126 |
| - | - | SWS_StbM_00127 |
| - | - | SWS_StbM_00128 |
| - | - | SWS_StbM_00129 |
| - | - | SWS_StbM_00130 |
| - | - | SWS_StbM_00131 |
| - | - | SWS_StbM_00133 |
| - | - | SWS_StbM_00134 |
| - | - | SWS_StbM_00135 |
| - | - | SWS_StbM_00136 |
| - | - | SWS_StbM_00137 |
| - | - | SWS_StbM_00138 |
| - | - | SWS_StbM_00141 |
| - | - | SWS_StbM_00142 |
| - | - | SWS_StbM_00143 |
| - | - | SWS_StbM_00144 |
| SRS_BSW_00004 | All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files | SWS_StbM_00068 |
| SRS_BSW_00005 | Modules of the æC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces | SWS_StbM_00140 |
| SRS_BSW_00006 | The source code of software modules above the æC 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_00159 | All modules of the AUTOSAR Basic Software shall support a tool based configuration | SWS_StbM_00062 |
| 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_00167 | All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks | SWS_StbM_00062, SWS_StbM_00063 |
| 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_00304 | - | SWS_StbM_00140 |
| SRS_BSW_00305 | Data types naming convention | SWS_StbM_00149, SWS_StbM_00150, SWS_StbM_00151, SWS_StbM_00152 |
| SRS_BSW_00306 | AUTOSAR Basic Software Modules shall be compiler and platform independent | SWS_StbM_00124, SWS_StbM_00140 |
| 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_00325 | The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short | SWS_StbM_00140 |
| SRS_BSW_00326 | - | SWS_StbM_00140 |

| | | |
|---|---|---|
| SRS_BSW_00327 | Error values naming convention | SWS_StbM_00041 |
| SRS_BSW_00328 | All AUTOSAR Basic Software Modules shall avoid the duplication of code | SWS_StbM_00140 |
| SRS_BSW_00330 | It shall be allowed to use macros instead of functions where source code is used and runtime is critical | 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_00140 |
| 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 |
| SRS_BSW_00339 | Reporting of production relevant error status | SWS_StbM_00058, SWS_StbM_00065 |
| SRS_BSW_00341 | Module documentation shall contains all needed informations | SWS_StbM_00140 |
| SRS_BSW_00342 | It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed | SWS_StbM_00140 |
| SRS_BSW_00344 | BSW Modules shall support link-time configuration | SWS_StbM_00140 |
| SRS_BSW_00347 | A Naming seperation of different instances of BSW drivers shall be in place | SWS_StbM_00140 |
| SRS_BSW_00353 | All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header | SWS_StbM_00140 |
| SRS_BSW_00355 | - | 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_00359 | All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible | SWS_StbM_00140 |
| SRS_BSW_00360 | AUTOSAR Basic Software Modules callback functions are allowed to have parameters | SWS_StbM_00140 |
| SRS_BSW_00361 | All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header | SWS_StbM_00140 |
| SRS_BSW_00370 | - | SWS_StbM_00140 |
| SRS_BSW_00371 | The passing of function pointers as | SWS_StbM_00140 |

| | | |
|---|---|---|
| | API parameter is forbidden for all AUTOSAR Basic Software Modules | |
| SRS_BSW_00375 | Basic Software Modules shall report wake-up reasons | SWS_StbM_00140 |
| SRS_BSW_00376 | - | SWS_StbM_00057 |
| SRS_BSW_00378 | AUTOSAR shall provide a boolean type | SWS_StbM_00140 |
| SRS_BSW_00380 | Configuration parameters being stored in memory shall be placed into separate c-files | 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 |
| 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 |
| SRS_BSW_00387 | The Basic Software Module specifications shall specify how the callback function is to be implemented | SWS_StbM_00140 |
| 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_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 | The init function may have parameters | SWS_StbM_00052 |
| 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_00425 | The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects | SWS_StbM_00057 |
| 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_00436 | - | SWS_StbM_00065 |
| 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_00441 | Naming convention for type, macro and function | SWS_StbM_00140, SWS_StbM_00151 |
| SRS_BSW_00442 | The AUTOSAR architecture shall support standardized debugging and tracing features | SWS_StbM_00076 |
| SRS_BSW_00449 | BSW Service APIs used by Autosar Application Software shall return a Std_ReturnType | SWS_StbM_00053, SWS_StbM_00054, SWS_StbM_00055 |
| SRS_BSW_00453 | BSW Modules shall be harmonized | SWS_StbM_00140 |

| | | |
|---|---|---|
| SRS_BSW_00455 | - | SWS_StbM_00140 |
| SRS_StbM_20001 | The configuration of the Synchronized Time-Base Manager shall allow the interaction with different types of customers | SWS_StbM_00020, SWS_StbM_00025, SWS_StbM_00026, SWS_StbM_00028, SWS_StbM_00029, SWS_StbM_00037, SWS_StbM_00038, SWS_StbM_00082 |
| SRS_StbM_20002 | The Synchronized Time-Base Manager shall trigger registered customers | SWS_StbM_00020, SWS_StbM_00022, SWS_StbM_00077, SWS_StbM_00083, SWS_StbM_00147 |
| SRS_StbM_20003 | The Synchronized Time-Base Manager shall allow customers to have access to the synchronized time-base | SWS_StbM_00025, SWS_StbM_00026, SWS_StbM_00028, SWS_StbM_00029, SWS_StbM_00082, SWS_StbM_00145, SWS_StbM_00146 |
| SRS_StbM_20005 | The Synchronized Time-Base Manager shall access the time-base provider/providers | SWS_StbM_00015, SWS_StbM_00050, SWS_StbM_00080, SWS_StbM_00081 |
| SRS_StbM_20006 | The Synchronized Time-Base Manager shall continuously provide a dependable provision of time | SWS_StbM_00050 |
| SRS_StbM_20007 | The Synchronized Time-Base Manager shall provide fault detection mechanism | SWS_StbM_00031, SWS_StbM_00032, SWS_StbM_00033, SWS_StbM_00034, SWS_StbM_00036, SWS_StbM_00151 |
| SRS_StbM_20008 | The Synchronized Time-Base Manager shall provide the functionality of notifying customers | SWS_StbM_00037, SWS_StbM_00038, SWS_StbM_00148 |
| SRS_StbM_20009 | The module shall provide configuration parameters for each triggered customer | SWS_StbM_00084, SWS_StbM_00085 |

# 7 Functional specification

## 7.1 Background & Rationale

The overall objective is to provide synchronized time-bases in safety-related systems built on AUTOSAR technology.

In the AUTOSAR context, the need for a synchronized time-base comes from the AUTOSAR Time Determinism concept. If AUTOSAR compliant application software must execute with a predictable timing (as the time determinism concept requires), a suitable common time definition must be used. The Synchronized Time-Base Manager is responsible for the propagation of such a time definition.

## 7.2 Overview

The purpose of the Synchronized Time-Base Manager is to provide synchronized time-bases for AUTOSAR software. Based on the use cases discussed in section 1.1, the Synchronized Time-Base Manager shall provide the following functionality:

- **Access to synchronized time-base**:

  As it is not in the scope of the Synchronized Time-Base Manager to introduce new time agreement protocols, the Synchronized Time-Base Manager uses the existing facilities in the AUTOSAR layered software architecture (see [2] for more details). Currently, the Synchronized Time-Base Manager synchronized time-bases propagated by FlexRay and TTCAN clusters are supported.

- **Provision of synchronized time-bases:**

  As described in section 1.4, the Synchronized Time-Base Manager has well-defined *customers* (e.g. the OS), which require the functionality of the module. For the provision of the synchronized time-bases, two general approaches are eligible. On the one hand, the Synchronized Time-Base Manager triggers the customer (e.g. by synchronizing the OS ScheduleTable). On the other hand, the Synchronized Time-Base Manager must provide an appropriate API that allows *customers* to have direct access to the module (e.g. when asking for the current calendar time).

- **Notification mechanism**:

  Whenever state changes occur or the Synchronized Time-Base Manager detects incorrect behavior, registered notification *customers* (e.g. application, see 1.4) must be informed about those circumstances.

In Figure 1, the arrows 3 and 4 show the communication caused by the provision of time-bases and the notification mechanism. In addition, arrows 1 and 2 depict the access of the Synchronized Time-Base Manager to synchronized time-bases.

The rest of this chapter provides a deeper look into the previously mentioned functionality of the Synchronized Time-Base Manager.

## 7.3 Implementation requirements

**[SWS_StbM_00091]** ⌈
The interfaces of the Synchronized Time-Base Manager shall not be implemented as macros, unless all interfacing modules are defined to be pre-compile. ⌋ ( )

**[SWS_StbM_00068]** ⌈
The header file *StbM.h* shall contain a software and specification version number. ⌋ (SRS_BSW_00004)

## 7.4 Clock time formats

**[SWS_StbM_00006]** ⌈
The Synchronized Time-Base Manager shall provide an abstract time format to its *customers*. ⌋ ( )

**[SWS_StbM_00078]** ⌈
The abstract time format provided by the Synchronized Time-Base Manager shall have the notion of ticks. ⌋ ( )

**[SWS_StbM_00079]** ⌈
In addition to the abstract time format, the Synchronized Time-Base Manager shall allow access to the respective tick duration, which defines the duration of one tick in microseconds. ⌋ ( )

The tick value in addition to the tick resolution defines the value of the respective time-base.

**[SWS_StbM_00135]** ⌈
The SynchronizedTimeBaseManager should provide an absolute time format to its customers. The absolute time format shall consist of a tick value and the number of overflows of the tick value counter. ⌋ ( )

Taking the overflows into account an absolute time can be determined.

## 7.5 Startup behavior

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

### 7.5.1 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.

**[SWS_StbM_00012]** ⌈
Required basic software modules for the Synchronized Time-Base Manager must be available (running) before the Synchronized Time-Base Manager accesses them. ⌋ ( )

### 7.5.2 Initialization

**[SWS_StbM_00013]** ⌈
The Synchronized Time-Base Manager shall establish the initial state of the module (initial state: No synchronization time-base available), preparing the module for the actual functionality of providing synchronized time-base to the *customers*. ⌋ ( )

## 7.6 Shutdown behavior

None.

## 7.7 Normal operation

### 7.7.1 Access to synchronized time-bases

As stated in section 4.1.2, the Synchronized Time-Base Manager does not provide any time agreement protocol. Such a protocol has the functionality of establishing a synchronized time-base among arbitrary members (e.g. ECUs).

**[SWS_StbM_00015]** ⌈
The Synchronized Time-Base Manager shall provide an adapter mechanism, which allows making the access to the time-base (e.g. FlexRay time-base) configurable. ⌋ (SRS_StbM_20005)

The Synchronized Time-Base Manager requires the following information of each time-base:

1. The *time-base value* is represented by the notion of ticks + tick duration.
2. The *sync state* indicates if the time-base is synchronized or not

Note:
The adapter will be triggered by the StbM via callouts (see section 8.1.5.3 for more details about the callout signature).

**[SWS_StbM_00080]** ⌈
The Synchronized Time-Base manager shall be independent of underlying *providers*. ⌋ (SRS_StbM_20005)

For each *provider*, a respective adapter shall be available which translates the provider specific information (time-base value + time-base state) into the abstract representation required by the Synchronized Time-Base Manager.

**[SWS_StbM_00081]** ⌈
Time-base *providers* shall be called periodically by the Synchronized Time-Base Manager. ⌋ (SRS_StbM_20005)

**[SWS_StbM_00050]** ⌈
In case the Synchronized Time-Base Manager cannot acquire the synchronized time-base from the *providers* (e.g. because a temporary FlexRay synchronization loss has been occurred), the module must be able to maintain the time-base autonomously. In this case, the interested *customers* must be informed about the

state change (see 7.9 for more information about the notification mechanism). ⌋ (SRS_StbM_20005, SRS_StbM_20006)

If the mentioned autonomous maintenance of the time-base is required, one and only one timebase provider would have to be configured to have access to the local time, i.e., for that provider STBM_LOCAL_TIME_REF (see section 10.2) would reference an OS counter.

In case synchronization fails for any of the synchronized timebases, a customer of that timebase can decide to switch to the mentioned local timebase (provider). In that case the local time replaces the (currently not available) time from the synchronized timebase. Any conversion from local to synchronized time, which is required by the customer while the timebase is not synchronized, would have to be performed by the customer (application) itself.

In addition, each *triggered customer* can specify via the configuration parameter STBM_TRIGGER_IN_SYNCSTATE (see section 10.2), if he wants to be triggered when only the local time is available.

### 7.7.2 Provision of synchronized time-bases

**[SWS_StbM_00019]** ⌈
The Synchronized Time-Base Manager shall use the concept of “Sync by absolute time value”.

Thus, the module itself does inform the *customers* about the absolute **definition of time** (e.g. 454 ticks). This is in contrast to “Sync by relative time value”, where the *customer* will be informed about the elapsed time **since** the last update. ⌋ ( )

**[SWS_StbM_00082]** ⌈
The Synchronized Time-Base Manager shall provide the following information to its *customers*:

1. The *time-base value* is represented by the notion of ticks + tick duration.
2. The *sync state* indicates if the time-base is synchronized or not⌋ (SRS_StbM_20001, SRS_StbM_20003)

**[SWS_StbM_00136]** ⌈
The Synchronized Time-Base Manager shall provide the following information to its *customers*:
The time-base absolute value is represented by the notion of ticks + tick duration + number of overflows⌋ ( )

**[SWS_StbM_00137]** ⌈
On every overflow of the tick value counter, the counter systemTicks shall be incremented by one. ⌋ ( )

**[SWS_StbM_00084]** ⌈
Customers of type *triggered customer* shall be invoked periodically by the Synchronized Time-Base Manager. ⌋ (SRS_StbM_20009)

**[SWS_StbM_00093]** ⌈
The triggering period shall be configurable for each *triggered customer*. ⌋ ( )

**[SWS_StbM_00083]** ⌈
Customers of type *triggered customer* will be invoked by the Synchronized Time-Base Manager by providing the tick value of the respective time-base. ⌋ (SRS_StbM_20002)

**[SWS_StbM_00085]** ⌈
Each customer of type *triggered customer* shall have the possibility to specify whether he wants to be invoked by the Synchronized Time-Base Manager in case the required time-base is not synchronized. ⌋ (SRS_StbM_20009)

**[SWS_StbM_00020]** ⌈
The Synchronized Time-Base Manager must 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)

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

**[SWS_StbM_00022]** ⌈
The Synchronized Time-Base Manager shall provide a means to configure the time-base to which the OS ScheduleTable should be synchronized. ⌋ (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 ask the OS for the status of the OS ScheduleTable before performing the synchronization. ⌋ (SRS_BSW_00429)

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).

**[SWS_StbM_00025]** ⌈
The Synchronized Time-Base Manager shall provide an interface for accessing the synchronized time-base. ⌋ (SRS_StbM_20001, SRS_StbM_20003)

In this case, the information exchange is triggered by the *active customer*, but not by the Synchronized Time-Base Manager itself. The calling customer specifies the required synchronized time-base.

**[SWS_StbM_00026]** ⌈
The API for accessing the synchronized time-bases shall be used by application software components (see [7]) as well as other BSW modules (see [1]). ⌋ (SRS_StbM_20001, SRS_StbM_20003)

**[SWS_StbM_00028]** ⌈
For the interaction with application software components, standardized AUTOSAR interfaces shall be specified. ⌋ (SRS_StbM_20001, SRS_StbM_20003)

**[SWS_StbM_00029]** ⌈
For the interaction with other BSW modules, respective interfaces (see 8 for information about the interface definition) shall be available. ⌋ (SRS_StbM_20001, SRS_StbM_20003)

### 7.7.3 Plausibility check

**[SWS_StbM_00031]** ⌈
The Synchronized Time-Base Manager shall monitor the cyclic execution of the StbM_Mainfunction (see section 8.1.4) for updating the status and values of the synchronized time-bases. ⌋ (SRS_StbM_20007)

## 7.8 Fault detection

**[SWS_StbM_00032]** ⌈
The Synchronized Time-Base Manager monitors its own triggering rate to guarantee a continuous time-base recalculation. ⌋ (SRS_StbM_20007)

**[SWS_StbM_00033]** ⌈
The Synchronized Time-Base Manager shall detect the loss of synchronized time-bases (e.g. because the FlexRay time agreement protocol can not retrieve a synchronized time-base in the network cluster). ⌋ (SRS_StbM_20007)

**[SWS_StbM_00034]** ⌈
The Synchronized Time-Base Manager shall detect the (re-) establishment of synchronized time-bases (e.g. because the FlexRay time agreement protocol has re-established the synchronized time-base in the network cluster). ⌋ (SRS_StbM_20007)

**[SWS_StbM_00036]** ⌈
The Synchronized Time-Base Manager shall detect errors during the invocation of *customers* (e.g. OS). ⌋ (SRS_StbM_20007)

## 7.9 Notification mechanism

**[SWS_StbM_00037]** ⌈
The Synchronized Time-Base Manager shall provide notifications for *customers* that want to be informed about status changes detected by the Synchronized Time-Base Manager. The data type defined in section 8.1.2.1 classifies the different states. ⌋ (SRS_StbM_20001, SRS_StbM_20008)

**[SWS_StbM_00038]** ⌈
The Synchronized Time-Base Manager shall support application software components (see [7]) as well as other BSW modules (see [1]) as potential *customers* for the notification mechanism. ⌋ (SRS_StbM_20001, SRS_StbM_20008)

**[SWS_StbM_00076]** ⌈
The status detected by the Synchronized Time-Base Manager and defined in section 8.1.2.1 shall be available for debugging. ⌋ (SRS_BSW_00442)

## 7.10 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 |
| API request integrity failed | Production | STBM_E_INTEGRITY_FAILED | Assigned by DEM |
| API request failed | Production | STBM_E_REQ_FAILED | Assigned by DEM |
| 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)

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.

## 7.11 Error detection

**[SWS_StbM_00101]** ⌈
If development error detection is enabled for this module (see 10.2), the following table specifies which DET error values shall be reported for each API call:

| ***API call*** | ***Error condition*** | ***DET related error value*** |
|---|---|---|
| StbM_Init | StbM initialization failed | STBM_E_INIT_FAILED |
| StbM_GetVersionInfo | Invalid pointer in parameter list | STBM_E_PARAM_POINTER |
| Stbm_GetSyncState | API requests called with wrong parameter | STBM_E_PARAM |
| | StbM is not yet initialized | STBM_E_NOT_INITIALIZED |
| | Invalid pointer in parameter list | STBM_E_PARAM_POINTER |
| StbM_GetGlobalTime | API requests called with wrong parameter | STBM_E_PARAM |
| | StbM is not yet initialized | STBM_E_NOT_INITIALIZED |
| | Invalid pointer in parameter list | STBM_E_PARAM_POINTER |
| StbM_GetTickDuration | API requests called with wrong parameter | STBM_E_PARAM |

| | | |
|---|---|---|
| | StbM is not yet initialized | STBM_E_NOT_INITIALIZED |
| StbM_GetAbsoluteTime | API requests called with wrong parameter | STBM_E_PARAM |
| | StbM is not yet initialized | STBM_E_NOT_INITIALIZED |
| | Invalid pointer in parameter list | STBM_E_PARAM_POINTER |

⌋ ( )

## 7.12 Error notification

For details refer to the chapter 7.3 “Error Detection” in *SWS_BSWGeneral.*

## 7.13 Production Errors

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

## 7.14 Extended Production Errors

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

## 7.15 Version checking

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

## 7.16 AUTOSAR Service Interface

### 7.16.1 Architecture

In the AUTOSAR ECU Architecture (see [2]) the “Synchronized Time-base Manager” BSW module implements an AUTOSAR Service as indicated in Figure 2.

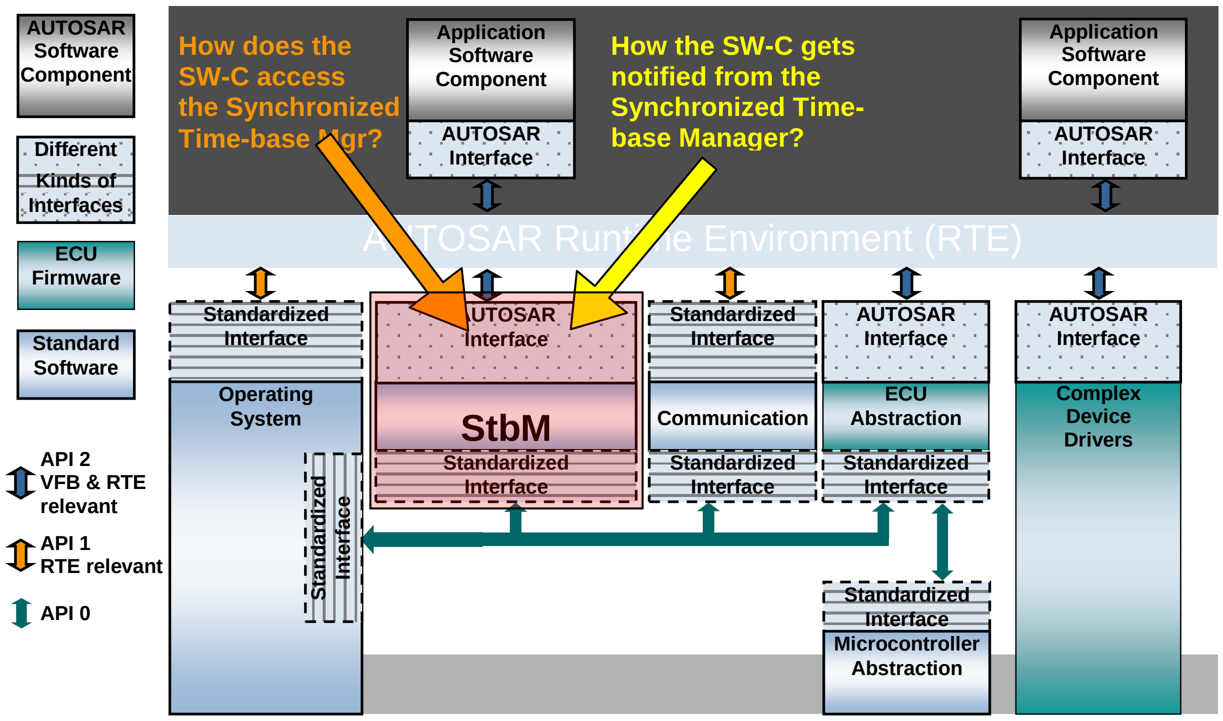


**Figure 2: The Synchronized Time-Base Manager in the ECU architecture**

From the viewpoint of the BSW module “Synchronized Time-Base Manager”, there are two kinds of dependencies between the service and application software components:

- The application accesses the Synchronized Time-Base Manager
- The application is optionally triggered with the value and/or notified upon state changes of the associated time-base. This invocation is initiated by the Synchronized Time-Base Manager.

These dependencies must be 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 StbM Service.

### 7.16.2 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.

### 7.16.3 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, thus the interaction with the Synchronized Time-Base Manager via the RTE shall be done in the following use cases:

- **Application software component accesses the Synchronized Time-Base Manager:** An “active” customer autonomously calls the Synchronized Time-Base Manager, getting knowledge about the definition of time and the state of the module.
- **Synchronized Time-Base Manager notifies application software component:** The Synchronized Time-Base Manager informs the (“triggered” / “notification”) customer about the current time and state changes and/or error occurrences (e.g. the synchronisation state of a time-base has changed).

### 7.16.4 Specification of the Ports and Ports Interfaces

This chapter specifies the ports and port interfaces which are needed in order to operate the Synchronized Time-Base Manager functionality over 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.

#### 7.16.4.1 Ports and Port Interfaces for accessing the service

The Port Interfaces for “active” customers to access the service are implemented as Client / Server interfaces (refer to 8.2.2). The corresponding ports are described in ch. 8.2.4.1 and 8.2.4.2

**[SWS_StbM_00127]** ⌈
Each software component must provide exactly one Port for each synchronized time-base it wants to access. ⌋ ( )

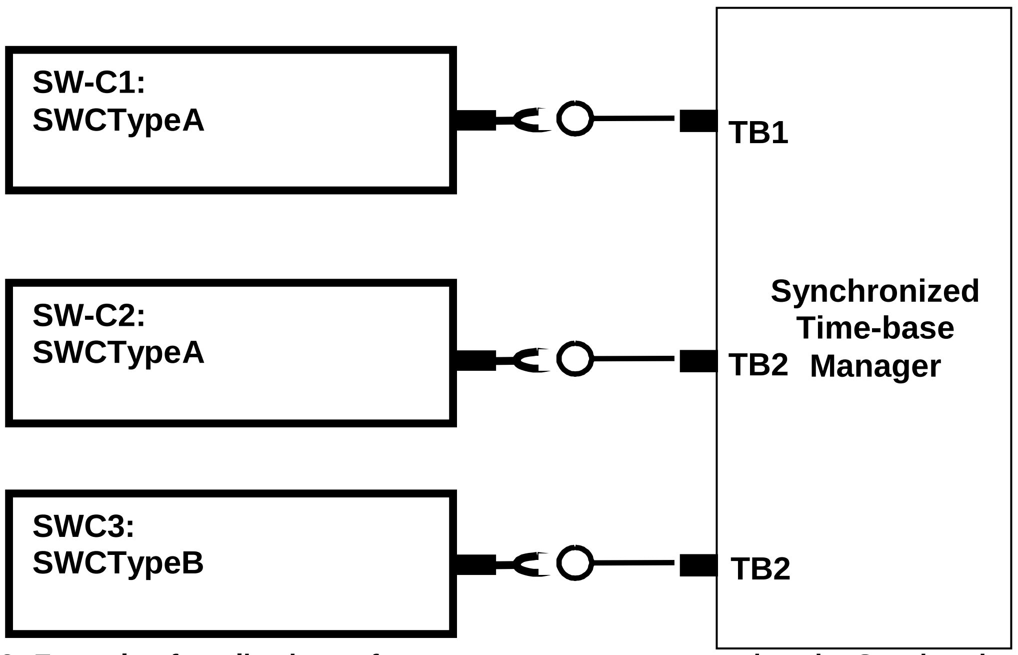


**Figure 3: Example of application software components connected to the Synchronized Time-Base Manager via service ports. On the left side, there are two instances of component SWCTypeA and one instance of component SWCTypeB. The Port names on the right side define the requested synchronized time-base. 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[2].

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. 8.1.3.3 to 8.1.3.6 for relevant API calls).

#### 7.16.4.2 Ports and Port Interfaces for application triggering and notification

As depicted in section 1.4, some application components may want to get informed by the Synchronized Time-Base Manager about the current time and when state changes occur. Thus, some kind of trigger / notification mechanism is required. The Synchronized Time-Base Manager uses this mechanism whenever such information should be forwarded to interested application software components.

The trigger / notification mechanism is implemented as Sender / Receiver interfaces (refer to 8.2.2). The corresponding ports are given in ch. ch. 8.2.4.3 and 8.2.4.2.

Note:

[2] 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.

There is no need to submit the respective time-base (e.g. as parameter by using a C/S interface instead of an S/R interface). 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.

**[SWS_StbM_00130]** ⌈In analogy to the Port definition for above, each application software component must provide:

- One Port for each synchronized time-base, about whose time value he wants to get informed periodically.
- One Port for each synchronized time-base, about whose state changes he wants to get informed. ⌋ ( )

### 7.16.5 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 7.16.6 for how the required synchronized time-base can be documented by augmenting the definition of Ports (feature "port defined argument value").

**[SWS_StbM_00131]** ⌈
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 GetSyncState
            symbol "StbM_GetSyncState"
            canbeInvokedConcurrently = TRUE

      RunnableEntity GetGlobalTime
            symbol "StbM_GetGlobalTime"
            canbeInvokedConcurrently = TRUE

      RunnableEntity GetTickDuration
            symbol "StbM_GetTickDurartion"
            canbeInvokedConcurrently = TRUE

      RunnableEntity GetAbsoluteTime
            symbol "StbM_GetAbsoluteTime"
            canbeInvokedConcurrently = TRUE
      // end of section "runnable entities"

};
```
⌋ ( )

### 7.16.6 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 |

⌋ ( )

### 8.1.2 Type definitions

#### 8.1.2.1 StbM_SyncStatusType

**[SWS_StbM_00141]⌈**

| | | |
|---|---|---|
| ***Name:*** | StbM_SyncStatusType | |
| ***Type:*** | Enumeration | |
| ***Range:*** | STBM_STATE_SYNC | The time-base is synchronized |
| | STBM_STATE_NOT_SYNC | The time-base is not synchronized |
| ***Description:*** | Variables of this type are used to represent the status of the synchronized time-base. | |

⌋()

#### 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. | | |

⌋()

#### 8.1.2.3 StbM_SystemTimeType

**[SWS_StbM_00143]⌈**

| | | | |
|---|---|---|---|
| ***Name:*** | StbM_SystemTimeType | | |
| ***Type:*** | Structure | | |
| ***Element:*** | StbM_TickType | ticks | number of ticks |
| | uint16 | tickDuration | duration of one tick in microseconds |
| | uint32 | systemTicks | number of overflows |
| ***Description:*** | Variables of this type are used for expressing long time intervals, e.g. time stamps, where the usage of StbM_TickTime would cause overflows. | | |

⌋()

#### 8.1.2.4 StbM_TickType

**[SWS_StbM_00144]⌈**

| | | | |
|---|---|---|---|
| ***Name:*** | StbM_TickType | | |
| ***Type:*** | uint16 | | |
| ***Range:*** | 0..{ecuc(StbM/StbMGeneral/StbMTickTypeRange)} | -- | -- |
| ***Description:*** | Variables of this type are used to represent the notion of ticks. The range of the type shall be configurable via the respective ECUC parameter "StbM_TickTypeRange". | | |

⌋()

### 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(`<br>`    Std_VersionInfoType* versioninfo`<br>`)` | |
| ***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). ⌋ ( )

#### 8.1.3.2 StbM_Init

**[SWS_StbM_00052]** ⌈

| | |
|---|---|
| ***Service name:*** | StbM_Init |
| ***Syntax:*** | `void StbM_Init(`<br>`    void`<br>`)` |
| ***Service ID[hex]:*** | 0x00 |
| ***Sync/Async:*** | Synchronous |
| ***Reentrancy:*** | Non Reentrant |
| ***Parameters (in):*** | None |
| ***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)

**[SWS_StbM_00097]** ⌈

The ECU State Manager shall call the function StbM_Init during the startup phase of the ECU in order to initialize the module. ⌋ ( )

The StbM is not functional until this function has been called.

**[SWS_StbM_00098]** ⌈
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 ⌋ ( )

**[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. ⌋ ( )

**[SWS_StbM_00121]** ⌈

StbM_Init shall set the static status variable to a value not equal to 0. ⌋ ( )

#### 8.1.3.3 StbM_GetSyncState

**[SWS_StbM_00053]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | StbM_GetSyncState | |
| ***Syntax:*** | `Std_ReturnType StbM_GetSyncState(`<br>`    StbM_SynchronizedTimeBaseType timeBaseID,`<br>`    StbM_SyncStatusType* syncState`<br>`)` | |
| ***Service ID[hex]:*** | 0x01 | |
| ***Sync/Async:*** | Synchronous | |
| ***Reentrancy:*** | Non Reentrant | |
| ***Parameters (in):*** | timeBaseID | The synchronized time-base, whose state is of interest. |
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | syncState | The synchronization state of the time-base. |
| ***Return value:*** | Std_ReturnType | E_OK : successful<br>E_NOT_OK : failed |
| ***Description:*** | Returns the current synchronization state of the submitted time-base. | |

⌋ (SRS_BSW_00449)

**[SWS_StbM_00102]** ⌈
The function StbM_GetSyncState shall return by function parameter the current synchronization state of the submitted time-base. ⌋ ( )

Regarding error detection, the requirement SWS_StbM_00101 is applicable to the function StbM_GetSyncState.

**[SWS_StbM_00116]** ⌈

If an error occurs during the execution of the function StbM_GetSyncState, all out parameters shall stay untouched. ⌋ ( )

### 8.1.3.4 StbM_GetGlobalTime

**[SWS_StbM_00054]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | StbM_GetGlobalTime | |
| ***Syntax:*** | `Std_ReturnType StbM_GetGlobalTime( StbM_SynchronizedTimeBaseType timeBaseID, StbM_TickType* ticks )` | |
| ***Service ID[hex]:*** | 0x02 | |
| ***Sync/Async:*** | Synchronous | |
| ***Reentrancy:*** | Non Reentrant | |
| ***Parameters (in):*** | timeBaseID | The synchronized time-base, whose time definition is of interest. |
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | ticks | The current definition of time represented by the notion of ticks. |
| ***Return value:*** | Std_ReturnType | E_OK : successful<br>E_NOT_OK : failed |
| ***Description:*** | Returns the current time of the submitted synchronized time-base. The time is represented by the notion of ticks. | |

⌋ (SRS_BSW_00449)

**[SWS_StbM_00103]** ⌈
The function StbM_GetGlobalTime shall return by function parameter
the current time of the submitted time-base. ⌋ ( )

Regarding error detection, the requirement SWS_StbM_00101 is applicable to the function StbM_GetGlobalTime.

**[SWS_StbM_00117]** ⌈
If an error occurs during the execution of the function StbM_GetGlobalTime, all out parameters shall stay untouched. ⌋ ( )

### 8.1.3.5 StbM_GetTickDuration

**[SWS_StbM_00055]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | StbM_GetTickDuration | |
| ***Syntax:*** | `Std_ReturnType StbM_GetTickDuration( StbM_SynchronizedTimeBaseType timeBaseID, uint16* tickDuration )` | |
| ***Service ID[hex]:*** | 0x03 | |
| ***Sync/Async:*** | Synchronous | |
| ***Reentrancy:*** | Non Reentrant | |
| ***Parameters (in):*** | timeBaseID | The synchronized time-base, whose tick duration is of interest. |
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | tickDuration | Number of microseconds of one tick. |

| | | |
|---|---|---|
| ***Return value:*** | Std_ReturnType | E_OK : successful<br>E_NOT_OK : failed |
| ***Description:*** | Returns the duration of one time-base tick in microseconds. | |

⌋ (SRS_BSW_00449)

**[SWS_StbM_00104]** ⌈
The function StbM_GetTickDuration shall return by function parameter the current time of the submitted time-base. ⌋ ( )

Regarding error detection, the requirement SWS_StbM_00101 is applicable to the function StbM_GetTickDuration.

**[SWS_StbM_00118]** ⌈
If an error occurs during the execution of the function StbM_GetTickDuration, all out parameters shall stay untouched. ⌋ ( )

### 8.1.3.6 StbM_GetAbsoluteTime

| | | |
|---|---|---|
| ***Service name:*** | StbM_GetAbsoluteTime | |
| ***Syntax:*** | `Std_ReturnType StbM_GetAbsoluteTime(`<br>`    StbM_SynchronizedTimeBaseType timeBaseID,`<br>`    StbM_SystemTimeType* systemTime`<br>`)` | |
| ***Service ID[hex]:*** | 0x06 | |
| ***Sync/Async:*** | Synchronous | |
| ***Reentrancy:*** | Non Reentrant | |
| ***Parameters (in):*** | timeBaseID | The synchronized time-base, whose state is of interest. |
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | systemTime | absolute time stamp |
| ***Return value:*** | Std_ReturnType | E_OK : successful<br>E_NOT_OK : failed |
| ***Description:*** | Returns an absolute time value relative to the start of the ECU. | |

**[SWS_StbM_00133]** ⌈
The function StbM_GetAbsoluteTime shall return by function parameter the current time of the submitted time-base plus the number of timer overflows. ⌋ ( )

Regarding error detection, the requirement SWS_StbM_00101 is applicable to the function StbM_GetAbsoluteTime.

**[SWS_StbM_00134]** ⌈
If an error occurs during the execution of the function StbM_GetAbsoluteTime, all out parameters shall stay untouched. ⌋ ( )

### 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 Scheduler.<br><br>The StbM_MainFunction invokes each time-base provider, gathering the required information (sync status, tick value, tick duration).<br>In addition, each triggered customer will be invoked, providing the status and tick value of the associated time-base. |

⌋(SRS_BSW_00425, SRS_BSW_00376)

**[SWS_StbM_00105]** ⌈
The function StbM_MainFunction shall perform the time-base *provider* callout. ⌋ ( )

**[SWS_StbM_00106]** ⌈
In case of state changes of associated time-bases, the function StbM_MainFunction shall perform the triggered customer sync state callback. ⌋ ( )

**[SWS_StbM_00107]** ⌈
The function StbM_MainFunction shall perform the triggered customer time provision callback. ⌋ ( )

### 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.<br>OBD Events Suppression shall be ignored for this computation. |
| Det_ReportError | Service to report development errors. |

⌋ (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 currently running OS-Application (a unique identifier has to be allocated to each application). |
| 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. |

⌋ ( )

### 8.1.5.3 Configurable Interfaces

In this chapter all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of these kinds of interfaces are not fixed because they are configurable.

**[SWS_StbM_00108]** ⌈
For each synchronized time-base, respective call-out functions shall be configurable which provide the synchronization state SWS_StbM_00086, the tick value SWS_StbM_00087 and the tick duration SWS_StbM_00088 of the time-base. ⌋ ( )

**[SWS_StbM_00109]** ⌈
For each *triggered customer*, respective callback functions shall be configurable for the provision of the synchronization state SWS_StbM_00089 and the tick value SWS_StbM_00090. The Synchronized Time-Base Manager invokes the callback functions within the StbM_Mainfunction (see section 8.1.4). ⌋ ( )

#### 8.1.5.3.1 SyncStateProviderCallout

**[SWS_StbM_00086]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | SyncStateProviderCallout | |
| ***Syntax:*** | `void SyncStateProviderCallout(`<br>`    StbM_SynchronizedTimeBaseType timeBaseID,`<br>`    StbM_SyncStatusType* syncState`<br>`)` | |
| ***Sync/Async:*** | Synchronous | |
| ***Reentrancy:*** | Non Reentrant | |
| ***Parameters (in):*** | timeBaseID | The synchronized time-base, whose state is of interest. |
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | syncState | The synchronization state of the time-base. |
| ***Return value:*** | None | |
| ***Description:*** | The provider returns the current synchronization state of the time-base via callout. | |

⌋ ( )

**[SWS_StbM_00110]** ⌈
The call-out function SyncStateProviderCallout shall be called by the Synchronized Time-Base Manager for gathering the current synchronization state of the time-base.
⌋ ( )

#### 8.1.5.3.2 GlobalTimeProviderCallout

**[SWS_StbM_00087]** ⌈

| | |
|---|---|
| ***Service name:*** | GlobalTimeProviderCallout |
| ***Syntax:*** | `void GlobalTimeProviderCallout(`<br>`    StbM_SynchronizedTimeBaseType timeBaseID,` |

| | |
|---|---|
| | `StbM_TickType* ticks`<br>`)` |
| ***Sync/Async:*** | Synchronous |
| ***Reentrancy:*** | Non Reentrant |
| ***Parameters (in):*** | timeBaseID — The synchronized time-base, whose time definition is of interest. |
| ***Parameters (inout):*** | None |
| ***Parameters (out):*** | ticks — The current definition of time represented by the notion of ticks. |
| ***Return value:*** | None |
| ***Description:*** | The provider returns the current time of the synchronized time-base via callout. The time is represented by the notion of ticks. |

⌋ ( )

**[SWS_StbM_00111]** ⌈
The call-out function GlobalTimeProviderCallout shall be called by the Synchronized Time-Base Manager for gathering the current tick value of the time-base. ⌋ ( )

#### 8.1.5.3.3 TickDurationProviderCallout

**[SWS_StbM_00088]** ⌈

| | |
|---|---|
| ***Service name:*** | TickDurationProviderCallout |
| ***Syntax:*** | `void TickDurationProviderCallout(`<br>`    StbM_SynchronizedTimeBaseType timeBaseID,`<br>`    uint16* tickDuration`<br>`)` |
| ***Sync/Async:*** | Synchronous |
| ***Reentrancy:*** | Non Reentrant |
| ***Parameters (in):*** | timeBaseID — The synchronized time-base, whose state is of interest. |
| ***Parameters (inout):*** | None |
| ***Parameters (out):*** | tickDuration — The duration of one time-base tick in microseconds. |
| ***Return value:*** | None |
| ***Description:*** | The provider returns the duration of one time-base tick in microseconds via callout. |

⌋ ( )

**[SWS_StbM_00112]** ⌈
The call-out function TickDurationProviderCallout shall be called by the Synchronized Time-Base Manager for gathering the duration of one time-base tick. ⌋ ( )

#### 8.1.5.3.4 SyncStateCustomerCallbackFunction

**[SWS_StbM_00089]** ⌈

| | |
|---|---|
| ***Service name:*** | SyncStateCustomerCallbackFunction |
| ***Syntax:*** | `void SyncStateCustomerCallbackFunction(`<br>`    StbM_SynchronizedTimeBaseType timeBaseID,`<br>`    StbM_SyncStatusType syncState` |

| | | |
|---|---|---|
| | ) | |
| ***Sync/Async:*** | Synchronous | |
| ***Reentrancy:*** | Non Reentrant | |
| ***Parameters (in):*** | timeBaseID | The synchronized time-base, whose time definition is of interest. |
| | syncState | The synchronization state of the time-base. |
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | None | |
| ***Description:*** | The triggered customer will be triggered with the current synchronization state of the time-base via callback.<br>The callback will be executed whenever a state change occurs. | |

⌋ ( )

**[SWS_StbM_00113]** ⌈
The callback function SyncStateCustomerCallbackFunction shall be invoked by the Synchronized Time-Base Manager for providing the current synchronization state of the time-base to the *triggered customer*. ⌋ ( )

#### 8.1.5.3.5 GlobalTimeCustomerCallbackFunction

**[SWS_StbM_00090]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | GlobalTimeCustomerCallbackFunction | |
| ***Syntax:*** | `void GlobalTimeCustomerCallbackFunction(`<br>`    StbM_SynchronizedTimeBaseType timeBaseID,`<br>`    StbM_TickType ticks`<br>`)` | |
| ***Sync/Async:*** | Synchronous | |
| ***Reentrancy:*** | Non Reentrant | |
| ***Parameters (in):*** | timeBaseID | The synchronized time-base, whose time definition is of interest. |
| | ticks | The current definition of time represented by the notion of ticks. |
| ***Parameters (inout):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | None | |
| ***Description:*** | The triggered customer will be triggered with the current time-base value via callback.<br>The callback will be executed periodically by the StbM. | |

⌋ ( )

**[SWS_StbM_00114]** ⌈
The callback function GlobalTimeCustomerCallbackFunction shall be invoked by the Synchronized Time-Base Manager for providing the current tick value of the time-base to the *triggered customer*. ⌋ ( )

## 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 Sender-Receiver-Interfaces

#### 8.2.1.1 StbM_TimeBase_TriggerCustomer

**[SWS_StbM_00128]** ⌈In order to notify the application about changes/errors in the Synchronized Time-Base Manager, the following AUTOSAR standardized interface must be implemented:

| Name | StbM_TimeBase_TriggerCustomer | |
|---|---|---|
| Comment | DataElement which provides the current value of the synchronized time-base | |
| IsService | true | |
| Variation | -- | |
| Data Elements | ticks | |
| | Type | StbM_TickType |
| | Variation | -- |

| Name | StbM_TimeBase_StateNotification | |
|---|---|---|
| Comment | DataElement which provides the current state of the synchronized time-base | |
| IsService | true | |
| Variation | -- | |
| Data Elements | syncState | |
| | Type | StbM_SyncStatusType |

| | | |
|---|---|---|
| | Variation | -- |

⌋ ( )

**[SWS_StbM_00129]** ⌈The Synchronized Time-Base Manager will trigger the first interface (StbM_TimeBase_TriggerCustomer) periodically when the value of the respective time-base has been updated. The second interface will be triggered (StbM_TimeBase_StateNotification) whenever a state change is detected. ⌋ ( )

### 8.2.2 Client-Server-Interfaces

#### 8.2.2.1 StbM_AbsoluteTimeBaseValue

**[SWS_StbM_00138]** ⌈
In order to allow the application software components to access the **status,** the **value** and the number of **timer overflows** of the synchronized time-base, the Client-Server Interface must have the following properties:

| Name | StbM_AbsoluteTimeBaseValue | |
|---|---|---|
| Comment | -- | |
| IsService | true | |
| Variation | -- | |
| Possible Errors | 0 | E_OK |
| | 1 | E_NOT_OK |

Operations

| GetAbsoluteTime | | |
|---|---|---|
| Comments | access the absolute time since ECU start | |
| Variation | -- | |
| Parameters | systemTime | |
| | Comment | -- |
| | Type | StbM_SystemTimeType |
| | Variation | -- |
| | Direction | OUT |
| Possible Errors | E_OK | Operation successful |

| | | |
|---|---|---|
| | E_NOT_OK | Operation failed |
| | | |
| GetGlobalTime | | |
| Comments | access the current definition of time represented by the notion of ticks | |
| Variation | -- | |
| Parameters | ticks | |
| | Comment | -- |
| | Type | StbM_TickType |
| | Variation | -- |
| | Direction | OUT |
| Possible Errors | E_OK | Operation successful |
| | E_NOT_OK | Operation failed |
| | | |
| GetSyncState | | |
| Comments | access the current status of the synchronized time-base | |
| Variation | -- | |
| Parameters | syncState | |
| | Comment | -- |
| | Type | StbM_SyncStatusType |
| | Variation | -- |
| | Direction | OUT |
| Possible Errors | E_OK | Operation successful |
| | E_NOT_OK | Operation failed |
| | | |
| GetTickDuration | | |
| Comments | access the current definition of tickDuration | |
| Variation | -- | |
| Parameters | tickDuration | |
| | Comment | -- |
| | Type | uint16 |
| | Variation | -- |

| | | |
|---|---|---|
| | Direction | OUT |
| Possible Errors | E_OK | Operation successful |
| | E_NOT_OK | Operation failed |

⌋ ( )

#### 8.2.2.2 StbM_TimeBaseValue

**[SWS_StbM_00125]** ⌈
In order to allow the application software components to access the **status** and the **value** of the synchronized time-base, the Client-Server Interface must have the following properties:

| Name | StbM_TimeBaseValue | |
|---|---|---|
| Comment | -- | |
| IsService | true | |
| Variation | -- | |
| Possible Errors | 0 | E_OK |
| | 1 | E_NOT_OK |

Operations

| GetGlobalTime | | |
|---|---|---|
| Comments | access the current definition of time represented by the notion of ticks | |
| Variation | -- | |
| Parameters | ticks | |
| | Comment | -- |
| | Type | StbM_TickType |
| | Variation | -- |
| | Direction | OUT |
| Possible Errors | E_OK | Operation successful |
| | E_NOT_OK | Operation failed |
| | | |
| GetSyncState | | |
| Comments | access the current status of the synchronized time-base | |
| Variation | -- | |

| | | |
|---|---|---|
| Parameters | syncState | |
| | Comment | -- |
| | Type | StbM_SyncStatusType |
| | Variation | -- |
| | Direction | OUT |
| Possible Errors | E_OK | Operation successful |
| | E_NOT_OK | Operation failed |
| | | |
| GetTickDuration | | |
| Comments | access the current definition of tickDuration | |
| Variation | -- | |
| Parameters | tickDuration | |
| | Comment | -- |
| | Type | uint16 |
| | Variation | -- |
| | Direction | OUT |
| Possible Errors | E_OK | Operation successful |
| | E_NOT_OK | Operation failed |

⌋ ( )

**[SWS_StbM_00126]** ⌈
Using operation “GetSyncState()”, the application has the possibility to access the current status of the synchronized time-base. The other two operations of the interface are required for the access to the clock time: a) the tick value of the associated time-base (operation “GetTime()”), b) the duration of one tick in ms (operation “GetTickDuration()”).⌋ ( )

### 8.2.3 Implementation Data Types

This chapter specifies the data types which will be used in 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)

#### 8.2.3.1 StbM_TickType

[SWS_StbM_00149]⌈

| | | | |
|---|---|---|---|
| Name | StbM_TickType | | |
| Kind | Type | | |
| Derived from | uint16 | | |
| Description | Variables of this type are used to represent the notion of ticks. The range of the type shall be configurable via the respective ECUC parameter "StbM_TickTypeRange". | | |
| Range | 0..{ecuc(StbM/StbMGeneral/StbMTickTypeRange)} | | -- |
| Variation | -- | | |

⌋(SRS_BSW_00305)

#### 8.2.3.2 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)

#### 8.2.3.3 StbM_SyncStatusType

[SWS_StbM_00151]⌈

| | | | |
|---|---|---|---|
| Name | StbM_SyncStatusType | | |
| Kind | Enumeration | | |
| Range | STBM_STATE_SYNC | 0 | The time-base is synchronized |
| | STBM_STATE_NOT_SYNC | 1 | The time-base is not synchronized |

| | |
|---|---|
| Description | Variables of this type are used to represent the status of the synchronized time-base. |
| Variation | -- |

⌋(SRS_BSW_00305, SRS_BSW_00441, SRS_StbM_20007)

#### 8.2.3.4 StbM_SystemTimeType

[SWS_StbM_00152]⌈

| | | | |
|---|---|---|---|
| Name | StbM_SystemTimeType | | |
| Kind | Structure | | |
| Elements | ticks | StbM_TickType | number of ticks |
| | tickDuration | uint16 | duration of one tick in microseconds |
| | systemTicks | uint32 | number of overflows |
| Description | Variables of this type are used for expressing long time intervals, e.g. time stamps, where the usage of StbM_TickTime would cause overflows. | | |
| Variation | -- | | |

⌋(SRS_BSW_00305)

### 8.2.4 Ports

#### 8.2.4.1 StbM_ATB

[SWS_StbM_00145]⌈

| | | | |
|---|---|---|---|
| Name | ATB_{Name} | | |
| Kind | ProvidedPort | Interface | StbM_AbsoluteTimeBaseValue |
| Description | -- | | |
| Variation | Name = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)} | | |

⌋(SRS_StbM_20003)

#### 8.2.4.2 StbM_TB

[SWS_StbM_00146]⌈

| | | | |
|---|---|---|---|
| Name | TB_{Name} | | |
| Kind | ProvidedPort | Interface | StbM_TimeBaseValue |
| Description | -- | | |
| Variation | Name = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)} | | |

⌋(SRS_StbM_20003)

#### 8.2.4.3 StbM_TBTC

[SWS_StbM_00147]⌈

| | | | |
|---|---|---|---|
| Name | TBTC_{Name} | | |
| Kind | ProvidedPort | Interface | StbM_TimeBase_TriggerCustomer |
| Description | -- | | |
| Variation | Name = {ecuc(StbM/StbMSynchronizedTimeBase.SHORT-NAME)} | | |

⌋(SRS_StbM_20002)

#### 8.2.4.4 StbM_TBSN

[SWS_StbM_00148]⌈

| | | | |
|---|---|---|---|
| Name | TBSN | | |
| Kind | ProvidedPort | Interface | StbM_TimeBase_StateNotification |
| Description | -- | | |
| Variation | -- | | |

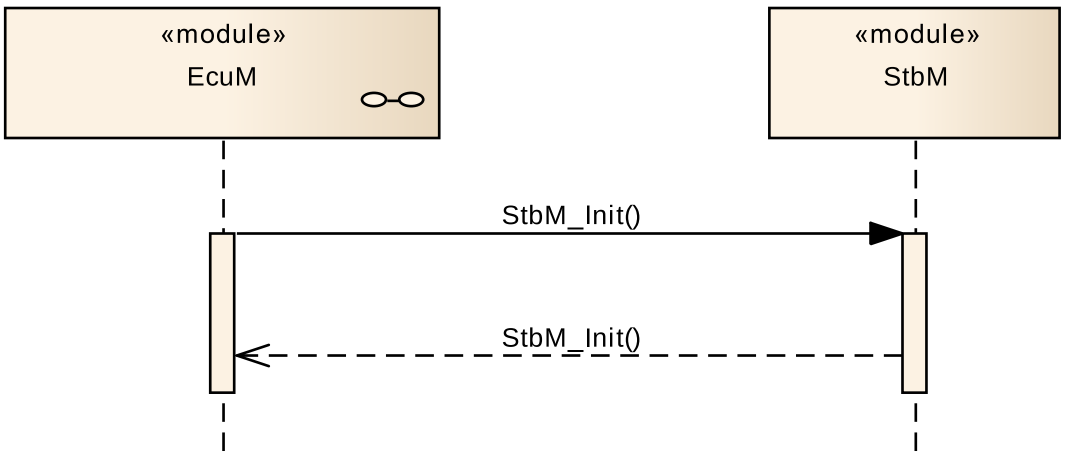
⌋(SRS_StbM_20008)

# 9 Sequence diagrams

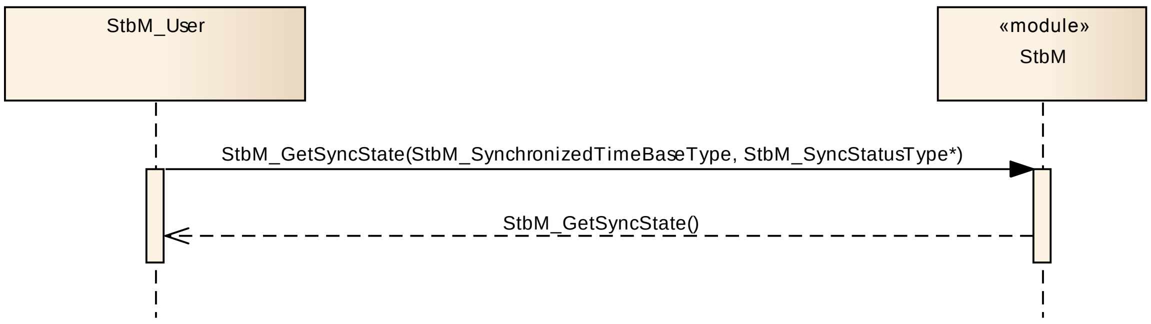
The sequence diagrams in this chapter show the basic operations of the Synchronized Time-Base Manager.

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

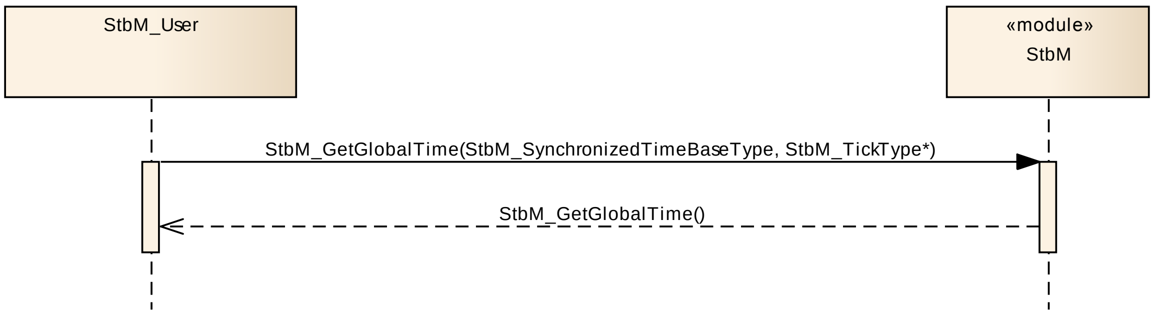
## 9.1 StbM_Init



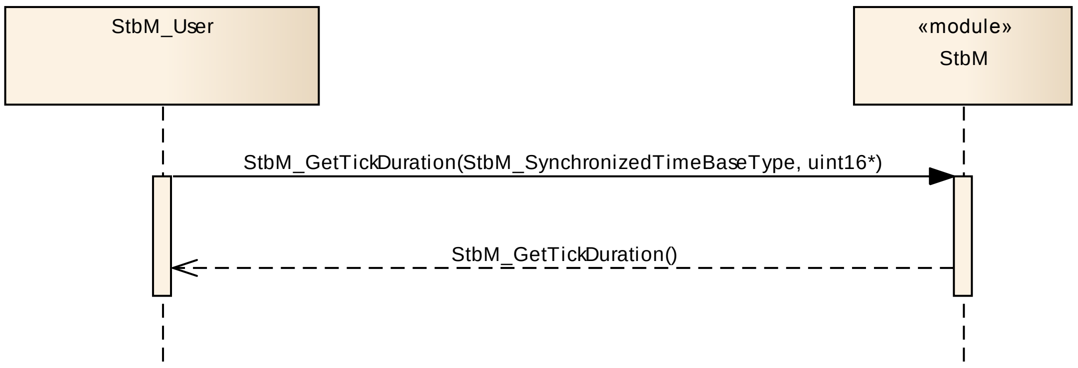
## 9.2 StbM_GetSyncState


StbM_User
«module»
StbM
StbM_GetSyncState(StbM_SynchronizedTimeBaseType, StbM_SyncStatusType*)
StbM_GetSyncState()

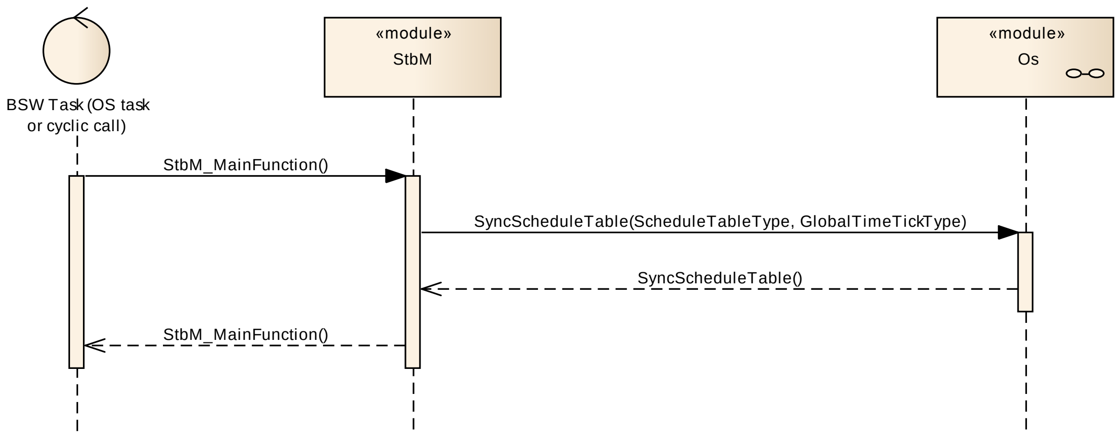
## 9.3 StbM_GetGlobalTime


StbM_User
«module»
StbM
StbM_GetGlobalTime(StbM_SynchronizedTimeBaseType, StbM_TickType*)
StbM_GetGlobalTime()

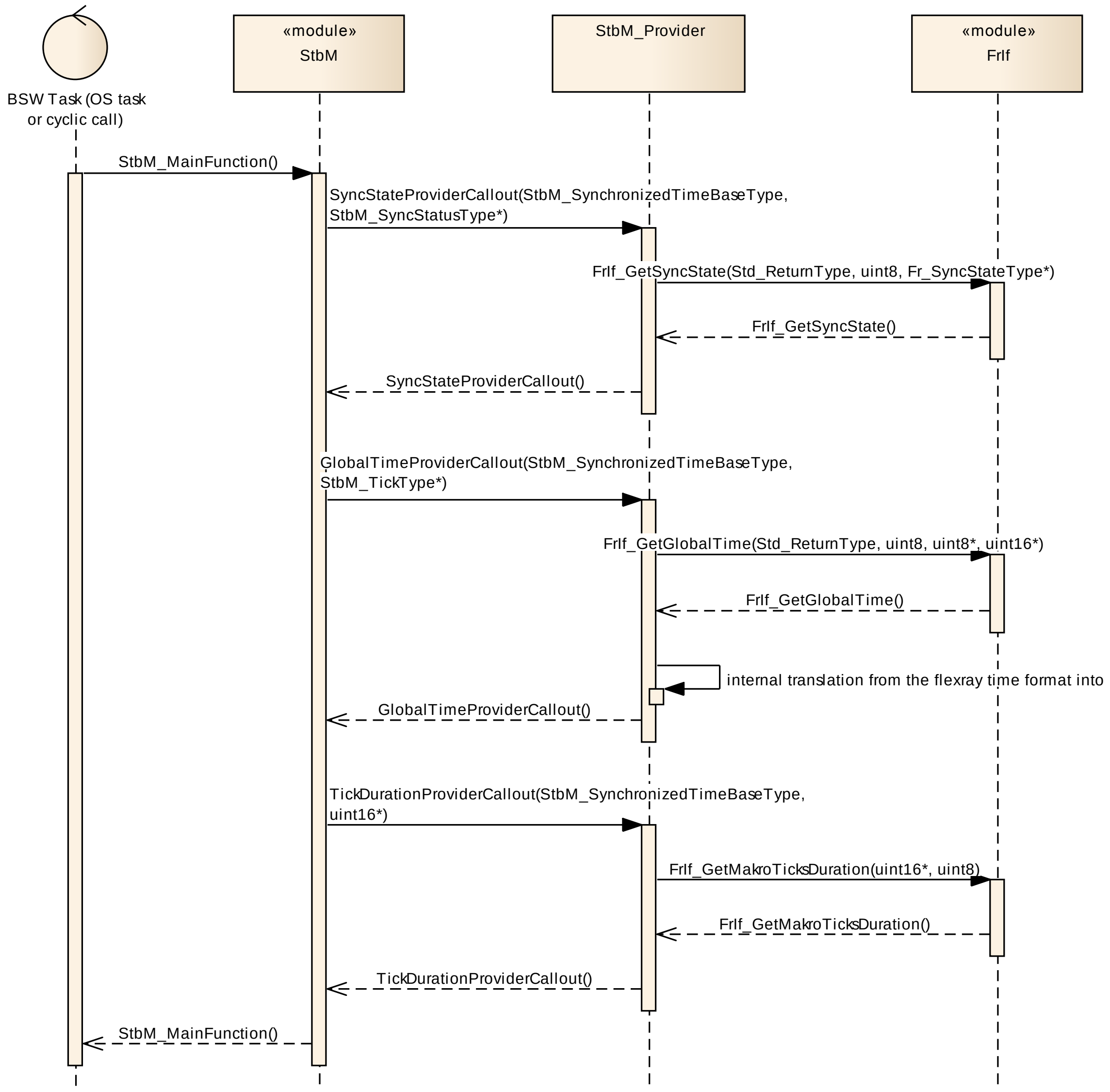
## 9.4 StbM_GetTTickDuration


StbM_User
«module»
StbM
StbM_GetTickDuration(StbM_SynchronizedTimeBaseType, uint16*)
StbM_GetTickDuration()

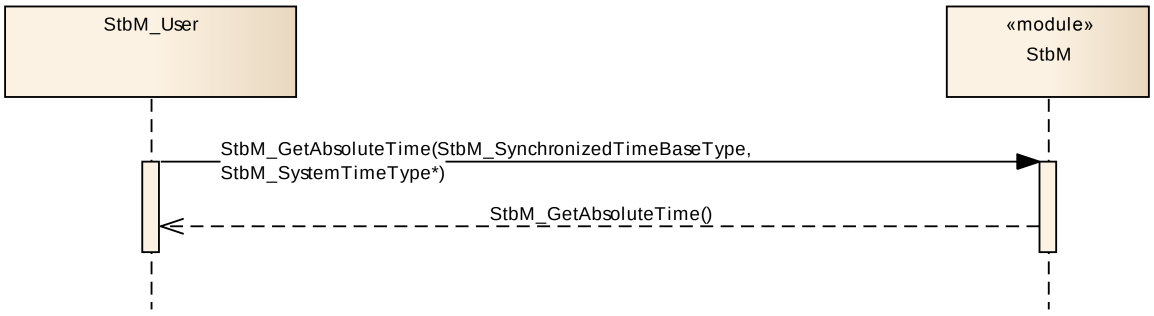
## 9.5 Explicit synchronization of OS ScheduleTable



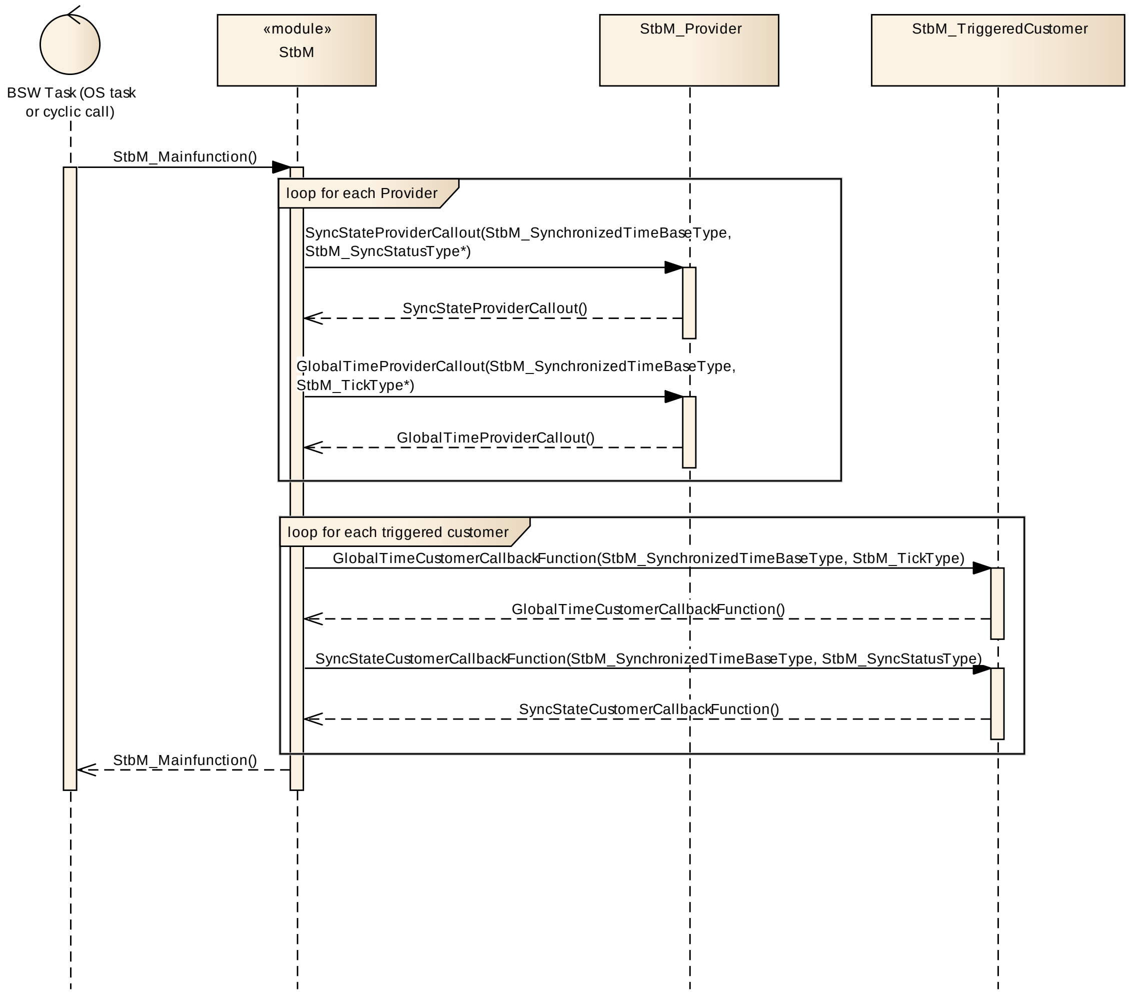
## 9.6 Provider callout with FlexRay


BSW Task (OS task or cyclic call)
«module» StbM
StbM_Provider
«module» FrIf
StbM_MainFunction()
SyncStateProviderCallout(StbM_SynchronizedTimeBaseType, StbM_SyncStatusType*)
FrIf_GetSyncState(Std_ReturnType, uint8, Fr_SyncStateType*)
FrIf_GetSyncState()
SyncStateProviderCallout()
GlobalTimeProviderCallout(StbM_SynchronizedTimeBaseType, StbM_TickType*)
FrIf_GetGlobalTime(Std_ReturnType, uint8, uint8*, uint16*)
FrIf_GetGlobalTime()
internal translation from the flexray time format into
GlobalTimeProviderCallout()
TickDurationProviderCallout(StbM_SynchronizedTimeBaseType, uint16*)
FrIf_GetMakroTicksDuration(uint16*, uint8)
FrIf_GetMakroTicksDuration()
TickDurationProviderCallout()
StbM_MainFunction()

## 9.7 StbM_GetAbsoluteTime


StbM_User
«module»
StbM
StbM_GetAbsoluteTime(StbM_SynchronizedTimeBaseType, StbM_SystemTimeType*)
StbM_GetAbsoluteTime()

## 9.8 StbM Cyclic Interaction with Provider and Triggered Customer


BSW Task (OS task or cyclic call)
«module»
StbM
StbM_Provider
StbM_TriggeredCustomer
StbM_Mainfunction()
loop for each Provider
SyncStateProviderCallout(StbM_SynchronizedTimeBaseType, StbM_SyncStatusType*)
SyncStateProviderCallout()
GlobalTimeProviderCallout(StbM_SynchronizedTimeBaseType, StbM_TickType*)
GlobalTimeProviderCallout()
loop for each triggered customer
GlobalTimeCustomerCallbackFunction(StbM_SynchronizedTimeBaseType, StbM_TickType)
GlobalTimeCustomerCallbackFunction()
SyncStateCustomerCallbackFunction(StbM_SynchronizedTimeBaseType, StbM_SyncStatusType)
SyncStateCustomerCallbackFunction()
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

**[SWS_StbM_00062]** ⌈
The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and Chapter 8. ⌋ (SRS_BSW_00159, SRS_BSW_00167)

**[SWS_StbM_00063]** ⌈
The configuration tool must check the consistency of the configuration at configuration time. ⌋ (SRS_BSW_00167)

### 10.2.1 Variants

### 10.2.2 StbM

| Module Name | StbM |
|---|---|
| Module Description | Configuration of the Synchronized Time-base Manager (StbM) module. |

| Included Containers | | |
|---|---|---|
| ***Container Name*** | ***Multiplicity*** | ***Scope / Dependency*** |
| StbMDemEventParameterRefs | 0..1 | Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references. |

| | | |
|---|---|---|
| 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 | 1..* | 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{STBM_GENERAL} |
| ***Description*** | This container holds the general parameters of the Synchronized Time-base Manager |
| ***Configuration Parameters*** | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_StbM_00026 :** | | |
| ***Name*** | StbMAbsoluteTimeApi {STBM_ABSOLUTE_TIME_API} | | |
| ***Description*** | Enables/Disables the StbM_GetAbsoluteTime API. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_StbM_00012 :** | | |
| ***Name*** | StbMDevErrorDetect {STBM_DEV_ERROR_DETECT} | | |
| ***Description*** | Switch for enabling the development error detection. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_StbM_00027 :** | | |
| ***Name*** | StbMMainFunctionPeriod {STBM_MAIN_FUNCTION_PERIOD} | | |
| ***Description*** | Schedule period of the main function StbM_MainFunction. Unit: [s]. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucFloatParamDef | | |
| ***Range*** | 1E-6 .. INF | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | |
|---|---|
| ***SWS Item*** | **ECUC_StbM_00001 :** |
| ***Name*** | StbMTickTypeRange {STBM_TICKTYPE_RANGE} |

| | | | |
|---|---|---|---|
| ***Description*** | Defines the upper range of the type "StbM_TickType". | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucIntegerParamDef | | |
| ***Range*** | 0 .. 65535 | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_StbM_00013 :** | | |
| ***Name*** | StbMVersionInfo {STBM_VERSION_INFO_API} | | |
| ***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*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| |
|---|
| ***No Included Containers*** |

## 10.2.4 StbMDemEventParameterRefs

| | |
|---|---|
| ***SWS Item*** | **ECUC_StbM_00022 :** |
| ***Container Name*** | StbMDemEventParameterRefs |
| ***Description*** | Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references. |
| ***Configuration Parameters*** | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_StbM_00024 : (Obsolete)** | | |
| ***Name*** | STBM_E_INTEGRITY_FAILED | | |
| ***Description*** | Reference to the DemEventParameter which shall be issued when the error "API request integrity failed" has occured.<br>**Tags:**<br>atp.Status=obsolete<br>atp.StatusComment=This reference is set to obsolete and will be removed in the next major release<br>atp.StatusRevisionBegin=4.1.1 | | |
| ***Multiplicity*** | 0..1 | | |
| ***Type*** | Symbolic name reference to [ DemEventParameter ] | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_StbM_00025 : (Obsolete)** | | |
| ***Name*** | STBM_E_REQ_FAILED | | |
| ***Description*** | Reference to the DemEventParameter which shall be issued when the error "API request failed" has occured.<br>**Tags:**<br>atp.Status=obsolete<br>atp.StatusComment=This reference is set to obsolete and will be removed in the next major release<br>atp.StatusRevisionBegin=4.1.1 | | |
| ***Multiplicity*** | 0..1 | | |
| ***Type*** | Symbolic name reference to [ DemEventParameter ] | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | | | |

| |
|---|
| ***No Included Containers*** |

## 10.2.5 StbMSynchronizedTimeBase

| | |
|---|---|
| ***SWS Item*** | **ECUC_StbM_00003 :** |
| ***Container Name*** | StbMSynchronizedTimeBase{STBM_SYNCHRONIZED_TIMEBASE} |
| ***Description*** | Synchronized time.base collects the information about a specific time-base provider within the system. |
| ***Configuration Parameters*** | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_StbM_00014 :** | | |
| ***Name*** | StbMGlobalTimeProviderCallout<br>{STBM_GLOBALTIME_PROVIDER_CALLOUT} | | |
| ***Description*** | Entry address of the time-base specific callout routine which shall be invoked by the StbM for gathering the current time-base value.<br>In case the synchronized time-base is derived from the local time, this container can be omitted. For this case, the StbM achieves the current time-base value by calling the OS interface "GetCounterValue" with the respective OSCounter configured via the ECUC param "StbMLocalTimeRef". | | |
| ***Multiplicity*** | 0..1 | | |
| ***Type*** | EcucFunctionNameDef | | |
| ***Default value*** | -- | | |
| ***maxLength*** | -- | | |
| ***minLength*** | -- | | |
| ***regularExpression*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | |
|---|---|
| ***SWS Item*** | **ECUC_StbM_00015 :** |
| ***Name*** | StbMSyncStateProviderCallout<br>{STBM_SYNCSTATE_PROVIDER_CALLOUT} |
| ***Description*** | Entry address of the time-base specific callout routine which shall be invoked by the StbM for gathering the current time-base status.<br>In case the synchronized time-base is derived from the local time, this container can be omitted. For this case, the state is always |

| | | | |
|---|---|---|---|
| | "STBM_STATE_SYNC". | | |
| ***Multiplicity*** | 0..1 | | |
| ***Type*** | EcucFunctionNameDef | | |
| ***Default value*** | -- | | |
| ***maxLength*** | -- | | |
| ***minLength*** | -- | | |
| ***regularExpression*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| ***SWS Item*** | **ECUC_StbM_00021 :** | | |
|---|---|---|---|
| ***Name*** | StbMSynchronizedTimeBaseIdentifier {STBM_SYNCHRONIZED_TIMEBASE_IDENTIFIER} | | |
| ***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*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| ***SWS Item*** | **ECUC_StbM_00016 :** | | |
|---|---|---|---|
| ***Name*** | StbMTickDurationProviderCallout {STBM_TICKDURATION_PROVIDER_CALLOUT} | | |
| ***Description*** | Entry address of the time-base specific callout routine which shall be invoked by the StbM for gathering the current time-base tick duration.<br>In case the synchronized time-base is derived from the local time, this container can be omitted. For this case, the StbM achieves the tick duration by configuring an alarm which uses the HW counter configured via the ECUC param "StbMLocalTimeRef". Note: The tick duration of the local time will not change during runtime. | | |
| ***Multiplicity*** | 0..1 | | |
| ***Type*** | EcucFunctionNameDef | | |
| ***Default value*** | -- | | |
| ***maxLength*** | -- | | |
| ***minLength*** | -- | | |
| ***regularExpression*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| ***SWS Item*** | **ECUC_StbM_00005 : (Obsolete)** | | |
|---|---|---|---|
| ***Name*** | StbMFlexRayClusterRef {STBM_FLEXRAY_CLUSTER_REF} | | |
| ***Description*** | Please note that this reference is deprecated and will be removed in future.<br>Old description: Optional reference to the FlexRay cluster.<br>**Tags:**<br>atp.Status=obsolete<br>atp.StatusRevisionBegin=4.1.2 | | |
| ***Multiplicity*** | 0..1 | | |
| ***Type*** | Foreign reference to [ FLEXRAY-CLUSTER ] | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |

| | | | |
|---|---|---|---|
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_StbM_00006 :** | | |
| ***Name*** | StbMLocalTimeRef {STBM_LOCAL_TIME_REF} | | |
| ***Description*** | Optional sub container in case a local time shall be accessed.<br>In case this subcontainer is used, the destinated OS counter has to be configured properly, meaning: - the counter is directly driven by a HW timer - the counter's OsCounterTicksPerBase set to one tick in ms. | | |
| ***Multiplicity*** | 0..1 | | |
| ***Type*** | Reference to [ OsCounter ] | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_StbM_00011 : (Obsolete)** | | |
| ***Name*** | StbMTtcanClusterRef {STBM_TTCAN_CLUSTER_REF} | | |
| ***Description*** | Please note that this reference is deprecated and will be removed in future.<br>Old description: Optional reference to the Ttcan cluster.<br>**Tags:**<br>atp.Status=obsolete<br>atp.StatusRevisionBegin=4.1.2 | | |
| ***Multiplicity*** | 0..1 | | |
| ***Type*** | Foreign reference to [ TTCAN-CLUSTER ] | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| |
|---|
| ***No Included Containers*** |

### 10.2.6 StbMTriggeredCustomer

| | |
|---|---|
| ***SWS Item*** | **ECUC_StbM_00004 :** |
| ***Container Name*** | StbMTriggeredCustomer{STBM_TRIGGERED_CUSTOMER} |
| ***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_00017 :** |
| ***Name*** | StbMGlobalTimeCustomerCallback {STBM_GLOBALTIME_CUSTOMER_CALLBACK} |
| ***Description*** | Entry address of the customer specific call-back routine which shall be invoked by the StbM periodically for time value propagation.<br>This configuration is only valid if the explicit OS ScheduleTable is NOT defined as triggered customer (via the reference "StbMOSScheduleTableRef"). |
| ***Multiplicity*** | 0..1 |
| ***Type*** | EcucFunctionNameDef |
| ***Default value*** | -- |
| ***maxLength*** | -- |

| | | | |
|---|---|---|---|
| ***minLength*** | -- | | |
| ***regularExpression*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_StbM_00018 :** | | |
| ***Name*** | StbMSyncStateCustomerCallback {STBM_SYNCSTATE_CUSTOMER_CALLBACK} | | |
| ***Description*** | Entry address of the customer specific call-back routine which shall be invoked by the StbM when state changes occur.<br>This configuration is only valid if the explicit OS ScheduleTable is NOT defined as triggered customer (via the reference "StbMOSScheduleTableRef"). | | |
| ***Multiplicity*** | 0..1 | | |
| ***Type*** | EcucFunctionNameDef | | |
| ***Default value*** | -- | | |
| ***maxLength*** | -- | | |
| ***minLength*** | -- | | |
| ***regularExpression*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_StbM_00019 :** | | |
| ***Name*** | StbMTriggerInSyncState {STBM_TRIGGER_IN_SYNCSTATE} | | |
| ***Description*** | Activate/Deactivate the triggering of the customer in case the related time-base (StbmSynchronizedTimeBaseRef) is not synchronized.<br>True: the customer will only be triggered when the related time-base is synchronized. False: the customer will be triggered with the local time-base when no synchronization for the related time-base is established. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_StbM_00020 :** | | |
| ***Name*** | StbMTriggeredCustomerPeriod {STBM_TRIGGERED_CUSTOMER_PERIOD} | | |
| ***Description*** | The triggering period of the triggered customer, called by the StbM_MainFunction.<br>The period is documented in microseconds. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucIntegerParamDef | | |
| ***Range*** | 0 .. 4294967295 | | |
| ***Default value*** | -- | | |
| ***ConfigurationClass*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| *SWS Item* | **ECUC_StbM_00007 :** | | |
|---|---|---|---|
| *Name* | StbMOSScheduleTableRef {STBM_OS_SCHEDULETABLE_REF} | | |
| *Description* | Optional reference to synchronized OS ScheduleTables.<br>This configuration is only valid if the triggered customer shall be an OS ScheduleTable. In this case, the OS ScheduleTable will be explicitly synchronized by the StbM. | | |
| *Multiplicity* | 0..1 | | |
| *Type* | Reference to [ OsScheduleTable ] | | |
| *ConfigurationClass* | *Pre-compile time* | X | All Variants |
| | *Link time* | -- | |
| | *Post-build time* | -- | |
| *Scope / Dependency* | scope: local | | |

| *SWS Item* | **ECUC_StbM_00010 :** | | |
|---|---|---|---|
| *Name* | StbMSynchronizedTimeBaseRef {SYNCHRONIZEDTIMEBASE_REF} | | |
| *Description* | Mandatory reference to the required synchronized time-base. | | |
| *Multiplicity* | 1 | | |
| *Type* | Symbolic name reference to [ StbMSynchronizedTimeBase ] | | |
| *ConfigurationClass* | *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_00344, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00170, SRS_BSW_00380, SRS_BSW_00412, SRS_BSW_00387, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00438, SRS_BSW_00375, SRS_BSW_00416, SRS_BSW_00437, SRS_BSW_00168, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00336, SRS_BSW_00422, SRS_BSW_00417, SRS_BSW_00455, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00415, SRS_BSW_00164, SRS_BSW_00325, SRS_BSW_00326, SRS_BSW_00342, SRS_BSW_00160, SRS_BSW_00453, SRS_BSW_00007, SRS_BSW_00413, SRS_BSW_00347, SRS_BSW_00441, SRS_BSW_00307, SRS_BSW_00314, SRS_BSW_00370, SRS_BSW_00353, SRS_BSW_00361, SRS_BSW_00328, SRS_BSW_00312, SRS_BSW_00006, SRS_BSW_00439, SRS_BSW_00304, SRS_BSW_00355, SRS_BSW_00378, SRS_BSW_00306, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00371, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00440, SRS_BSW_00330, SRS_BSW_00009, SRS_BSW_00010, SRS_BSW_00333, SRS_BSW_00341, SRS_BSW_00334)