MICROSAR Classic StbM

Technical Reference

Version 9.6.0

Authors	visssf, visbbk, visgut, vistra, vissi, visgig, visjwe, istaehle, vrietz, fmommer
Status	Released

Document Information

History

Author	Date	Version	Remarks
visssf	2012-12-06	1.0.0	Initial version
visssf	2013-04-10	1.1.0	Set status to released Added StbM_InitMemory Added description of critical sections
visssf	2013-10-30	2.0.0	Fixed review findings
visssf	2014-11-05	3.0.0	Support new global time synchronization concept
visssf	2015-01-29	3.1.0	Renamed document
visssf	2015-08-12	3.2.0	Changed StbM_TimeStampRawType to structure
visssf	2016-01-08	4.0.0	Support of High Resolution Time Base Reference Clock based on GPT Added EthTSyn_SetGlobalTime to used services
visssf	2016-03-22	4.1.0	Support of time gateways
visssf	2017-03-29	4.2.0	Support of time correction Removed EthTSyn from used services Added EthIf to used services
visssf	2017-06-29	5.1.0	Support time precision measurement
visbbk, visgut, visssf, vistra	2017-08-01	5.2.0	New Notifications for time expiration and status changed events Support of immediate time synchronization Added handling of time leaps
vistra, visssf	2017-09-27	5.3.0	Minor AR4.3 extensions Added StbM_GetTimeBaseStatus Changed availability of service ports and operations according to AR 4.3
vissi	2018-02-28	5.4.0	Updated AUTOSAR architecture Updated referenced documents
visssf	2018-05-22	5.5.0	Support schedule table synchronization for different counter resolutions Support time base specific time difference calculation according to AR 4.3.1 Improved exclusive area description
visssf	2018-07-16	5.6.0	First ASIL D version Minor corrections

vistra	2018-08-14	5.7.0	StbM avoids accessing an uncertain Ethernet clock source Added description of parameter 'ConfigPtr' of StbM_Init() Updated chapter 4.1 to new template
visssf	2018-10-22	5.8.0	MISRA-C:2012 compliance Updated used services
vissi	2019-01-23	5.8.1	Minor improvements
vissi	2019-09-05	6.0.0	Minor improvements Module rework and introduction of sub units for functional safety Time correction and time leap behavior implemented according AR 4.3.1 Documented deviations to AR 4.3.1
visgig, vistra	2020-03-30	7.0.0	Support of enhanced precision of Global Time (AR RfC 79959) Initialization by BswM instead of EcuM Added limitations for Local Time
visssf	2020-05-07	8.0.0	Support of time validation for Ethernet use case (beta)
visjwe, visgig, visssf	2020-07-03	8.1.0	Support of time validation for Ethernet use case (production) Minor improvements Post-Build Variant Support
visssf, vistra	2021-03-08	8.2.0	Corrected value range of StbM_TimeBaseStatusType Reserved time bases are not supported
visjwe, visssf, istaehle	2022-04-20	9.0.0	ASR memmap include structure ALL SLP Multicore Distribution for CAN Added API StbM_GetMainTime Product name updated to MICROSAR Classic
visssf	2022-06-13	9.0.1	Minor corrections
vrietz	2022-06-24	9.1.0	Added error checks and parameters for STBM_E_PARAM_USERDATA and STBM_E_PARAM_TIMESTAMP
vrietz, istaehle, visssf, visjwe	2022-09-02	9.2.0	Support of time base cloning Updated document to reference AR 21-11 Updated adjacent interface diagram Add hint for GPT timer usage Minor corrections
istaehle, visssf	2022-10-19	9.3.0	Support rate correction of sync reception delay Improved documentation of Multi-Partition support Option to use time domains 32 to 127 as synchronized time bases

visssf	2023-01-13	9.4.0	Improved documentation of Multi-Partition support
vrietz	2023-09-19	9.5.2	Updated description of StbM_GetMainTime API and GlobalTime_Master_<TB> port
fmommer	2023-11-20	9.6.0	Updated description of the critical sections

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	Specification of Synchronized Time-Base Manager	R21-11
[2]	AUTOSAR	List of Basic Software Modules	R21-11
[3]	AUTOSAR	Specification of Default Error Tracer	R21-11
[4]	AUTOSAR	Specification of RTE Software	R21-11
[5]	AUTOSAR	Specification of Operating System	R21-11
[6]	AUTOSAR	Specification of Ethernet Interface	R21-11
[7]	AUTOSAR	Specification of GPT Driver	R21-11
[8]	AUTOSAR	Specification of ECU State Manager	R21-11
[9]	AUTOSAR	Specification of Time Synchronization over CAN	R19-11
[10]	AUTOSAR	Specification of Time Synchronization over Ethernet	R4.4.0
[11]	AUTOSAR	Specification of Time Synchronization over FlexRay	R4.4.0

Scope of the Document

This technical reference describes the general use of the Synchronized Time-Base Manager.

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations

Tables

1 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module StbM as specified in [1].

Supported Configuration Variants:	pre-compile, post-build-selectable	
Vendor ID:	STBM_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	STBM_MODULE_ID	160 decimal (according to ref. [2])

The purpose of the Synchronized Time-Base Manager is to provide synchronized time bases to its customers, i.e., time bases, which are synchronized with time bases on other nodes of a distributed system.

1.1 Architecture Overview

The following figure shows where the StbM is located in the AUTOSAR architecture.

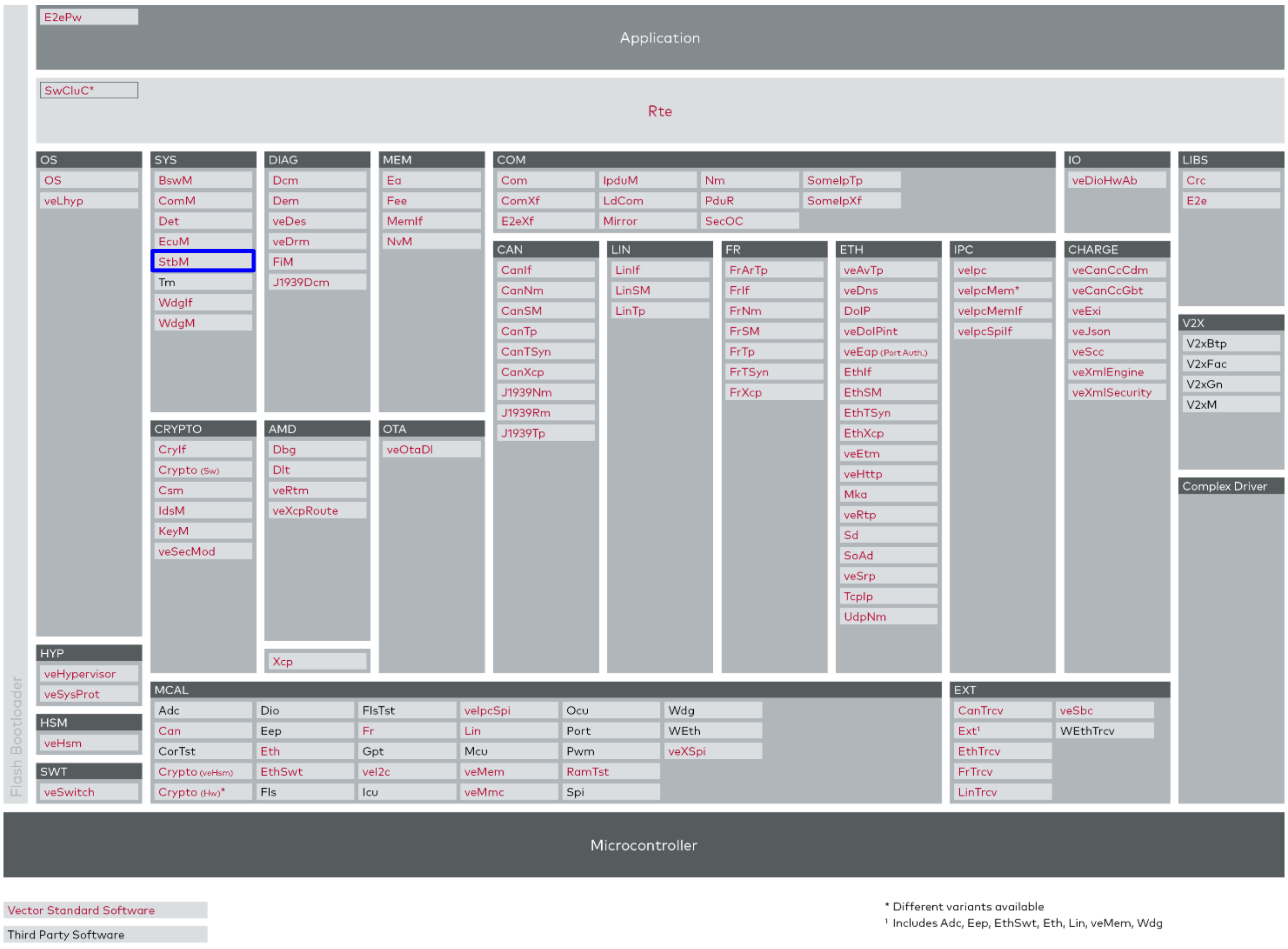


Figure 1-1 AUTOSAR Architecture Overview

The next figure shows the interfaces to adjacent modules of the StbM. These interfaces are described in chapter 4.

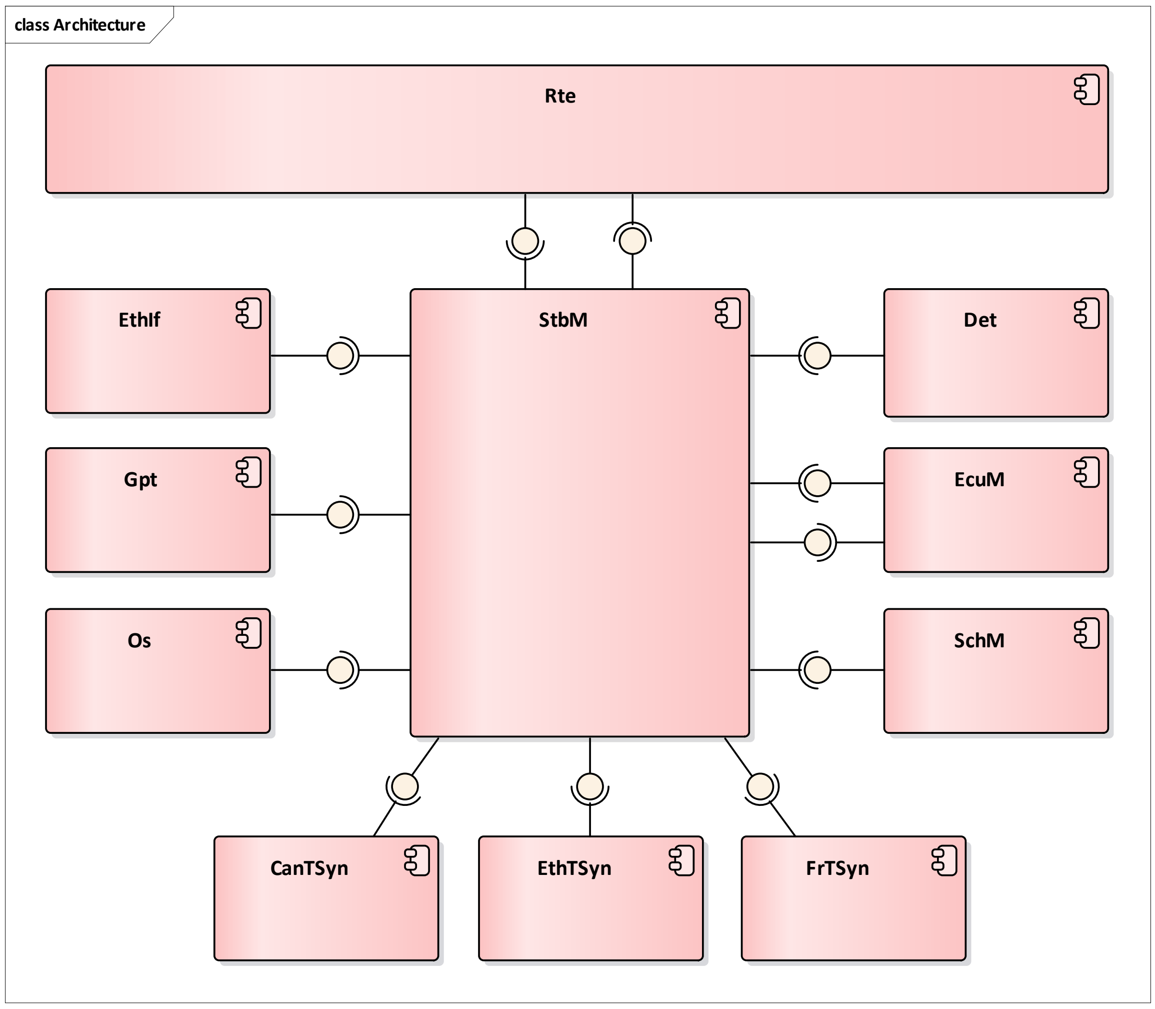


Figure 1-2 Interfaces to adjacent modules of the StbM

Applications do not access the services of the BSW modules directly. They use the service ports provided by the BSW modules via the RTE. The service ports provided by the StbM are listed in chapter 4.5 and are defined in [1].

2 Functional Description

2.1 Features

The features listed in the following tables cover the complete functionality specified for the StbM.

The AUTOSAR standard functionality is specified in [1], the corresponding features are listed in the tables

> Table 2-1 Supported AUTOSAR standard conform features

> Table 2-2 Not supported AUTOSAR standard conform features

Vector Informatik provides further StbM functionality beyond the AUTOSAR standard. The corresponding features are listed in the table

> Table 2-3 Features provided beyond the AUTOSAR standard

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Synchronization of runnable entities and OS schedule tables
Provision of absolute time value
Autonomous maintenance of the time-base
Time gateways
High Resolution Time Base Reference Clock based on GPT
Time correction
Time precision measurement
Status notifications
Time Notifications
Immediate time synchronization
Time validation for Ethernet use case
Variant Post-Build
Multi-Partition
Multi-Instance CanTSyn
Time Base Cloning

Table 2-1 Supported AUTOSAR standard conform features

2.1.1 Deviations

The following features specified in [1] are not supported:

Category	Description
Functional	Storage of the time base at shutdown
Functional	Loading of the time base during initialization

Category	Description
Functional	Complex Device Driver Interface (e.g. StatusNotification and TimeNotification callbacks)
Functional	Time Deviation is not calculated in StbM_TimeNotificationCallback (interrupt context), but in StbM_NotificationFunction (task context).
Functional	Status Notification is implemented according to AR 4.3.1 (Parameter StbMNotificationInterface not supported).
Functional	Initialization by BswM instead of EcuM
Functional	Time validation for CAN and FR use case
Functional	For every master of a time base, which has time validation enabled, a separate Master Timing Record Table is implemented.
Functional	Multi-Instance FrTSyn and EthTSyn
Functional	Reserved time bases are not supported
Functional	CAN hardware timestamping is not supported
API	User data handling is not supported for offset time bases.The signatures of the APIs StbM_SetOffset and StbM_GetOffset are still implemented according AR 4.2.x.
API	StbM_SetBusProtocolParam, StbM_GetBusProtocolParam and corresponding types are not supported.

Table 2-2 Not supported AUTOSAR standard conform features

2.1.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
Memory Initialization
API to retrieve main time tuple, status and rate deviation
Additional error check in StbM_StartTimer
Option to use time domains 32 to 127 as synchronized time bases

Table 2-3 Features provided beyond the AUTOSAR standard

2.1.2.1 Memory Initialization

AUTOSAR expects the startup code to automatically initialize RAM. Not every startup code of embedded targets reinitializes all variables correctly. It is possible that the state of a variable may not be initialized as expected. To avoid this problem, the Vector AUTOSAR StbM provides an additional function to initialize the relevant variables of the StbM. See also chapters 2.2 and 4.2.2 for details.

2.1.2.2 API to retrieve main time tuple, status and rate deviation

The Vector AUTOSAR StbM provides an additional function that retrieves the global time sync, virtual local time sync, status and rate deviation for a submitted time-base. See also chapter 4.2.10 for details.

2.1.2.3 Additional error check in StbM_StartTimer

In addition to the error checks specified by the AUTOSAR standard, the function StbM_StartTimer also checks the given timestamp (expire time) for invalid elements and reports STBM_E_PARAM_TIMESTAMP on error.

2.1.2.4 Option to use time domains 32 to 127 as synchronized time bases

StbM allows to configure time bases 32 to 127 as pure local or synchronized time bases. The time base type is identified depending on whether a reference from a TSyn module exists.

2.1.3 Limitations

2.1.3.1 Local Time

The Virtual Local Time is restricted by the 64bit nanoseconds of the underlying `StbM_VirtualLocalTimeType`. StbM will stop working on any overflow of the Virtual Local Time.

StbM supports only one ethernet controller per time domain in EthTSyn, if ethernet is used as source for the Virtual Local Time.

2.1.4 Notifications for time expiration and status changed events

Time notifications or status changed events can be registered by notification customers. A notification customer can either be a SWC or a BSW-module. The StbM allows customers to be notified when an alarm expires or any of the registered status changed events occurs.

2.1.4.1 Status Notifications

The StbM allows notification customers to register for one or more status change events. Whenever a status change is triggered the customer is informed by a notification callback. This callback function is called in the context of the `StbM_MainFunction()`. However, this may result in a delay from the point of time when the event occurred and when the customer is actually notified. To decrease this delay, a smaller MainFunction cycle time needs to be configured.

Since the notification of the customers is realized via service ports, the delay is also influenced by the task mapping in the RTE and the OS task configuration. If a customer has to be notified as early as possible, the priority of the task the notification runnable is mapped to has to be increased. Otherwise this task will wait until other higher priority tasks are finished.

Table 4-12 lists all events which can be detected.

2.1.4.2 Time Notifications

The StbM allows notification customers to be notified whenever a defined timer expires. This expire time is set through function `StbM_StartTimer` by the notification customer.

A GPT timer is used to monitor the timeout of the expiration timer within a configurable interval. After the GPT timer expires, `StbM_TimerCallback` is called to notify StbM about the expiration. The GPT timer must not be started outside the StbM. The GPT timer needs to be referenced in

> /MICROSAR/StbM/StbMGeneral/StbMGptTimerRef

Further, the referenced GPT Timer has to configure `StbM_TimerCallback` as callback function in

> /Gpt/GptChannelConfigSet/GptChannelConfiguration/GptNotification

`StbM_TimerCallback` might be called in interrupt context. However, customer notifications are called in task context to decouple the application from the interrupt context. Therefore, the user has to map the function `StbM_NotificationFunction` to an existent task. Depending on the priority of the task, the time deviation between the calculated expire time and actual time may be more or less. This time deviation is provided along with the callback function. If a notification customer has to be notified as early as possible, the priority of this task has to be increased. Otherwise this task will wait until other higher priority tasks are finished.

It should be noted that this time notification feature is only compatible with the MICROSAR Classic RTE.

Caution
StbM_NotificationFunction only works with MICROSAR Classic RTE.

The provided expire time in `StbM_StartTimer` must, at least, exceed the cycle time of the Main Function to ensure that the timer can be monitored properly. During runtime, the start of a timer may also be denied, if the GPT Timer is currently running and the newly provided expire time would expire before expiration of the GPT Timer.

2.2 Initialization

The Synchronized Time-Base Manager is initialized in four steps:

- `StbM_InitMemory()`: On platforms in which the Random Access Memory (RAM) is not initialized to zero by the startup code, this function has to be called first before `StbM_PreInit()` is called.
- `StbM_PreInit()`: Called by the EcuM before the Os is started.
- `StbM_Init()`: Called by the BSW Mode Manager (BswM) for every partition where the StbM is running after Os has started.

- `StbM_PostInit()`: Called by the BSW Mode Manager (BswM) after every partition where the StbM is running has been initialized.

GPT timers used as local clock by the StbM must not be started previously, these are started by the StbM through the initialization phase.

2.3 States

The StbM has no internal state machine, it is operational after initialization.

2.4 Main Functions

The `StbM_MainFunction()` updates the local time bases, monitors timeouts for the detection of lost synchronization and triggers customers, which includes the synchronization of OS ScheduleTables and notification of customers about status related events. Besides, the `StbM_MainFunction()` notifies the application about new available measurement data blocks and starts a GPT timer, if a time notification customer wants to be notified when an alarm expires.

2.4.1 Main Functions with Multi-Partition configurations

For configurations where time bases are mapped to more than one partition, a `StbM_MainFunction_<OsApplication>()` per partition with at least one mapped time base is generated. Each of these generated main functions needs to be mapped to the correct partition. See chapter 5.2 for details about the partition mapping of time bases.

2.5 Error Handling

2.5.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [3], if development error reporting is enabled (i.e. pre-compile parameter `STBM_DEV_ERROR_DETECT==STD_ON`).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportError()`.

The reported StbM ID is 160.

The reported service IDs identify the services which are described in 4.2. The following table presents the service IDs and the related services:

Service ID	Service
0x00	StbM_Init
0x04	StbM_MainFunction
0x05	StbM_GetVersionInfo
0x07	StbM_GetCurrentTime
0x08	StbM_GetCurrentTimeExtended
0x0B	StbM_SetGlobalTime
0x0C	StbM_SetUserData
0x0D	StbM_SetOffset

Service ID	Service
0x0E	StbM_GetOffset
0x0F	StbM_BusSetGlobalTime
0x10	StbM_UpdateGlobalTime
0x11	StbM_GetRateDeviation
0x12	StbM_SetRateCorrection
0x13	StbM_GetTimeLeap
0x14	StbM_GetTimeBaseStatus
0x15	StbM_StartTimer
0x16	StbM_GetSyncTimeRecordHead
0x17	StbM_GetOffsetTimeRecordHead
0x1B	StbM_GetTimeBaseUpdateCounter
0x1C	StbM_TriggerTimeTransmission
0x1D	StbM_GetMasterConfig
0x1E	StbM_GetCurrentVirtualLocalTime
0x1F	StbM_BusGetCurrentTime
0x22	StbM_EthSetMasterTimingData
0x23	StbM_EthSetPdelayInitiatorData
0x24	StbM_EthSetPdelayResponderData
0x28	StbM_EthSetSlaveTimingData
0x2B	StbM_CloneTimeBase
0xC0	StbM_PreInit
0xC1	StbM_PostInit
0xC2	StbM_GetMainTime

Table 2-4 Service IDs

The errors reported to DET are described in the following table:

Error Code		Description
0x11	STBM_E_INIT_FAILED	StbM_Init called with an invalid configuration pointer
0x0A	STBM_E_PARAM	API requests called with wrong parameter
0x0B	STBM_E_NOT_INITIALIZED	Synchronized Time-Base Manager is not initialized
0x10	STBM_E_PARAM_POINTER	Invalid pointer in parameter list
0x12	STBM_E_SERVICE_DISABLED	API disabled by configuration
0x25	STBM_E_PARAM_TIMESTAMP	API called with timestamp having invalid elements
0x26	STBM_E_PARAM_USERDATA	API called with invalid user data (length > 3)

Table 2-5 Errors reported to DET

2.5.2 Production Code Error Reporting

No production error codes are currently used by StbM.

3 Integration

This chapter gives necessary information for the integration of the MICROSAR Classic StbM into an application environment of an ECU.

3.1 Embedded Implementation

The delivery of the StbM consists out of these files:

File Name	Description	Integration Tasks
StbM.c	Main implementation file of the StbM.	-
StbM.h	Main header file of the StbM.	-
StbM_Types.h	Header file that contains the type definitions of the StbM.	-
StbM_EthTSyn.h	Header file that contains prototypes for Ethernet specific functions.	-
StbM_Cfg.c	Generated file that contains definitions of structures in pre-compile-time and post-build variant.	-
StbM_Cfg.h	Generated file that contains declarations of structures in pre-compile-time and post-build variant.	-
StbM_Cfg_<OsApplication>.c	Generated file that contains definitions of structures in pre-compile-time and post-build variant for the respective OsApplication.	-

Table 3-1 Implementation files

3.2 Critical Sections

The StbM has code sections which need protection against interrupts and OS tasks which can interrupt each other. Therefore, the StbM uses two exclusive areas which require a global interrupt lock:

> `STBM_EXCLUSIVE_AREA_0`

> `STBM_EXCLUSIVE_AREA_1`

The exclusive areas are defined differently for single-core and multi-core configurations. See chapter 3.2.1 and 3.2.2 for more information.

Depending on the StbM configuration, the StbM calls OS APIs like `GetCounterValue()` and `GetElapsedValue()`. According to the AUTOSAR OS specification, it is not allowed to call these OS APIs with disabled interrupts. Nevertheless, the StbM module requires the interrupt lock to be able to guarantee high accuracy and data consistency.

3.2.1 Single-Core Configuration

For single-core configurations, both exclusive areas synchronize data locally on the core and both need to be configured to ensure data consistency. It is recommended to configure the exclusive areas with the same implementation mechanism.

> `STBM_EXCLUSIVE_AREA_0 and STBM_EXCLUSIVE_AREA_1: These critical sections synchronize data locally on one core.`

Caution

If the StbM is configured to use OS APIs and the implementation method of the exclusive areas is configured to `OS_INTERRUPT_BLOCKING` or `ALL_INTERRUPT_BLOCKING`, the OS may report the error `E_OS_DISABLEDINT`. In that case, the implementation method of the exclusive areas inside the RTE / SchM configuration needs to be set to `OS_RESOURCE` or `CUSTOM`.

If `OS_RESOURCE` is selected, the ISR(s) of the bus specific TSyn modules, e.g. the CAN ISR(s), need to reference the OS Resource created by the RTE.

If `CUSTOM` is selected, the SchM APIs for entering and exiting the exclusive area need to be implemented manually by using an interrupt lock mechanism, but without calling OS APIs like `SuspendOSInterrupts()` or `DisableAllInterrupts()`.

Note:

The exclusive area implementation method `CUSTOM` is a MICROSAR Classic RTE extension and might not be available in other RTEs.

3.2.2 Multi-Core Configuration

For multi-core configurations, the exclusive areas have the following definition:

> `STBM_EXCLUSIVE_AREA_0`: This critical section synchronizes data locally on one core.

> `STBM_EXCLUSIVE_AREA_1`: This critical section synchronizes data across all cores on which a time base is mapped.

Caution

If the StbM is configured to use OS APIs and the implementation method of the exclusive areas is configured to `OS_INTERRUPT_BLOCKING` or `ALL_INTERRUPT_BLOCKING`, the OS may report the error `E_OS_DISABLEDINT`. In that case, the implementation method of the exclusive areas inside the RTE / SchM configuration needs to be set to `OS_RESOURCE` or `CUSTOM`.

If `OS_RESOURCE` is selected, the ISR(s) of the bus specific TSyn modules, e.g. the CAN ISR(s), need to reference the OS Resource created by the RTE.

If `CUSTOM` is selected, the SchM APIs for entering and exiting the exclusive area need to be implemented manually by using an interrupt lock mechanism, but without calling OS APIs like `SuspendOSInterrupts()` or `DisableAllInterrupts()`.

In multi-core configurations the implementation method of the exclusive area `STBM_EXCLUSIVE_AREA_1` needs to be set to `OS_SPINLOCK` or `CUSTOM`. Furthermore, in the multi-core use case, it is not possible to use OS Resources, because they cannot be shared between different cores.

If the StbM is configured to use OS APIs in a multi-core configuration, `OS_SPINLOCK` can also not be used, because spinlocks block OS interrupts. Therefore, in this case, the implementation method of both exclusive areas needs to be set to `CUSTOM`. In addition, an OS spinlock needs to be created manually with the lock method `LOCK_NOTHING` and referenced by the exclusive area `STBM_EXCLUSIVE_AREA_1`. The SchM APIs for entering and exiting the exclusive areas need to be implemented manually.

Note:

The exclusive area implementation method `CUSTOM` is a MICROSAR Classic RTE extension and might not be available in other RTEs.

For details about exclusive areas refer to [4].

3.3 OSScheduleTable Synchronization

For the synchronization of OS schedule tables by the StbM, an OS is needed that supports the synchronization of schedule tables. Furthermore, it is required that the ticks of an OS counter, which drives a schedule table, have a duration of at most 1 microsecond.

3.4 Memory Sections

The StbM_MemMap.h is generated by the MemMap Generator (/ActiveEcuC/MemMap). If adaptions should be done to the Memory Mapping of the StbM, the changes must be configured in the MemMap Generator.

4 API Description

For an interfaces overview please see Figure 1-2.

4.1 Type Definitions

The types defined by the StbM are described in this chapter.

Type Name	C-Type	Description	Value Range
StbM_SynchronizedTimeBaseType	uint16	Variables of this type are used to represent the kind of synchronized time-base.	`0..2^16-1`
StbM_RateDeviationType	sint16	Variables of this type are used to express a rate deviation in ppm.	`-32000..32000`
StbM_TimeDiffType	sint32	Variables of this type are used to express time differences as signed values in nanoseconds	`-2147483647 .. 2147483647`
StbM_CustomerIdType	uint16	Unique identifier of a notification customer	`0..255`
StbM_MasterConfigType	uint8	Variables of this type are used to indicate if the system wide master functionality for a given time base is available or not.	`0x00: STBM_SYSTEM_WIDE_MASTER_DISABLED`
			`0x01: STBM_SYSTEM_WIDE_MASTER_ENABLED`

Table 4-1 Type definitions

[StbM_ConfigType]

This structure contains the configuration data of the StbM module. The elements depend on the configuration.

[StbM_TimeBaseStatusType]

This structure is used to express if and how a local time base is synchronized to the global time master. The type is a bit field of individual status bits, although not every combination is possible, i.e. any of the bits STBM_TIMEOUT, STBM_TIMELEAP and STBM_SYNC_TO_GATEWAY can only be set if the STBM_GLOBAL_TIME_BASE bit is set.

Struct Element Name	Kind	Mask	Description	Value Range
TIMEOUT	bit	0x01	Bit 0	`0x00` No timeout on receiving Synchronization Messages
				`0x01` Timeout on receiving Synchronization Messages
				`0x00`

Struct Element Name	Kind	Mask	Description	Value Range
SYNC_TO_GATEWAY	bit	0x04	Bit 2	Local Time Base is synchronous to Global Time Master
				`0x01` Local Time Base updates are based on a Time Gateway below the Global Time Master
GLOBAL_TIME_BASE	bit	0x08	Bit 3	`0x00` Local Time Base is based on Local Time Base reference clock only (never synchronized with Global Time Base)
				`0x01` Local Time Base was at least one time synchronized with Global Time Base
TIMELEAP_FUTURE	bit	0x10	Bit 4	`0x00` No leap into the future within the received time for Time Base
				`0x01` Leap into the future within the received time for Time Base exceeds a configured threshold
TIMELEAP_PAST	bit	0x20	Bit 5	`0x00` No leap into the past within the received time for Time Base
				`0x01` Leap into the past within the received time for Time Base exceeds a configured threshold

Table 4-2 StbM_TimeBaseStatusType

[StbM_TimeStampType]

This structure is used for expressing time stamps including relative time and absolute calendar time.

Struct Element Name	C-Type	Description	Value Range
timeBaseStatus	StbM_TimeBase StatusType	Status of the Time Base	`0..2^8-1`
nanoseconds	uint32	Nanoseconds part of the time	`0..999999999`
seconds	uint32	32 bit LSB of the 48 bits Seconds part of the time	`0..2^32-1`
secondsHi	uint16	16 bit MSB of the 48 bits Seconds part of the time	`0..2^16-1`

Table 4-3 StbM_TimeStampType

[StbM_VirtualLocalTimeType]

This structure is used for expressing time stamps of the Virtual Local Time. The unit is Nanoseconds.

Struct Element Name	C-Type	Description	Value Range
nanosecondsLo	uint32	Least significant 32 bits of the 64 bit Virtual Local Time	`0..2^32-1`
nanosecondsHi	uint32	Most significant 32 bits of the 64 bit Virtual Local Time	`0..2^32-1`

Table 4-4 StbM_VirtualLocalTimeType

[StbM_TimeStampExtendedType]

This structure is used for expressing time stamps including relative time and absolute calendar time.

Struct Element Name	C-Type	Description	Value Range
timeBaseStatus	StbM_TimeBase StatusType	Status of the Time Base	`0..2^8-1`
nanoseconds	uint32	Nanoseconds part of the time	`0..999999999`
seconds	uint64	48 bit Seconds part of the time	`0..2^48-1`

Table 4-5 StbM_TimeStampExtendedType

[StbM_UserDataType]

This structure is used for expressing the user data of the time base.

Struct Element Name	C-Type	Description	Value Range
userDataLength	uint8	User Data Length in bytes	`0..3`
userByte0	uint8	User Byte 0	`0..2^8-1`
userByte1	uint8	User Byte 1	`0..2^8-1`
userByte2	uint8	User Byte 2	`0..2^8-1`

Table 4-6 StbM_UserDataType

[StbM_MeasurementType]

This structure contains additional measurement data.

Struct Element Name	C-Type	Description	Value Range
pathDelay	uint32	Propagation delay in nanoseconds	`0..999999999`

Table 4-7 StbM_MeasurementType

[StbM_SyncRecordTableHeadType]

This structure contains the information of the record table header of the Synchronized Time Base.

Struct Element Name	C-Type	Description	Value Range
SynchronizedTimeDomain	uint8	Time Domain	`0..15`
HWfrequency	uint32	HW Frequency in Hz	`0..2^32-1`
HWprescaler	uint32	Prescaler value	`0..2^32-1`

Table 4-8 StbM_SyncRecordTableHeadType

[StbM_SyncRecordTableBlockType]

This structure contains the information of the record table block of the Synchronized Time Base.

Struct Element Name	C-Type	Description	Value Range
GlbSeconds	uint32	Seconds of the Local Time Base directly after synchronization with the Global Time Base	`0..2^32-1`
GlbNanoSeconds	uint32	Nanoseconds of the Local Time Base directly after synchronization with the Global Time Base	`0..999999999`
TimeBaseStatus	StbM_TimeBaseStatusType	Time Base Status of the Local Time Base directly after synchronization with the Global Time Base	`0..2^8-1`
VirtualLocalTimeLow	uint32	Least significant 32 bit of the Virtual Local Time directly after synchronization with the Global Time Base	`0..2^32-1`
RateDeviation	sint16	Calculated Rate Deviation directly after rate deviation measurement	`-32000..32000`
LocSeconds	uint32	Seconds of the Local Time Base directly before synchronization with the Global Time Base	`0..2^32-1`
LocNanoSeconds	uint32	Nanoseconds of the Local Time Base directly before synchronization with the Global Time Base	`0..999999999`
PathDelay	uint32	Current propagation delay in nanoseconds	`0..999999999`

Table 4-9 StbM_SyncRecordTableBlockType

[StbM_OffsetRecordTableHeadType]

This structure contains the information of the record table header of the Offset Time Base.

Struct Element Name	C-Type	Description	Value Range
OffsetTimeDomain	uint8	Time Domain	`16..31`

Table 4-10 StbM_OffsetRecordTableHeadType

[StbM_OffsetRecordTableBlockType]

This structure contains the information of the record table block of the Offset Time Base.

Struct Element Name	C-Type	Description	Value Range
GlbSeconds	uint32	Seconds of the Offset Time Base	`0..2^32-1`
GlbNanoSeconds	uint32	Nanoseconds of the Offset Time Base	`0..999999999`
TimeBaseStatus	StbM_TimeBaseStatusType	Time Base Status of the Local Time Base directly after synchronization with the Global Time Base	`0..2^8-1`

Table 4-11 StbM_OffsetRecordTableBlockType

[StbM_TimeBaseNotificationType]

This 32 Bit bitfield defines a number of global time related events. This Type is used for storing the events in the status variable NotificationEvents and for setting the mask variable NotificationMask which defines a subset of events for which an interrupt request shall be raised.

Status Event Name	Kind	Mask	Status Event Set Condition
EV_GLOBAL_TIME_BASE	bit	0x01U	1: GLOBAL_TIME_BASE bit has changed from 0 to 1 0: otherwise
EV_TIMEOUT_OCCURED	bit	0x02U	1: TIMEOUT bit has changed from 0 to 1 0: otherwise
EV_TIMEOUT_REMOVED	bit	0x04U	1: TIMEOUT bit has changed from 1 to 0 0: otherwise
EV_TIMELEAP_FUTURE	bit	0x08U	1: TIMELEAP_FUTURE bit has changed from 0 to 1 0: otherwise
EV_TIMELEAP_FUTURE_REMOVED	bit	0x10U	1: TIMELEAP_FUTURE bit has changed from 1 to 0 0: otherwise
EV_TIMELEAP_PAST	bit	0x20U	1: TIMELEAP_PAST bit has changed from 0 to 1

			0: otherwise
EV_TIMELEAP_PAST_REMOVED	bit	0x40U	1: TIMELEAP_PAST bit has changed from 1 to 0 0: otherwise
EV_SYNC_TO_SUBDOMAIN	bit	0x80U	1: SYNC_TO_GATEWAY bit has changed from 0 to 1 0: otherwise
EV_SYNC_TO_GLOBAL_MASTER	bit	0x100U	1: SYNC_TO_GATEWAY bit has changed from 1 to 0 0: otherwise
EV_RESYNC	bit	0x200U	1: resynchronization has occurred and a new time value has been applied 0: otherwise
EV_RATECORRECTION	bit	0x400U	1: a valid rate correction has been calculated (not beyond limits) 0: otherwise

Table 4-12 StbM_TimeBaseNotificationType

[StbM_PortIdType]

This structure contains port identity data.

Struct Element Name	C-Type	Description	Value Range
clockIdentity	uint64	Clock identity of the clock	`0..2^64-1`
portNumber	uint16	Number of Ethernet port	`0..2^16-1`

Table 4-13 StbM_PortIdType

[StbM_TimeStampShortType]

This structure contains a time stamp with a limited range including relative time and absolute calendar time.

Struct Element Name	C-Type	Description	Value Range
nanoseconds	uint32	Nanoseconds part of the time	`0..999999999`
seconds	uint32	32 bit LSB of the 48 bits Seconds part of the time	`0..2^32-1`

Table 4-14 StbM_TimeStampShortType

[StbM_EthTimeMasterMeasurementType]

This structure contains detailed data for Time Validation of the Time Master on Ethernet.

Struct Element Name	C-Type	Description	Value Range
sequenceId	uint16	sequenceId of sent Ethernet frame	`0..2^16-1`

Struct Element Name	C-Type	Description	Value Range
sourcePortId	StbM_PortIdType	sourcePortId of sending Ethernet port	See Table 4-13
syncEgressTimestamp	StbM_VirtualLocalTimeType	Egress timestamp of Sync frame	See Table 4-4
preciseOriginTimestamp	StbM_TimeStampShortType	preciseOriginTimestamp as copied to the Follow_Up frame	See Table 4-14
correctionField	sint64	correctionField as copied to the Follow_Up frame	`-2^63 .. 2^63-1`

Table 4-15 StbM_EthTimeMasterMeasurementType

[StbM_EthTimeSlaveMeasurementType]

This structure contains detailed data for Time Validation of the Time Slave on Ethernet.

Struct Element Name	C-Type	Description	Value Range
sequenceId	uint16	sequenceId of received Sync frame	`0..2^16-1`
sourcePortId	StbM_PortIdType	sourcePortId of received Sync frame	See Table 4-13
syncIngressTimestamp	StbM_VirtualLocalTimeType	Ingress timestamp of Sync frame converted to Virtual Local Time	See Table 4-4
preciseOriginTimestamp	StbM_TimeStampShortType	preciseOriginTimestamp taken from the received Follow_Up frame	See Table 4-14
correctionField	sint64	correctionField taken from the received Follow_Up frame	`-2^63 .. 2^63-1`
pDelay	uint32	Currently valid pDelay value	`0..2^32-1`
referenceLocalTimestamp	StbM_VirtualLocalTimeType	SyncLocal Time Tuple (Virtual Local Time part)	See Table 4-4
referenceGlobalTimestamp	StbM_TimeStampShortType	SyncLocal Time Tuple (Global Time part)	See Table 4-14

Table 4-16 StbM_EthTimeSlaveMeasurementType

[StbM_PdelayInitiatorMeasurementType]

This structure contains detailed timing data for the pDelay Initiator.

Struct Element Name	C-Type	Description	Value Range
sequenceId	uint16	sequenceId of sent Pdelay_Req frame	`0..2^16-1`
requestPortId	StbM_PortIdType	sourcePortId of sent Pdelay_Req frame	See Table 4-13
responsePortId	StbM_PortIdType	sourcePortId of received Pdelay_Resp frame	See Table 4-13
requestOriginTimestamp	StbM_VirtualLocalTimeType	Egress timestamp of Pdelay_Req in Virtual Local Time	See Table 4-4
responseReceiptTimestamp	StbM_VirtualLocalTimeType	Ingress timestamp of Pdelay_Resp in Virtual Local Time	See Table 4-4

Struct Element Name	C-Type	Description	Value Range
requestReceipt Timestamp	StbM_TimeStam pShortType	Ingress timestamp of Pdelay_Req in Global Time taken from the received Pdelay_Resp	See Table 4-14
responseOrigin Timestamp	StbM_TimeStam pShortType	Egress timestamp of Pdelay_Resp in Global Time taken from the received Pdelay_Resp_Follow_Up	See Table 4-14
referenceLocal Timestamp	StbM_VirtualLoc alTimeType	Value of the Virtual Local Time of the reference Global Time Tuple	See Table 4-4
referenceGloba lTimestamp	StbM_TimeStam pShortType	Value of the local instance of the Global Time of the reference Global Time Tuple	See Table 4-14
pdelay	uint32	Currently valid Pdelay value	`0..2^32-1`

Table 4-17 StbM_PdelayInitiatorMeasurementType

[StbM_PdelayResponderMeasurementType]

This structure contains detailed timing data for the pDelay Responder.

Struct Element Name	C-Type	Description	Value Range
sequenceId	uint16	sequenceId of received Pdelay_Req frame	`0..2^16-1`
requestPortId	StbM_PortIdType	sourcePortId of received Pdelay_Req frame	See Table 4-13
responsePortId	StbM_PortIdType	sourcePortId of sent Pdelay_Resp frame	See Table 4-13
requestReceipt Timestamp	StbM_VirtualLoc alTimeType	Ingress timestamp of Pdelay_Req converted to Virtual Local Time	See Table 4-4
responseOrigin Timestamp	StbM_VirtualLoc alTimeType	Egress timestamp of Pdelay_Resp converted to Virtual Local Time	See Table 4-4
referenceLocal Timestamp	StbM_VirtualLoc alTimeType	Value of the Virtual Local Time of the reference Global Time Tuple used to convert requestReceiptTimestamp and responseOriginTimestamp into Global Time	See Table 4-4
referenceGloba lTimestamp	StbM_TimeStam pShortType	Value of the local instance of the Global Time of the reference Global Time Tuple used to convert requestReceiptTimestamp and responseOriginTimestamp into Global Time	See Table 4-14

Table 4-18 StbM_PdelayResponderMeasurementType

[StbM_CloneConfigType]

This 8 bit bitfield is used to refine how a time base is cloned from source to destination. Bits 0..2 can be set individually while bits 3..7 are always 0 (reserved for future usage).

Struct Element Name	Kind	Mask	Description	Value Range
DEFERRED_COPY	bit	0x01	Bit 0	`0x00` Cloning request is processed immediately
				`0x01` Cloning request is deferred until the next update of Source Time Base by <Bus>TSyn module
IMMEDIATE_TX	bit	0x02	Bit 1	`0x00` Time information is transmitted on the next cyclic transmission after cloning
				`0x01` Time information is transmitted on destination bus immediately after cloning
APPLY_RATE	bit	0x04	Bit 2	`0x00` Rate correction value of Source Time Base is ignored
				`0x01` Rate correction value of Source Time Base is applied to Destination Time Base

Table 4-19 StbM_CloneConfigType

4.2 Services provided by StbM

4.2.1 StbM_GetVersionInfo

Prototype	
`void` **`StbM_GetVersionInfo`** `( Std_VersionInfoType *versioninfo )`	
Parameter	
`versioninfo`	Pointer to the memory location holding the version information of the StbM.
Return code	
–	-
Functional Description	
This API can be used to get the version information of the StbM.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. > This API is only available if enabled by the configuration parameter `StbMVersionInfoApi`.	
Expected Caller Context	
> No restriction	

Table 4-20 StbM_GetVersionInfo

4.2.2 StbM_InitMemory

Prototype	
`void` **`StbM_InitMemory`** `( void )`	
Parameter	
–	-
Return code	
–	-
Functional Description	
Initializes the global variables in case an initializing startup code is not used. This function sets the StbM into an uninitialized state.	
Particularities and Limitations	
> This function is synchronous. > This function is non-reentrant. > If this function is used it shall be called before any other StbM function after startup.	
Expected Caller Context	
> Task context	

Table 4-21 StbM_InitMemory

4.2.3 StbM_PreInit

Prototype	
`void` **`StbM_PreInit`** `( const StbM_ConfigType *ConfigPtr )`	
Parameter	
`ConfigPtr`	Pointer to the selected configuration set.
Return code	
–	-
Functional Description	
This API pre-initializes the StbM.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > This API should be called by the ECU State Manager during the startup phase. > This function has to be called before any other StbM service function is called (except StbM_InitMemory()).	
Expected Caller Context	
> Task context	

Table 4-22 StbM_PreInit

4.2.4 StbM_Init

Prototype	
`void StbM_Init ( const StbM_ConfigType *ConfigPtr )`	
Parameter	
`ConfigPtr`	Pointer to the selected configuration set.
Return code	
–	-
Functional Description	
This API initializes the StbM.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > This API should be called by the BSW Mode Manager during the startup phase. > This function has to be called before any other StbM service function is called (except StbM_InitMemory() and StbM_PreInit()).	
Expected Caller Context	
> Task context > In multi-partition use case this function has to be called for each partition context.	

Table 4-23 StbM_Init

4.2.5 StbM_PostInit

Prototype	
`void` **`StbM_PostInit`** `( void )`	
Parameter	
–	-
Return code	
–	-
Functional Description	
This API post-initializes the StbM.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > This API should be called by the BSW Mode Manager during the startup phase. > This function has to be called before any other StbM service function is called (except StbM_InitMemory(), StbM_PreInit() and StbM_Init()).	
Expected Caller Context	
> Task context > In multi-partition use case this function has to be called in context of the partition that has the highest ASIL level. If several partitions have the same ASIL level, it should be called from the first partition sorted by name.	

Table 4-24 StbM_PostInit

4.2.6 StbM_GetCurrentTime

Prototype	
`Std_ReturnType` **`StbM_GetCurrentTime`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeStampType *timeStampPtr, StbM_UserDataType *userDataPtr )`	
Parameter	
`timeBaseId`	The synchronized time-base, whose time is of interest.
`timeStampPtr`	Current time stamp that is valid at this time.
`userDataPtr`	User data of the time base.
Return code	
`Std_ReturnType`	E_OK: The time stamp of the time-base has been updated. E_NOT_OK: A DET error occurred or the EthIf is not available and the time stamp has not been updated.
Functional Description	
This API can be used to get the current time value of the submitted time-base in standard format.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-25 StbM_GetCurrentTime

Caution
It is the responsibility of the user to make sure that the StbM_GetCurrentTime() API is called inside an exclusive area together with the code which processes the new time value. Otherwise an interrupt or task can postpone further processing and the read value might be outdated depending on the length of the interruption.

If the time base uses an OS counter as local time clock an implementation mechanism that does not disable interrupts has to be used for the exclusive area.

4.2.7 StbM_GetCurrentTimeExtended

Prototype	
`Std_ReturnType` **`StbM_GetCurrentTimeExtended`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeStampExtendedType *timeStampPtr, StbM_UserDataType *userDataPtr )`	
Parameter	
`timeBaseId`	The synchronized time-base, whose time is of interest.
`timeStampPtr`	Current time stamp that is valid at this time.
`userDataPtr`	User data of the time base.
Return code	
`Std_ReturnType`	E_OK: The time stamp of the time-base has been updated. E_NOT_OK: A DET error occurred or the EthIf is not available and the time stamp has not been updated.
Functional Description	
This API can be used to get the current time value of the submitted time-base in extended format.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > This API is only available if enabled by the configuration parameter `StbMGetCurrentTimeExtendedAvailable`.	
Expected Caller Context	
> No restriction	

Table 4-26 StbM_GetCurrentTimeExtended

Caution
It is the responsibility of the user to make sure that the StbM_GetCurrentTimeExtended() API is called inside an exclusive area together with the code which processes the new time value. Otherwise an interrupt or task can postpone further processing and the read value might be outdated depending on the length of the interruption.

If the time base uses an OS counter as local time clock an implementation mechanism that does not disable interrupts has to be used for the exclusive area.

4.2.8 StbM_GetCurrentVirtualLocalTime

Prototype	
`Std_ReturnType` **`StbM_GetCurrentVirtualLocalTime`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_VirtualLocalTimeType *localTimePtr )`	
Parameter	
`timeBaseId`	The synchronized time-base, whose virtual local time is of interest.
`localTimePtr`	Current virtual local time value
Return code	
`Std_ReturnType`	E_OK: The time stamp has been updated. E_NOT_OK: A DET error occurred or the EthIf is not available and the time stamp has not been updated.
Functional Description	
This API can be used to the virtual local time from the submitted time-base.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-27 StbM_GetCurrentVirtualLocalTime

4.2.9 StbM_BusGetCurrentTime

Prototype	
`Std_ReturnType` **`StbM_BusGetCurrentTime`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeStampType *globalTimePtr, StbM_VirtualLocalTimeType *localTimePtr, StbM_UserDataType *userDataPtr)`	
Parameter	
`timeBaseId`	The synchronized time-base, whose time tuple is of interest.
`globalTimePtr`	The global time of the time-base
`localTimePtr`	The virtual local time of the time-base
`userDataPtr`	The user data of the time-base
Return code	
`Std_ReturnType`	E_OK: The time difference value has been updated. E_NOT_OK: A DET error occurred or the EthIf is not available and the time difference value has not been updated.
Functional Description	
This API can be used to get the time tuple (TL, TV) from the submitted time-base.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-28 StbM_BusGetCurrentTime

4.2.10 StbM_GetMainTime

Prototype	
`Std_ReturnType` **`StbM_GetMainTime`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeStampType *globalTimeSyncPtr, StbM_VirtualLocalTimeType *localTimeSyncPtr, StbM_RateDeviationType *rateDeviation)`	
Parameter	
`timeBaseId`	The synchronized time-base, whose main time tuple, status and rate deviation are of interest.
`globalTimeSyncPtr`	The synchronized global time value (part of the main time tuple) and the status of the time-base
`localTimeSyncPtr`	The synchronized virtual local time value (part of the main time tuple) of the time-base
`rateDeviation`	The rate deviation of the time-base
Return code	
`Std_ReturnType`	E_OK: The time stamps and rate deviation of the time base have been updated. E_NOT_OK: A DET error occurred and the time stamps and rate deviation have not been updated.
Functional Description	
This API can be used to get the main time tuple (TGsync, TVsync), status and rate deviation from the submitted time-base.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > This function returns 0 for the rate deviation if time correction is not configured for the time base or the current value is invalid at the time of the call.	
Expected Caller Context	
> No restriction	

Table 4-29 StbM_GetMainTime

Caution
It is the responsibility of the user to make sure that if the main time tuple retrieved from StbM_GetMainTime() is used to calculate the local time, the same clock counter referenced in config parameter `StbMLocalTimeHardware` is used for the local time, that was used for the main time tuple.

4.2.11 StbM_SetGlobalTime

Prototype	
`Std_ReturnType` **`StbM_SetGlobalTime`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeStampType *timeStampPtr, StbM_UserDataType *userDataPtr )`	
Parameter	
`timeBaseId`	The synchronized time-base, whose time is set.
`timeStampPtr`	New time stamp.
`userDataPtr`	New user data.
Return code	
`Std_ReturnType`	E_OK: The time stamp and user data of the time-base have been updated. E_NOT_OK: A DET error occurred or the EthIf is not available and the time stamp and user data have not been updated.
Functional Description	
This API allows the customers to set the new global time that has to be valid for the system.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-30 StbM_SetGlobalTime

4.2.12 StbM_SetUserData

Prototype	
`Std_ReturnType` **`StbM_SetUserData`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_UserDataType *userDataPtr )`	
Parameter	
`timeBaseId`	The synchronized time-base, whose user data is set.
`userDataPtr`	New user data.
Return code	
`Std_ReturnType`	E_OK: The user data of the time-base has been updated. E_NOT_OK: A DET error occurred and the user data has not been updated.
Functional Description	
This API allows the customers to set the new user data that has to be valid for the system.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-31 StbM_SetUserData

4.2.13 StbM_SetOffset

Prototype	
`Std_ReturnType` **`StbM_SetOffset`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeStampType *timeStampPtr )`	
Parameter	
`timeBaseId`	The offset time-base, whose offset time is set.
`timeStampPtr`	New offset time stamp.
Return code	
`Std_ReturnType`	E_OK: The offset time stamp of the time-base has been updated. E_NOT_OK: A DET error occurred or the EthIf is not available and the offset time stamp has not been updated.
Functional Description	
This API allows the customers and timebase provider modules to set the offset time that has to be valid for the system.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-32 StbM_SetOffset

Caution
The signature of StbM_SetOffset is implemented according AR 4.2.
User data handling is not considered.

4.2.14 StbM_GetOffset

Prototype	
`Std_ReturnType` **`StbM_GetOffset`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeStampType *timeStampPtr )`	
Parameter	
`timeBaseId`	The offset time-base, whose offset time is of interest.
`timeStampPtr`	Current offset time stamp.
Return code	
`Std_ReturnType`	E_OK: The offset time stamp has been updated. E_NOT_OK: A DET error occurred and the offset time stamp has not been updated.
Functional Description	
This API allows the timebase provider modules to get the current offset time.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-33 StbM_GetOffset

Caution
The signature of StbM_GetOffset is implemented according AR 4.2.
User data handling is not considered.

4.2.15 StbM_BusSetGlobalTime

Prototype	
`Std_ReturnType` **`StbM_BusSetGlobalTime`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeStampType *globalTimePtr, StbM_UserDataType *userDataPtr, StbM_MeasurementType *measureDataPtr, StbM_VirtualLocalTimeType *localTimePtr )`	
Parameter	
`timeBaseId`	The synchronized time-base, whose time is set.
`globalTimePtr`	New global time value.
`userDataPtr`	New user data.
`measureDataPtr`	New measurement data.
`localTimePtr`	Value of the virtual local time associated to the new global time.
Return code	
`Std_ReturnType`	E_OK: The time stamp and user data of the time-base have been updated. E_NOT_OK: A DET error occurred or the EthIf is not available and the time stamps have not been updated.
Functional Description	
This API allows the timebase provider modules to forward a new Global Time tuple to the StbM, which has been received from different busses.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-34 StbM_BusSetGlobalTime

4.2.16 StbM_GetRateDeviation

Prototype	
`Std_ReturnType` **`StbM_GetRateDeviation`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_RateDevationType *rateDeviation )`	
Parameter	
`timeBaseId`	The time-base, whose rate deviation is of interest.
`rateDevation`	Value of the current rate deviation of a time base.
Return code	
`Std_ReturnType`	E_OK: The rate deviation has been updated. E_NOT_OK: A DET error occurred and the rate deviation has not been updated.
Functional Description	
This API returns the value of the current rate deviation of a time base.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant.	
Expected Caller Context	
> No restriction	

Table 4-35 StbM_GetRateDeviation

4.2.17 StbM_SetRateCorrection

Prototype	
`Std_ReturnType` **`StbM_SetRateCorrection`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_RateDevationType rateDeviation )`	
Parameter	
`timeBaseId`	The time-base, whose rate deviation is set.
`rateDevation`	Value of the applied rate deviation.
Return code	
`Std_ReturnType`	E_OK: The rate correction of the time-base has been updated. E_NOT_OK: A DET error occurred or the EthIf is not available and the rate correction has not been updated.
Functional Description	
This API allows to set the rate of a synchronized time base (being either a pure local time base or not).	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant.	
Expected Caller Context	
> No restriction	

Table 4-36 StbM_SetRateCorrection

4.2.18 StbM_GetSyncTimeRecordHead

Prototype	
`Std_ReturnType` **`StbM_GetSyncTimeRecordHead`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_SyncRecordTableHeadType *syncRecordTableHead )`	
Parameter	
`timeBaseId`	The time-base, whose header is of interest.
`syncRecordTableHead`	Header of the recorded snapshot data.
Return code	
`Std_ReturnType`	E_OK: The record table header has been updated. E_NOT_OK: A DET error occurred and the record table header has not been updated.
Functional Description	
This API allows the customers to access the recorded snapshot data header of the table belonging to the Synchronized Time Base.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant.	
Expected Caller Context	
> No restriction	

Table 4-37 StbM_GetSyncTimeRecordHead

4.2.19 StbM_GetOffsetTimeRecordHead

Prototype	
`Std_ReturnType` **`StbM_GetOffsetTimeRecordHead`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_OffsetRecordTableHeadType *offsetRecordTableHead )`	
Parameter	
`timeBaseId`	The time-base, whose header is of interest.
`offsetRecordTableHead`	Header of the recorded snapshot data.
Return code	
`Std_ReturnType`	E_OK: The record table header has been updated. E_NOT_OK: A DET error occurred and the record table header has not been updated.
Functional Description	
This API allows the customers to access the recorded snapshot data header of the table belonging to the Offset Time Base.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant.	
Expected Caller Context	
> No restriction	

Table 4-38 StbM_GetOffsetTimeRecordHead

4.2.20 StbM_StartTimer

Prototype	
`Std_ReturnType` **`StbM_StartTimer`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_CustomerIdType customerId, StbM_TimeStampType expireTime )`	
Parameter	
`timeBaseId`	ID of the Time Base, relative to which the timer shall be started
`customerId`	ID of the notification customer
`expireTime`	Time value relative to current Time Base value of the Notification Customer, when the Timer shall expire
Return code	
`Std_ReturnType`	E_OK: Starting the timer was successful. E_NOT_OK: Starting timer was not successful.
Functional Description	
This API sets a time value which the Time Base value is compared against.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reenttrant.	
Expected Caller Context	
> No restriction	

Table 4-39 StbM_StartTimer

4.2.21 StbM_NotificationFunction

Prototype	
`void` **`StbM_NotificationFunction`** `( void )`	
Parameter	
–	-
Return code	
–	-
Functional Description	
This API calls the callback functions for time notification customers. This function only works with MICROSAR Classic RTE.	
Particularities and Limitations	
> This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> Task context	

Table 4-40 StbM_NotificationFunction

4.2.22 StbM_UpdateGlobalTime

Prototype	
`Std_ReturnType` **`StbM_UpdateGlobalTime`** `( StbM_SynchronizedTimeBaseType timeBaseId, const StbM_TimeStampType *timeStamp, const StbM_UserDataType *userData )`	
Parameter	
`timeBaseId`	The synchronized time-base, whose time is updated.
`timeStamp`	New time stamp.
`userData`	New user data.
Return code	
`Std_ReturnType`	E_OK: The time stamp and user data of the time-base have been updated. E_NOT_OK: A DET error occurred or the EthIf is not available and the time stamp and user data have not been updated.
Functional Description	
This API allows the customers to set the new global time that has to be valid for the system. Using UpdateGlobalTime will not lead to an immediate transmission of the global time.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-41 StbM_UpdateGlobalTime

4.2.23 StbM_TriggerTimeTransmission

Prototype	
`Std_ReturnType` **`StbM_TriggerTimeTransmission`** `( StbM_SynchronizedTimeBaseType timeBaseId )`	
Parameter	
`timeBaseId`	The synchronized time-base, whose immediate transmission shall be triggered.
Return code	
`Std_ReturnType`	E_OK: The immediate transmission of the time-base has been triggered. E_NOT_OK: A DET error occurred and the immediate time transmission has not been triggered.
Functional Description	
This API allows the customers to force the Timesync modules to transmit the current time base again.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-42 StbM_TriggerTimeTransmission

4.2.24 StbM_GetTimeBaseUpdateCounter

Prototype	
`uint8` **`StbM_GetTimeBaseUpdateCounter`** `( StbM_SynchronizedTimeBaseType timeBaseId )`	
Parameter	
`timeBaseId`	The synchronized time-base, whose update counter is of interest.
Return code	
`uint8`	Current counter value of the time base.
Functional Description	
This API allows the Timesync modules to detect, whether a time base should be transmitted immediately in the subsequent <Bus>TSyn_MainFunction() cycle.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-43 StbM_GetTimeBaseUpdateCounter

4.2.25 StbM_GetTimeLeap

Prototype	
`Std_ReturnType` **`StbM_GetTimeLeap`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeDiffType *timeJump )`	
Parameter	
`timeBaseId`	The time-base, whose time leap is of interest.
`timeJump`	Value of the last time leap of a time base.
Return code	
`Std_ReturnType`	E_OK: Time leap is valid. E_NOT_OK: A DET error occurred or no time leap occured or time leap is out of range.
Functional Description	
This API returns the value of the last time leap, if StbMTimeLeapFuture/PastThreshold is exceeded.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant.	
Expected Caller Context	
> No restriction	

Table 4-44 StbM_ GetTimeLeap

4.2.26 StbM_GetTimeBaseStatus

Prototype	
`Std_ReturnType` **`StbM_GetTimeBaseStatus`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_TimeBaseStatusType *syncTimeBaseStatus, StbM_TimeBaseStatusType *offsetTimeBaseStatus )`	
Parameter	
`timeBaseId`	The time-base, whose status is of interest.
`syncTimeBaseStatus`	Status of the Synchronized Time Base.
`offsetTimeBaseStatus`	Status of the Offset Time Base.
Return code	
`Std_ReturnType`	E_OK: Status is valid. E_NOT_OK: A DET error occurred or no status could be retrieved.
Functional Description	
This API returns the status of a Time Base. For Offset Time Bases the status of the underlying Synchronized Time Base is also returned.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant.	
Expected Caller Context	
> No restriction	

Table 4-45 StbM_ GetTimeBaseStatus

4.2.27 StbM_CloneTimeBase

Prototype	
`Std_ReturnType` **`StbM_CloneTimeBase`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_CloneConfigType cloneCfg, StbM_TimeBaseStatusType statusMask, StbM_TimeBaseStatusType statusValue )`	
Parameter	
`timeBaseId`	Destination Time Base for cloning
`cloneCfg`	Refines how Source Time Base is cloned to destination
`statusMask`	Status flags mask for definition of relevant status flags
`statusValue`	Status flags value define whether cloning shall take place
Return code	
`Std_ReturnType`	E_OK: Time base cloning was performed successfully. E_NOT_OK: One or more of the following conditions occurred: > A DET error > Time base cloning failed > The masked source time base status does not match the `statusValue` parameter > A deferred clone request was placed for a pure local or master source time base
Functional Description	
This API allows the cloning of time base data (current time, user data, rate correction) from one time base to another.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function behaves synchronously/asynchronously depending on the value of the `cloneCfg` parameter. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-46 StbM_CloneTimeBase

Caution
It is the responsibility of the user to make sure that if an immediate cloning is requested in call of StbM_CloneTimeBase(), the global time of the source time base is set prior to calling StbM_CloneTimeBase(). Otherwise the cloned time value will be incorrect.

4.2.28 StbM_EthSetSlaveTimingData

Prototype	
`Std_ReturnType` **`StbM_EthSetSlaveTimingData`** `( StbM_SynchronizedTimeBaseType timeBaseId, const StbM_EthTimeSlaveMeasurementType* measureDataPtr)`	
Parameter	
`timeBaseId`	The time-base, whose measurement data is set.
`measureDataPtr`	New measurement data.
Return code	
`Std_ReturnType`	E_OK: The slave timing data of the time-base has been updated. E_NOT_OK: A DET error occurred and the slave timing data has not been updated.
Functional Description	
This API allows the EthTSyn module to forward Ethernet specific slave timing data to the StbM.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-47 StbM_ EthSetSlaveTimingData

4.2.29 StbM_EthSetMasterTimingData

Prototype	
`Std_ReturnType` **`StbM_EthSetMasterTimingData`** `( StbM_SynchronizedTimeBaseType timeBaseId, const StbM_EthTimeMasterMeasurementType* measureDataPtr)`	
Parameter	
`timeBaseId`	The time-base, whose measurement data is set.
`measureDataPtr`	New measurement data.
Return code	
`Std_ReturnType`	E_OK: The master timing data of the time-base has been updated. E_NOT_OK: A DET error occurred and the master timing data has not been updated.
Functional Description	
This API allows the EthTSyn module to forward Ethernet specific master timing data to the StbM.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-48 StbM_ EthSetMasterTimingData

4.2.30 StbM_EthSetPdelayInitiatorData

Prototype	
`Std_ReturnType` **`StbM_EthSetPdelayInitiatorData`** `( StbM_SynchronizedTimeBaseType timeBaseId, const StbM_PdelayInitiatorMeasurementType* measureDataPtr)`	
Parameter	
`timeBaseId`	The time-base, whose measurement data is set.
`measureDataPtr`	New measurement data.
Return code	
`Std_ReturnType`	E_OK: The pDelay initiator data of the time-base has been updated. E_NOT_OK: A DET error occurred and the pDelay initiator data has not been updated.
Functional Description	
This API allows the EthTSyn module to forward Ethernet specific pDelay initiator data to the StbM.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-49 StbM_ EthSetPdelayInitiatorData

4.2.31 StbM_EthSetPdelayResponderData

Prototype	
`Std_ReturnType` **`StbM_EthSetPdelayResponderData`** `( StbM_SynchronizedTimeBaseType timeBaseId, const StbM_PdelayResponderMeasurementType* measureDataPtr)`	
Parameter	
`timeBaseId`	The time-base, whose measurement data is set.
`measureDataPtr`	New measurement data.
Return code	
`Std_ReturnType`	E_OK: The pDelay responder data of the time-base has been updated. E_NOT_OK: A DET error occurred and the pDelay responder data has not been updated.
Functional Description	
This API allows the EthTSyn module to forward Ethernet specific pDelay responder data to the StbM.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-50 StbM_ EthSetPdelayResponderData

4.2.32 StbM_GetMasterConfig

Prototype	
`Std_ReturnType` **`StbM_GetMasterConfig`** `( StbM_SynchronizedTimeBaseType timeBaseId, StbM_MasterConfigType* masterConfig )`	
Parameter	
`timeBaseId`	The time-base, whose availability of system wide master functionality is of interest.
`masterConfig`	Indicates if system wide master functionality is supported.
Return code	
`Std_ReturnType`	E_OK: The masterConfig data of the time-base has been updated. E_NOT_OK: A DET error occurred and the masterConfig data has not been updated.
Functional Description	
This API indicates if the functionality for a system wide master (e.g. StbM_SetGlobalTime) for a given time base is available or not.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-51 StbM_GetMasterConfig

4.2.33 StbM_MainFunction

Prototype	
void **StbM_MainFunction** (void)	
Parameter	
–	-
Return code	
–	-
Functional Description	
This function will be called cyclically by a task body provided by the BSW Scheduler. It will invoke the triggered customers and synchronize the referenced OS ScheduleTables.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> Task context	

Table 4-52 StbM_MainFunction

4.3 Services used by StbM

In the following table services provided by other components, which are used by the StbM are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
Det	Det_ReportError
Os	GetCounterValue GetElapsedValue GetScheduleTableStatus SyncScheduleTable SetEvent ActivateTask GetApplicationID
EthIf	EthIf_GetCurrentTime EthIf_GetControllerMode
RTE / SchM	SchM_Enter_StbM_STBM_EXCLUSIVE_AREA_0 SchM_Exit_StbM_STBM_EXCLUSIVE_AREA_0 SchM_Enter_StbM_STBM_EXCLUSIVE_AREA_1 SchM_Exit_StbM_STBM_EXCLUSIVE_AREA_1
Gpt	Gpt_StartTimer Gpt_GetTimeElapsed Gpt_EnableNotification
EcuM	EcuM_BswErrorHook

Table 4-53 Services used by the StbM

4.4 Configurable Interfaces

4.4.1 Notifications

At its configurable interfaces the StbM defines notifications that can be mapped to callback functions provided by other modules. The mapping is not statically defined by the StbM but can be performed at configuration time. The function prototypes that can be used for the configuration have to match the appropriate function prototype signatures, which are described in the following sub-chapters.

4.4.1.1 SyncTimeRecordBlockCallback

Prototype	
`Std_ReturnType` **`SyncTimeRecordBlockCallback<TimeBase>`** `( StbM_SyncRecordTableBlockType *syncRecordTableBlock )`	
Parameter	
`syncRecordTableBlock`	Block of the table.
Return code	
`Std_ReturnType`	E_OK: Table access done. E_NOT_OK: Table contains no data or access invalid.
Functional Description	
This function provides a recorded snapshot data block of the measurement data table belonging to the Synchronized Time Base.	
Particularities and Limitations	
> This function is synchronous. > This function is non-reentrant.	
Call context	
> Task context	

Table 4-54 SyncTimeRecordBlockCallback

4.4.1.2 OffsetTimeRecordBlockCallback

Prototype	
`Std_ReturnType` **`OffsetTimeRecordBlockCallback<TimeBase>`** `( StbM_OffsetRecordTableBlockType *offsetRecordTableBlock )`	
Parameter	
`offsetRecordTableBlock`	Block of the table.
Return code	
`Std_ReturnType`	E_OK: Table access done. E_NOT_OK: Table contains no data or access invalid.
Functional Description	
This function provides a recorded snapshot data block of the measurement data table belonging to the Offset Time Base.	

Particularities and Limitations
> This function is synchronous. > This function is non-reentrant.
Call context
> Task context

Table 4-55 OffsetTimeRecordBlockCallback

4.4.1.3 StatusNotificationCallback

Prototype	
`Std_ReturnType` **`StatusNotificationCallback<TimeBase>`** `(StbM_TimeBaseNotificationType eventNotification )`	
Parameter	
`eventNotification`	Holds the notification bits for the different Time Base related events
Return code	
`Std_ReturnType`	E_OK: successful. E_NOT_OK: failed.
Functional Description	
The callback notifies the customers, when a <TimeBase> related event occurs, which is enabled by the notification mask.	
Particularities and Limitations	
> This function is synchronous. > This function is non-reentrant.	
Call context	
> Task context	

Table 4-56 StatusNotificationCallback

4.4.1.4 <Customer>_TimeNotificationCallback

Prototype	
`Std_ReturnType` **`<Customer>_TimeNotificationCallback<TimeBase>`** `(StbM_TimeDiffType deviationtime )`	
Parameter	
`deviationTime`	Deviation between actual time value captured when callback is called and expiration time.
Return code	
`Std_ReturnType`	E_OK: successful. E_NOT_OK: failed.

Functional Description
This callback notifies the <Customer>, when a Time Base reaches the time value set by StbM_StartTimer for the <TimeBase>.
Particularities and Limitations
> This function is synchronous. > This function is non-reentrant.
Call context
> Task context

Table 4-57 <Customer>_TimeNotificationCallback

4.5 Service Ports

The service port names use placeholders. Their meaning is described in the following table.

Placeholder	Meaning
TB	Name of the time base
C	Name of the time notification customer

Table 4-58 Meaning of placeholders in service port names

4.5.1 Client Server Interface

A client server interface is related to a Provide Port at the server side and a Require Port at client side.

4.5.1.1 Provide Ports on StbM Side

At the Provide Ports of the StbM the API functions described in 4.2 that are invoked via Operations are available as Runnable Entities. The mapping from a SWC client call to an Operation is performed by the RTE. In this mapping the RTE adds Port Defined Argument Values to the client call of the SWC, if configured.

The following sub-chapters present the Provide Ports defined for the StbM and the Operations defined for the Provide Ports, the API functions related to the Operations and the Port Defined Argument Values to be added by the RTE.

4.5.1.1.1 GlobalTime_Master_<TB>

This provide port is only available, if the configuration parameter `StbMIsSystemWideGlobalTimeMaster` or `StbMAllowSystemWideGlobalTimeMaster` is enabled for the appropriate time base and the `StbMSynchronizedTimeBaseIdentifier` of the time base is less than 128.

The operation CloneTimeBase is only available if the time base has an `StbMSourceTimeBase` configured and is not an offset time base, i.e. its `StbMSynchronizedTimeBaseIdentifier` is not in the range 16 till 31 .

The operation GetMasterConfig is only available, if the configuration parameter `StbMAllowSystemWideGlobalTimeMaster` is available for the appropriate time base.

The operation SetOffset is only available, if the time base is an offset time base, i.e. it has an `StbMSynchronizedTimeBaseIdentifier` in the range 16 till 31.

The operation TriggerTimeTransmission is only available, if the `StbMSynchronizedTimeBaseIdentifier` of the time base is less than 32 or refers to a synchronized time base with `StbMSynchronizedTimeBaseIdentifier` between 32 and 127.

Operation	API Function	Port Defined Argument Values
CloneTimeBase	StbM_CloneTimeBase	StbM_SynchronizedTimeBaseType 0..n
GetMasterConfig	StbM_GetMasterConfig	StbM_SynchronizedTimeBaseType 0..n
SetGlobalTime	StbM_SetGlobalTime	StbM_SynchronizedTimeBaseType 0..n
SetOffset	StbM_SetOffset	StbM_SynchronizedTimeBaseType 0..n
SetUserData	StbM_SetUserData	StbM_SynchronizedTimeBaseType 0..n
SetRateCorrection	StbM_SetRateCorrection	StbM_SynchronizedTimeBaseType 0..n
TriggerTimeTransmission	StbM_TriggerTimeTransmission	StbM_SynchronizedTimeBaseType 0..n
UpdateGlobalTime	StbM_UpdateGlobalTime	StbM_SynchronizedTimeBaseType 0..n

Table 4-59 GlobalTime_Master_<TB>

4.5.1.1.2 GlobalTime_Slave_<TB>

This provide port is only available, if the time base has an `StbMSynchronizedTimeBaseIdentifier` less than 128.

The operation GetCurrentTimeExtended is only available, if the parameter `StbMGetCurrentTimeExtendedAvailable` is enabled.

The operation GetSyncTimeRecordHead is only available, if the parameter `StbMTimeRecordingSupport` is enabled and the time base is a synchronized time base, i.e. it has an `StbMSynchronizedTimeBaseIdentifier` less than 16 or between 32 and 127 with a reference from a TSyn Module.

The operation GetOffsetTimeRecordHead is only available, if the parameter `StbMTimeRecordingSupport` is enabled and the time base is an offset time base, i.e. it has an `StbMSynchronizedTimeBaseIdentifier` in the range 16 till 31.

The operation GetTimeLeap is only available, if the time base has an `StbMSynchronizedTimeBaseIdentifier` less than 32 or refers to a synchronized time base with `StbMSynchronizedTimeBaseIdentifier` between 32 and 127.

Operation	API Function	Port Defined Argument Values
GetCurrentTime	StbM_GetCurrentTime	StbM_SynchronizedTimeBaseType 0..n
GetCurrentTimeExtended	StbM_GetCurrentTimeExtended	StbM_SynchronizedTimeBaseType 0..n
GetRateDeviation	StbM_GetRateDeviation	StbM_SynchronizedTimeBaseType 0..n
GetSyncTimeRecordHead	StbM_GetSyncTimeRecordHead	StbM_SynchronizedTimeBaseType 0..n
GetOffsetTimeRecordHead	StbM_GetOffsetTimeRecordHead	StbM_SynchronizedTimeBaseType 0..n

Operation	API Function	Port Defined Argument Values
GetTimeLeap	StbM_GetTimeLeap	StbM_SynchronizedTimeBaseType 0..n
GetTimeBaseStatus	StbM_GetTimeBaseStatus	StbM_SynchronizedTimeBaseType 0..n

Table 4-60 GlobalTime_Slave_<TB>

4.5.1.1.3 StartTimer_<TB>_<C>

This provide port is only available for each notification customer of the time base, if the `StbMSynchronizedTimeBaseIdentifier` of the time base is less than 128.

Operation	API Function	Port Defined Argument Values
StartTimer	StbM_StartTimer	StbM_SynchronizedTimeBaseType 0..n StbM_CustomerIdType 0..n

Table 4-61 StartTimer_<TB>_<C>

4.5.1.2 Require Ports on StbM Side

At its Require Ports the StbM calls Operations. These Operations have to be provided by the SWCs by means of Runnable Entities. These Runnable Entities implement the callback functions expected by the StbM.

The following sub-chapters present the Require Ports defined for the StbM, the Operations that are called from the StbM and the related Notifications, which are described in chapter 4.4.

4.5.1.2.1 StbM_MeasurementNotification_<TB>

This required port is only available, if the parameter `StbMTimeRecordingSupport` is enabled and the `StbMSynchronizedTimeBaseIdentifier` of the time base is less than 32 or refers to a synchronized time base with `StbMSynchronizedTimeBaseIdentifier` between 32 and 127.

The operation SetSyncTimeRecordTable is only available, if the time base is a synchronized time base, i.e. it has an `StbMSynchronizedTimeBaseIdentifier` less than 16 or between 32 and 127 with a reference from a TSyn Module.

The operation SetOffsetTimeRecordTable is only available, if the time base is an offset time base, i.e. it has an `StbMSynchronizedTimeBaseIdentifier` in the range 16 till 31.

Operation	Notification
SetSyncTimeRecordTable	SyncTimeRecordBlockCallback
SetOffsetTimeRecordTable	OffsetTimeRecordBlockCallback

Table 4-62 StbM_MeasurementNotification_<TB>

4.5.1.2.2 GlobalTime_TimeEvent_<TB>_<C>

This required port is only available for each notification customer of the time base, if the `StbMSynchronizedTimeBaseIdentifier` of the time base is less than 128.

Operation	Notification
NotifyTime	TimeNotificationCallback

Table 4-63 GlobalTime_TimeEvent_<TB>_<C>

4.5.1.2.3 TimeBaseProviderNotification_Eth_<TB>

This required port is only available, if the configuration container `StbMTimeValidation` exists for the appropriate time base and the `StbMSynchronizedTimeBaseIdentifier` of the time base is less than 16 or between 32 and 127, and the time base is referenced by an EthTSyn time domain.

The operation SetMasterTimingData is only available, if the time base is used in the role master.

The operation SetPdelayInitiatorData is only available, if the time base is used in the role slave.

The operation SetPdelayResponderData is only available, if the time base is used in the role master.

The operation SetSlaveTimingData is only available, if the time base is used in the role slave.

4.5.2 Sender-Receiver Interface

The Sender-Receiver interfaces and ports described here are used to generate the RTE between application software components and the StbM.

4.5.2.1 Provided Ports on StbM side

4.5.2.1.1 GlobalTime_StatusEvent_<TB>

The StbM is able to send status change events via Provided Sender-Receiver Ports. This provide port is only available, if the time base has an `StbMSynchronizedTimeBaseIdentifier` less than 128 and a status notification mask configured.

The related Provided Ports are named as

- GlobalTime_StatusEvent_<TB>

The Sender-Receiver-Interface is named as

- StatusNotification

Further, the belonging data element is specified as

- eventNotification

5 Configuration

In the StbM the attributes can be configured with the following tools:

> Configuration in DaVinci Configurator

5.1 Configuration Variants

The StbM supports the configuration variants

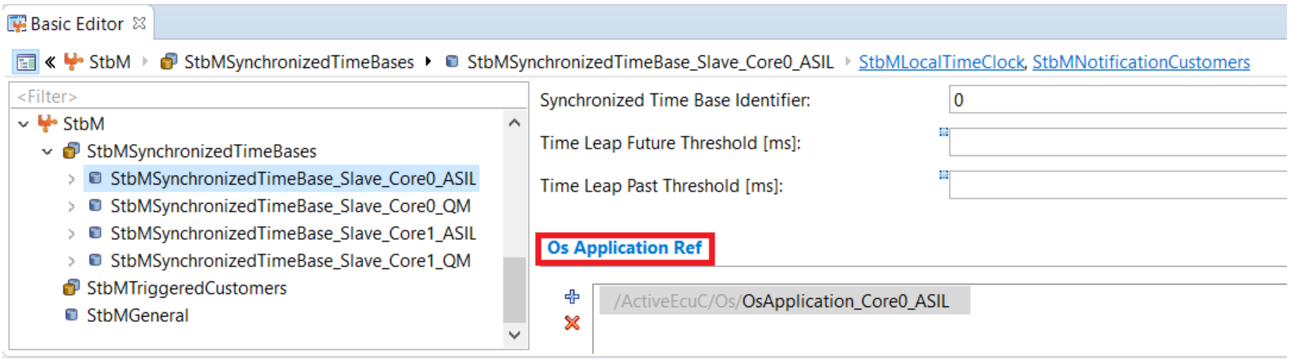
> `VARIANT-PRE-COMPILE`

> `VARIANT-POST-BUILD-SELECTABLE`

The configuration classes of the StbM parameters depend on the supported configuration variants. For their definitions please see the StbM_bswmd.arxml file.

5.2 Multi-Partition configuration

If the StbM shall be used in multiple partitions, the user has to configure for each time base the OS applications the time base shall be used in.


Basic Editor
StbM
StbMSynchronizedTimeBases
StbMSynchronizedTimeBase_Slave_Core0_ASIL
StbMLocalTimeClock, StbMNotificationCustomers
<Filter>
StbM
StbMSynchronizedTimeBases
StbMSynchronizedTimeBase_Slave_Core0_ASIL
StbMSynchronizedTimeBase_Slave_Core0_QM
StbMSynchronizedTimeBase_Slave_Core1_ASIL
StbMSynchronizedTimeBase_Slave_Core1_QM
StbMTriggeredCustomers
StbMGeneral
Os Application Ref

Synchronized Time Base Identifier: 0

Time Leap Future Threshold [ms]:

Time Leap Past Threshold [ms]:

/ActiveEcuC/Os/OsApplication_Core0_ASIL

Figure 5-1 Configuration of partitions for time bases

The StbM generator will then create partition specific service components for each referenced OS application with service ports for each time base that references the OS application.

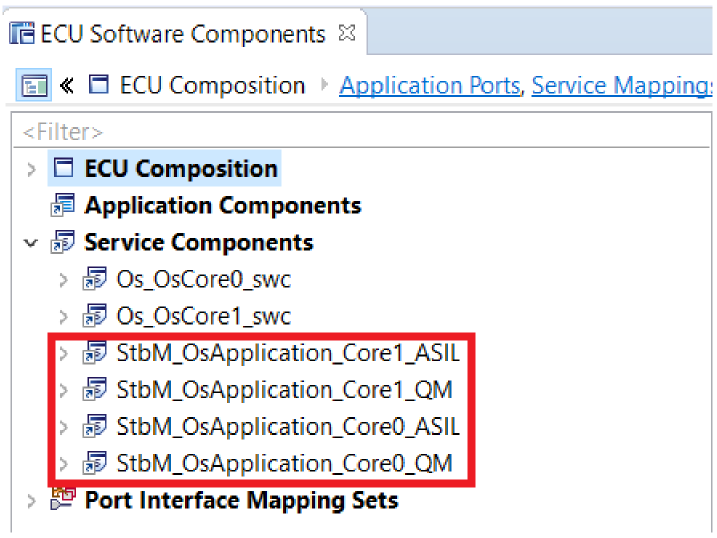

ECU Software Components
ECU Composition ▸ Application Ports, Service Mapping:
<Filter>
- ECU Composition
- Application Components
- Service Components
  - Os_OsCore0_swc
  - Os_OsCore1_swc
  - StbM_OsApplication_Core1_ASIL
  - StbM_OsApplication_Core1_QM
  - StbM_OsApplication_Core0_ASIL
  - StbM_OsApplication_Core0_QM
- Port Interface Mapping Sets

Figure 5-2 Partition specific StbM service components

Furthermore in this use case partition specific files and MainFunctions are generated and StbM_Init() needs to be called for every partition the StbM runs in. See chapters 2.2 and 2.4.1 for details.

5.2.1 Multi-Partition usage of StbM service ports

The following table describes the supported connections of StbM service ports in Multi-Partition configurations. See also Figure 5-3 and Figure 5-4 for an illustration of the supported use cases.

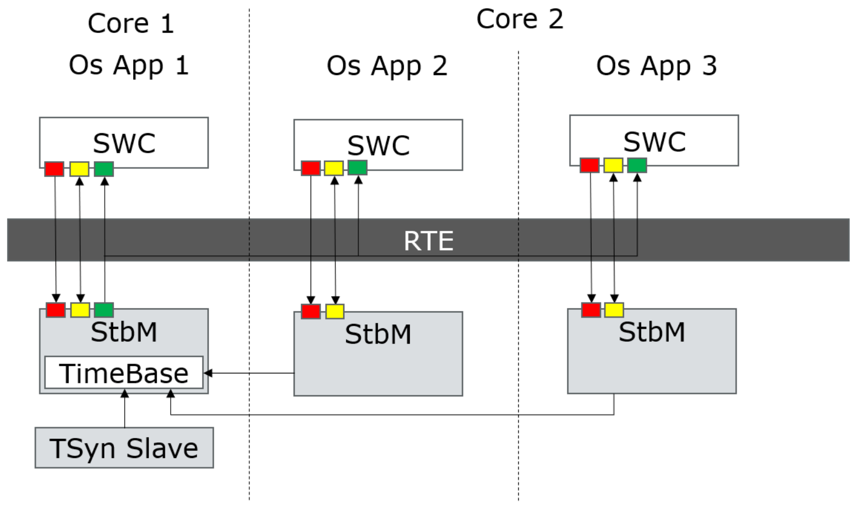
Service port	Possible connections for access from SWC	
GlobalTime_Master_<TB>	Can be connected in the partition of the SWC that sets the global time (only one master SWC is possible)	🟦
GlobalTime_Slave_<TB>	Can be connected in each partition	🟥
StartTimer_<TB>_<C>	Can be connected in the partition of the SWC with the notification customer	🟨
GlobalTime_TimeEvent_<TB>_<C>		
GlobalTime_StatusEvent_<TB>	Can be connected directly only in the partition the time base is mapped to, SWCs in other partitions can also connect to the service port in the partition of the time base	🟩
StbM_MeasurementNotification_<TB>		
TimeBaseProviderNotification_Eth_<TB>		

Table 5-1 Supported service port connections in Multi-Partition configurations

5.2.2 Multi-Partition usage of slave time bases

A slave time base is automatically mapped to the partition that is referenced by the PDU or, if the PDU has no partition mapping, to the partition the TSyn module runs in.

The following Figure 5-3 illustrates the partition mapping and possible service port connections for slave time bases.



🟥 GlobalTime_Slave_<TB> port

🟨 StartTimer_<TB>_<C> or GlobalTime_TimeEvent_<TB>_<C> port

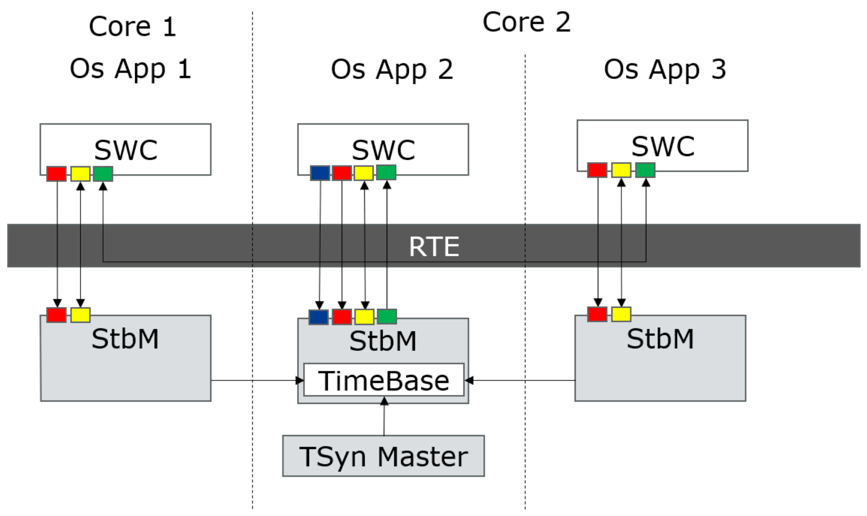
🟩 GlobalTime_StatusEvent_<TB>, StbM_MeasurementNotification_<TB> or TimeBaseProviderNotification_Eth_<TB> port

Figure 5-3 Multi-Partition usage of slave time base

5.2.3 Multi-Partition usage of master time bases

A master time base is automatically mapped to the partition where the user connects the master port. If the master port is not connected in any partition, the time base is mapped per default to the partition that has the highest ASIL level.

The following Figure 5-4 illustrates the partition mapping and possible service port connections for master time bases.



🟦 GlobalTime_Master_<TB> port

🟥 GlobalTime_Slave_<TB> port

🟨 StartTimer_<TB>_<C> or GlobalTime_TimeEvent_<TB>_<C> port

🟩 GlobalTime_StatusEvent_<TB> or TimeBaseProviderNotification_Eth_<TB> port

Figure 5-4 Multi-Partition usage of master time base

6 Glossary and Abbreviations

6.1 Glossary

Term	Description
DaVinci Configurator	Configuration and generation tool for MICROSAR Classic components

Table 6-1 Glossary

6.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
CAN	Controller Area Network
CanTSyn	Time Synchronization over CAN
DET	Development Error Tracer
ECU	Electronic Control Unit
EthIf	Ethernet Interface
EthTSyn	Time Synchronization over Ethernet
FR	FlexRay
FrTSyn	Time Synchronization over FlexRay
Gpt	General Purpose Timer
HIS	Hersteller Initiative Software
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
OS	Operating System
RTE	Runtime Environment
SchM	Schedule Manager
SRS	Software Requirement Specification
StbM	Synchronized Time-Base Manager
SWC	Software Component
SWS	Software Specification

Table 6-2 Abbreviations

7 Contact

Visit our website for more information on

> News
> Products
> Demo software
> Support
> Training data
> Addresses

www.vector.com