Document Title	Acceptance Test Specification of Global Time Synchronization
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	841
Document Classification	Auxiliary

Document Status	Final
Part of AUTOSAR Standard	Acceptance Tests for Classic Platform
Part of Standard Release	1.2.0

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2016-12-15	1.2.0	AUTOSAR Release Management	Initial release, including test suites on • RS_BRF_01660– Global Time Synchronization over CAN • RS_BRF_01660 – Global Time Synchronization over FlexRay • RS_BRF_01660 – Global Time Synchronization over Multiple Bus

Disclaimer

Advice for users

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

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

Table of Contents

1 Acronyms and abbreviations

Abbreviation / Acronym:	*Description:*
AT	Acceptance Test
CAN	Controller Area Network
ECU	Electronic Control Unit
LT	Lower Tester
PCO	Point of Control and Observation
Rx	Reception
SUT	System Under Test
SWC	Software Component
TCP	Test Coordination Procedures
Tx	Transmission
UT	Upper Tester

2 Related Documentation

2.1 Input documents

[1] Specification of Synchronized Time-Base Manager
AUTOSAR_SWS_SynchronizedTimeBaseManager.pdf

[2] Specification of Time Synchronization over CAN
AUTOSAR_SWS_TimeSyncOverCAN.pdf

[3] Specification of Time Synchronization over FlexRay
AUTOSAR_SWS_TimeSyncOverFlexRay.pdf

[4] Specification of Operating System
AUTOSAR_SWS_OS.pdf

[5] Specification of Basic Software Mode Manager
AUTOSAR_SWS_BSWModeManager.pdf

[6] Specification of CAN Interface
AUTOSAR_SWS_CANInterface.pdf

[7] Specification of FlexRay Interface
AUTOSAR_SWS_FlexRayInterface.pdf

[8] Specification of RTE
AUTOSAR_SWS_RTE.pdf

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

[10] Requirements on Acceptance Tests
AUTOSAR_ATR_Requirements.pdf

[11] Requirements on AUTOSAR Features
AUTOSAR_RS_Features.pdf

[12] System Template
AUTOSAR_TPS_SystemTemplate.pdf

3 Scope

The following test cases are used to verify the correct behavior of all the Global Time Synchronization features.

Each test case documents for which releases of the AUTOSAR software specification it can be used:

- When test cases are known to be applicable for a release, this is mentioned in the “AUTOSAR Releases” field of the test case specifications.
  You can find a summary of the applicability of all test cases to the software specification releases in the “AUTOSAR_TR_ATSReleaseApplicability” document.
- When test cases are known to require adaptations (in their configuration requirements or test sequences), this is mentioned in the “Needed Adaptation to other Releases” field of the test case specifications.

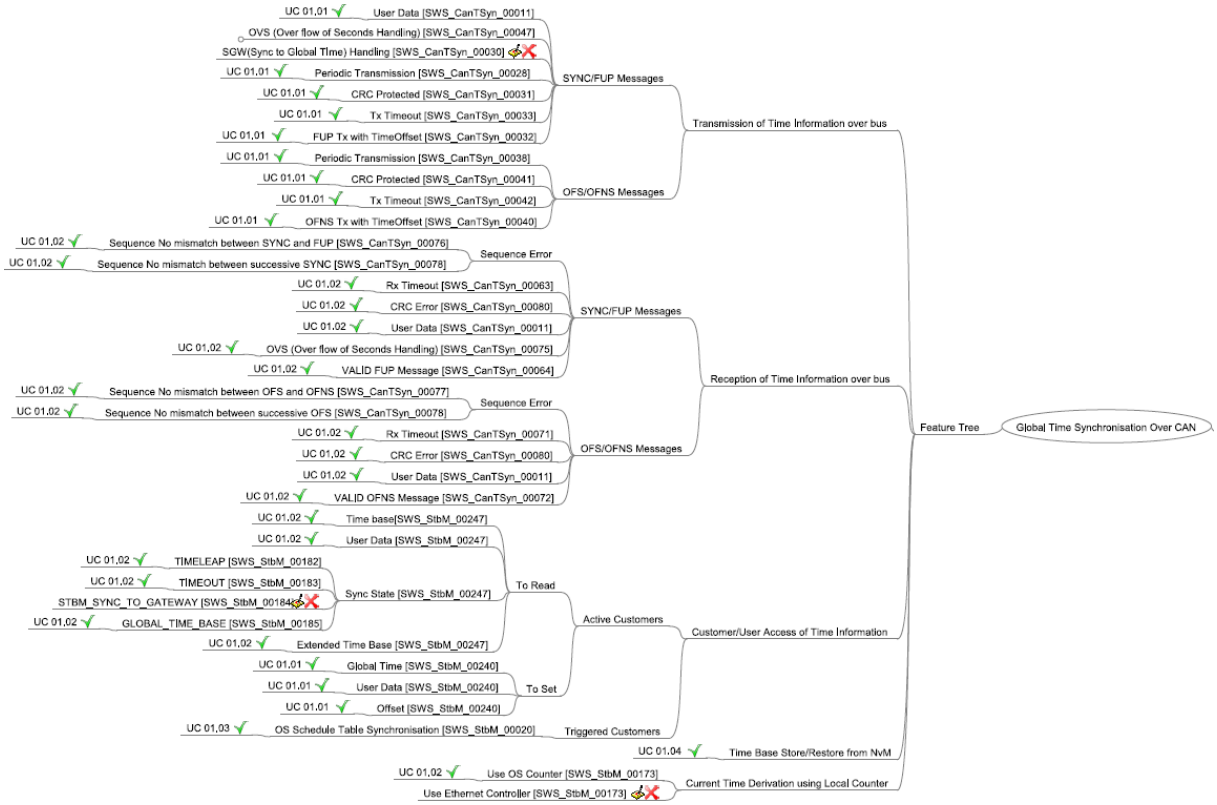
4 RS_BRF_01660 - Global Time Synchronization over CAN

4.1 General Test Objective and Approach

This Test Specification intends to cover the Global Time Synchronization feature of StbM and CanTSyn as described in the AUTOSAR Feature [RS_BRF_01660].

The tests use a test bench environment and Embedded Software Components that use the feature.

This test case document has been established to cover the following features:



Below Features are not tested in this test suite:

- SGW(Sync to Global Time) Handling [SWS_CanTSyn_00030] and STBM_SYNC_TO_GATEWAY [SWS_StbM_00184]: Feature are tested in the test suite 'Global Time Synchronization over Multiple Bus'.
- Use Ethernet Controller [SWS_StbM_00173]: Testing over Ethernet bus is out of scope of this ATS.

This specification gives the description of required tests environments (test bench, uses case, configuration files) and detailed tests cases for executing tests.

4.1.1 Test System

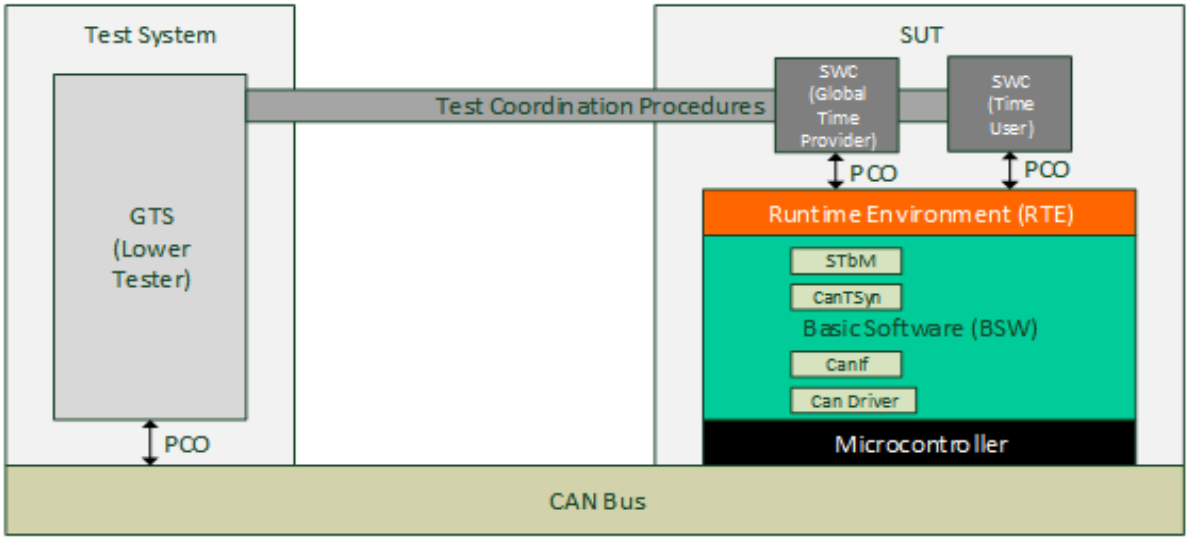
4.1.1.1 Overview on Architecture

In order to cover the required features / sub-features coverage, the environment has been separated in several use cases.

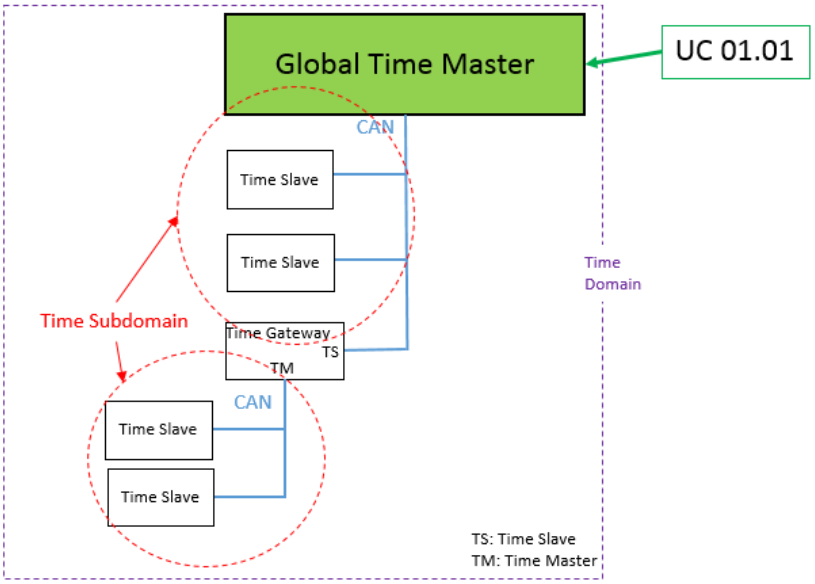
Test Cases are derived based on below use cases

4.1.1.1.1 UC 01.01: Global Time Master over CAN

SUT acts as Global time Master and Sets time base, offset time base and Trigger for transmission of Synchronization over CAN bus.

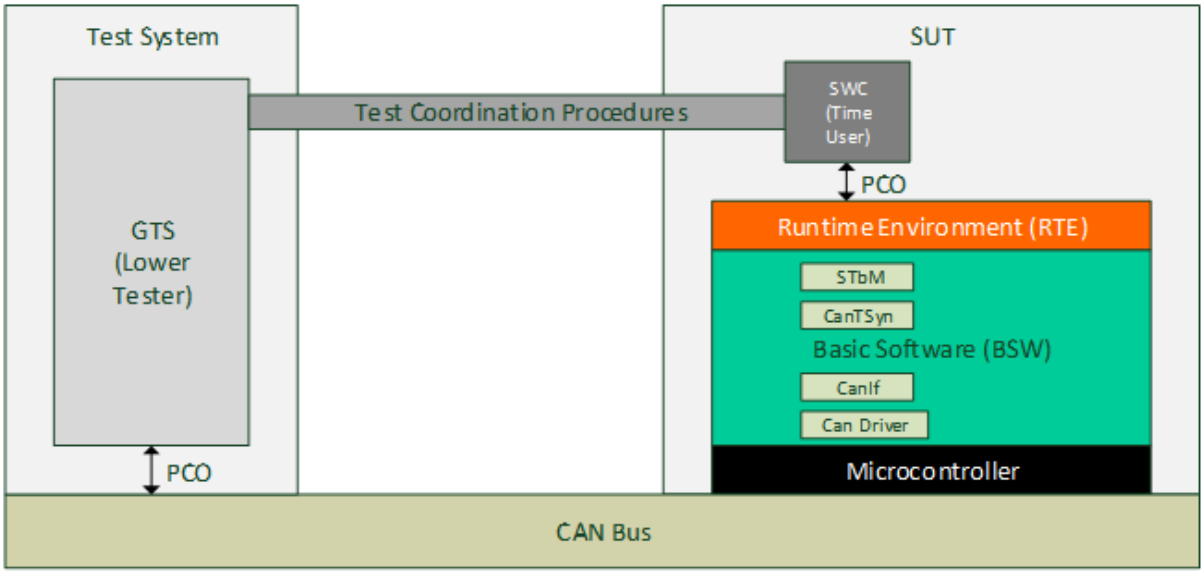


As shown in below figure, Functionalities of Global Time master of a time domain are tested over CAN in the use case 01.01

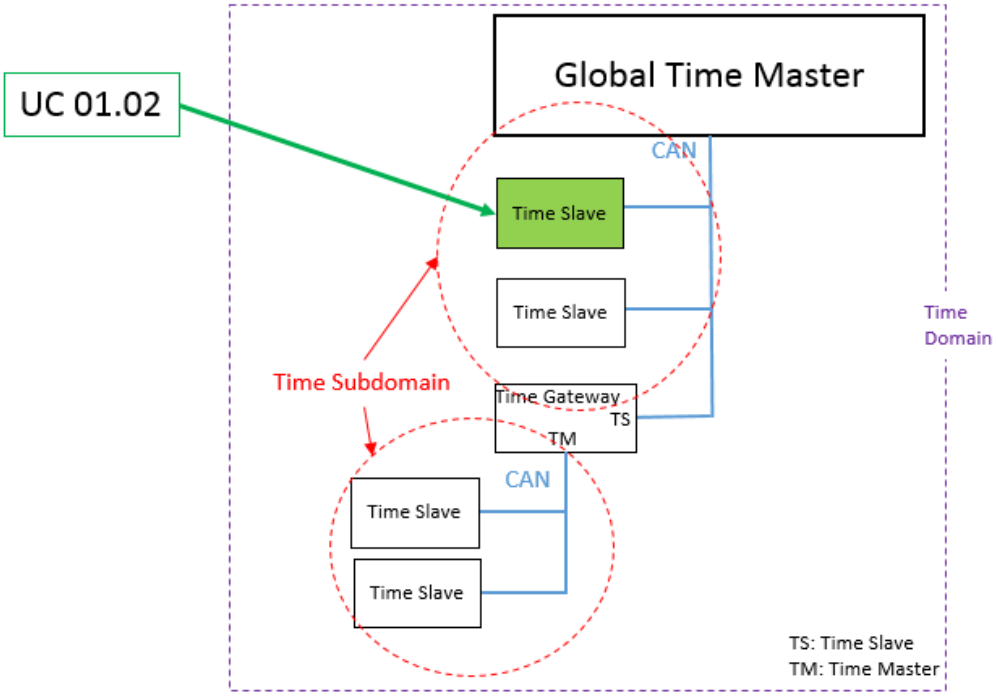


4.1.1.1.2 UC 01.02: Time Slave over CAN

SUT acts as Time Slave and Gets time base, offset time base and Synchronizes Local time to Global Time base.

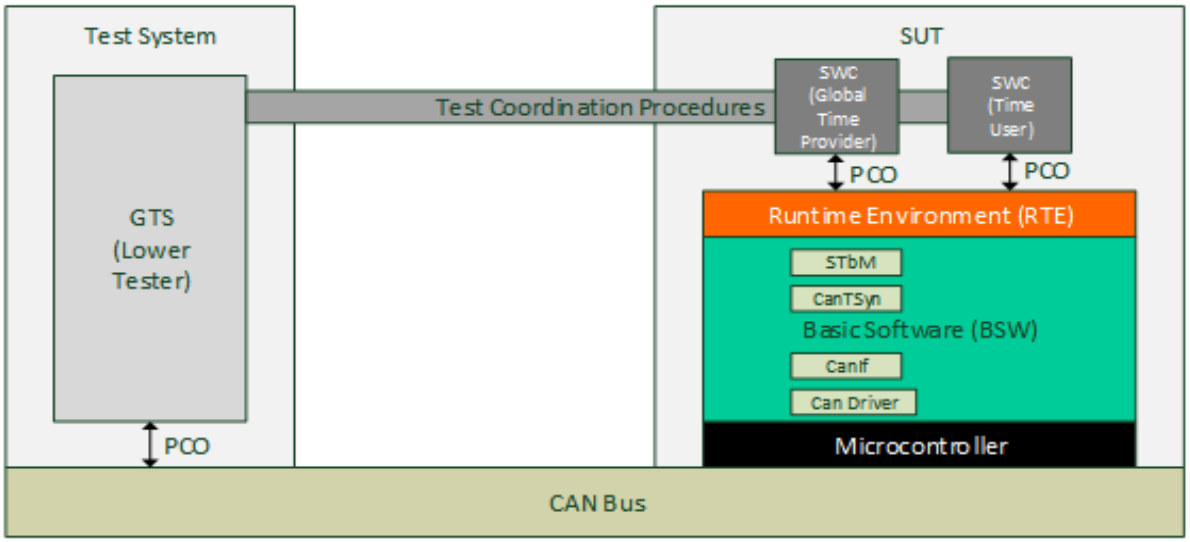


As shown in below figure, Functionalities of Time Slave of a time domain are tested over CAN in the use case 01.02



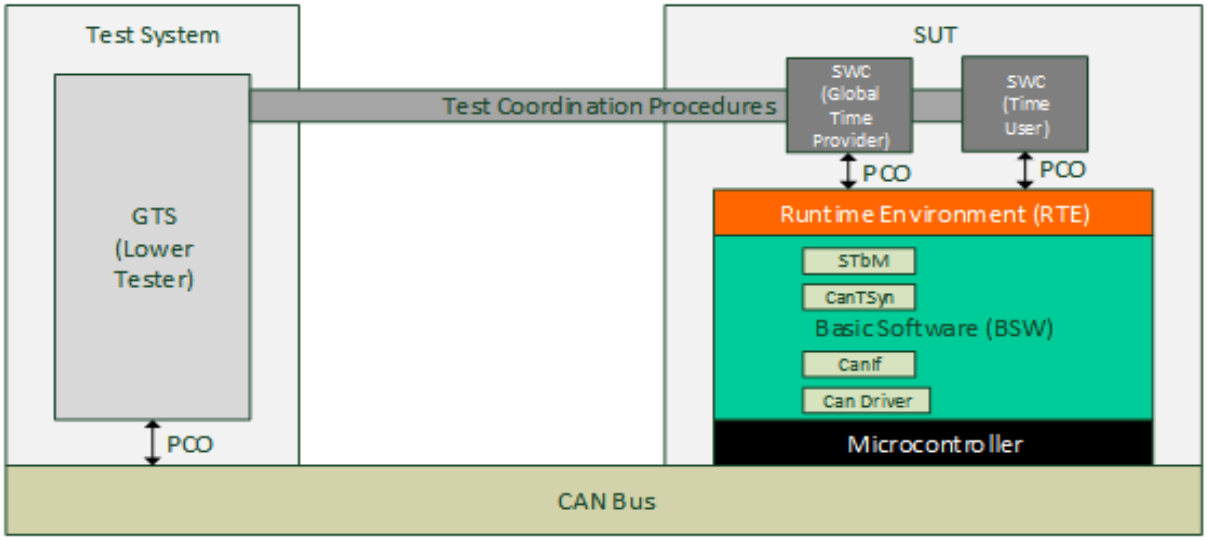
4.1.1.1.3 UC 01.03: Synchronization of Runnable entities to Global Time

SUT Synchronizes Runnable entities to Global Time base using OS Scheduler (Triggered Customer).



4.1.1.1.4 UC 01.04: Initialization of time base from value stored in Non-volatile Memory

During initialization, SUT updates time base from Non-volatile Memory.



4.1.1.2 Specific Requirements

Not Applicable.

4.1.1.3 Test Coordination Requirements

<u>UC 01.01: Global Time Master over CAN</u>

- Test System (LT <CAN>) shall read the CanTSyn CAN Frames and decode the same as per Frame Format provided in AUTOSAR_SWS_TimeSyncOverCAN.

UC 01.02: Time Slave over CAN

- Test System (LT <CAN>) shall encode the CanTSyn CAN Frames as per Frame Format provided in AUTOSAR_SWS_TimeSyncOverCAN and transmit over bus.

Requirements for CRC Calculation

- Test System (LT <CAN>) shall use the Crc_CalculateCRC8H2F() (Refer AUTOSAR Specification of CRC Routines AUTOSAR_SWS_CRCLibrary.pdf) to calculate the CRC of the Frame. Below are the parameters used for CRC calculation:
  The CRC start value shall be 0xFF.
  The CRC final XOR-value shall be 0xFF.
  The CRC polynomial shall be 0x2F.
  The DataIDList shall be same as provided in CanTSyn Static Configuration.

4.1.2 Test Case Design

Below diagrams explain test design for different use cases

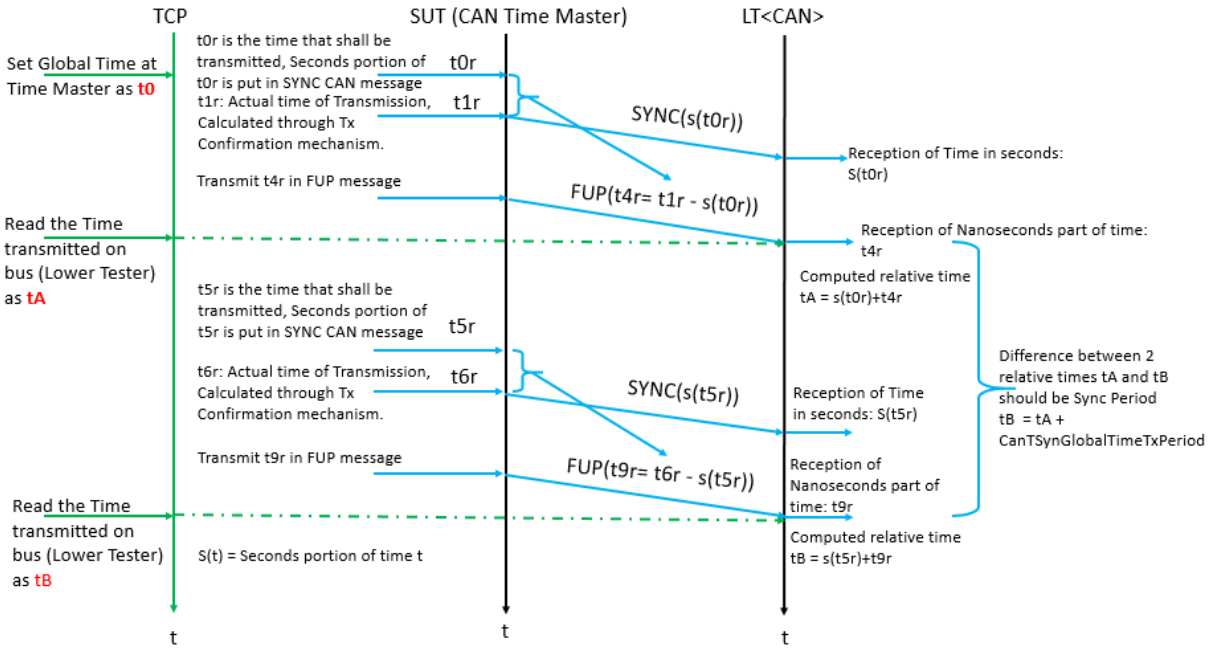


Fig: UC 01.01 - Global Time Master over CAN Test Design

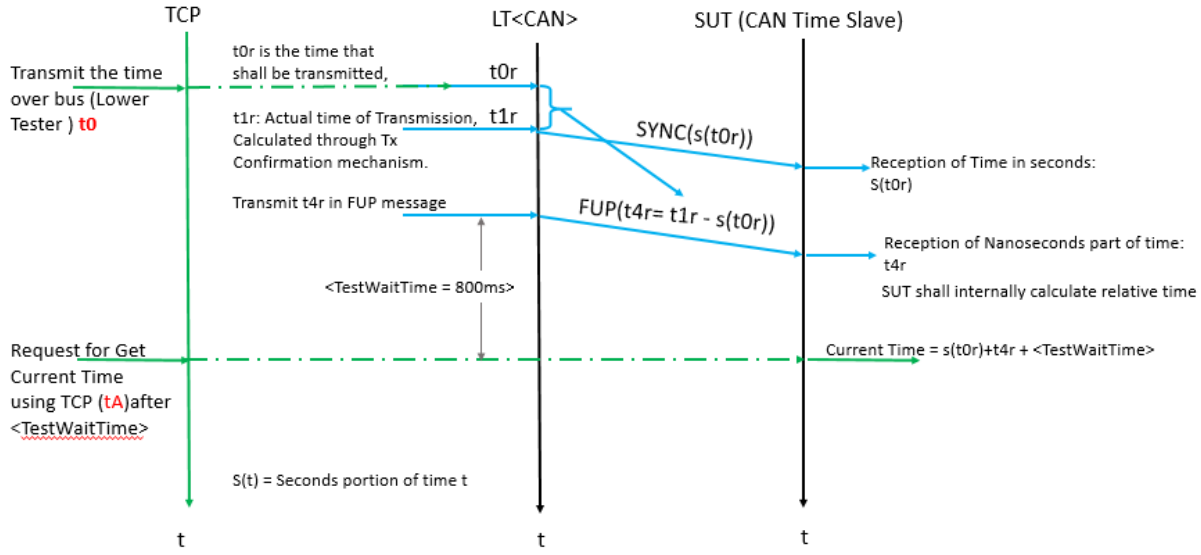


Fig: UC 01.02 - Time Slave over CAN Test Design

4.2 Configuration requirements

The configuration can be divided into two separate parts. The *test configuration* describes variables used to parameterize the test case. The *static configuration* describes the necessary settings of the DUT in order to allow a test case to perform.

4.2.1 Test Configuration

Communication data base for CanTSyn is depicted below

test configuration parameters			
I-Pdu	CAN ID	Tx ECU	Rx ECU
AT_101_Ipdu	101	SUT	Test Bench
AT_102_Ipdu	102	Test Bench	SUT

4.2.2 Static Configuration

4.2.2.1 Static Configuration Groups

SCG_ATS_GlobalTimeSync_Time Master	
System Description Parameters	
StbM	
SystemTemplate::GlobalTime::GlobalTimeMaster.isSystemWideGlobalTimeMaster	TRUE
CanTSyn	
SystemTemplate::GlobalTime::GlobalTimeMaster.syncPeriod	2000ms
SystemTemplate::GlobalTime::CAN::GlobalTimeCanMaster.syncConfirmationTimeout	80ms
SystemTemplate::GlobalTime::GlobalTimeDomain.globalTimePdu	Ref. To PDU
ECU Configuration Parameters	
StbM	
StbM.StbMGeneral.StbMMainFunctionPeriod	5ms

StbM.StbMSynchronizedTimeBase.StbMLocalTimeRef	Ref. to OSCounter
CanTSyn	
CanTSyn.CanTSynGeneral.CanTSynMainFunctionPeriod	5ms
CanTSyn.CanTSynGlobalTimeDomain. CanTSynGlobalTimeDomainId	1
CanTSyn.CanTSynGlobalTimeDomain. CanTSynSynchronizedTimeBaseRef	Ref. to StbM time base
CanTSyn.CanTSynGlobalTimeDomain. CanTSynGlobalTimeMaster. CanTSynGlobalTimeTxFollowUpOffset	100ms
CanTSyn.CanTSynGlobalTimeDomain. CanTSynGlobalTimeMaster.CanTSynGlobalTimeMasterPdu. CanTSynGlobalTimeMasterConfirmationHandleId	0
Use Cases	
UC 01.01	

SCG_ATS_GlobalTimeSync_Time Slave	
System Description Parameters	
StbM	
SystemTemplate::GlobalTime::GlobalTimeMaster.isSystemWideGlobalTimeMaster	FALSE
CanTSyn	
SystemTemplate::GlobalTime::GlobalTimeMaster.syncPeriod	2000ms
SystemTemplate::GlobalTime::GlobalTimeDomain.followUpTimeoutValue	300ms
ECU Configuration Parameters	
StbM	
StbM.StbMGeneral.StbMMainFunctionPeriod	5ms
StbM.StbMSynchronizedTimeBase.StbMLocalTimeRef	Ref. to OSCounter
CanTSyn	
CanTSyn.CanTSynGeneral.CanTSynMainFunctionPeriod	5ms
CanTSyn.CanTSynGlobalTimeDomain. CanTSynGlobalTimeDomainId	1
CanTSyn.CanTSynGlobalTimeDomain. CanTSynGlobalTimeSequenceCounterJumpWidth	1
CanTSyn.CanTSynGlobalTimeDomain. CanTSynSynchronizedTimeBaseRef	Ref. to StbM time base
CanTSyn.CanTSynGlobalTimeDomain. CanTSynGlobalTimeSlave.CanTSynGlobalTimeSlavePdu. CanTSynGlobalTimeSlaveConfirmationHandleId	1
CanTSyn.CanTSynGlobalTimeDomain. CanTSynGlobalTimeSlave.CanTSynGlobalTimeSlavePdu.	Ref. to PDU

CanTSynGlobalTimePduRef	
Use Cases	
UC 01.02	

SCG_ATS_GlobalTimeSync_Schedule table synchronization	
System Description Parameters	
StbM	
SystemTemplate::GlobalTime::GlobalTimeMaster.isSystemWideGlobalTimeMaster	TRUE
ECU Configuration Parameters	
StbM	
StbM.StbMGeneral.StbMMainFunctionPeriod	5ms
StbM.StbMTriggeredCustomer.StbMTriggeredCustomerPeriod	5ms
StbM.StbMSynchronizedTimeBase.StbMLocalTimeRef	Ref. to OSCounter
StbM.StbMTriggeredCustomer.StbMOSScheduleTableRef	Ref. to OS Schedule table
StbM.StbMTriggeredCustomer.StbMSynchronizedTimeBaseRef	Ref. to StbM time base
Test Cases	
UC 01.03	

SCG_ATS_GlobalTimeSync_NvM	
System Description Parameters	
StbM	
SystemTemplate::GlobalTime::GlobalTimeMaster.isSystemWideGlobalTimeMaster	TRUE
ECU Configuration Parameters	
StbM	
StbM.StbMGeneral.StbMMainFunctionPeriod	5ms
StbM.StbMSynchronizedTimeBase.StbMLocalTimeRef	Ref. to OSCounter
StbM.StbMSynchronizedTimeBase.StbMStoreTimebaseNonVolatile	STORAGE_AT_SHUTDOWN
Use Cases	
UC 01.04	

4.2.2.2 Required System Description

Refer section 3.2.2.1

4.2.2.3 Required ECU Configuration

Refer section 3.2.2.1

4.2.2.4 Required Software Components

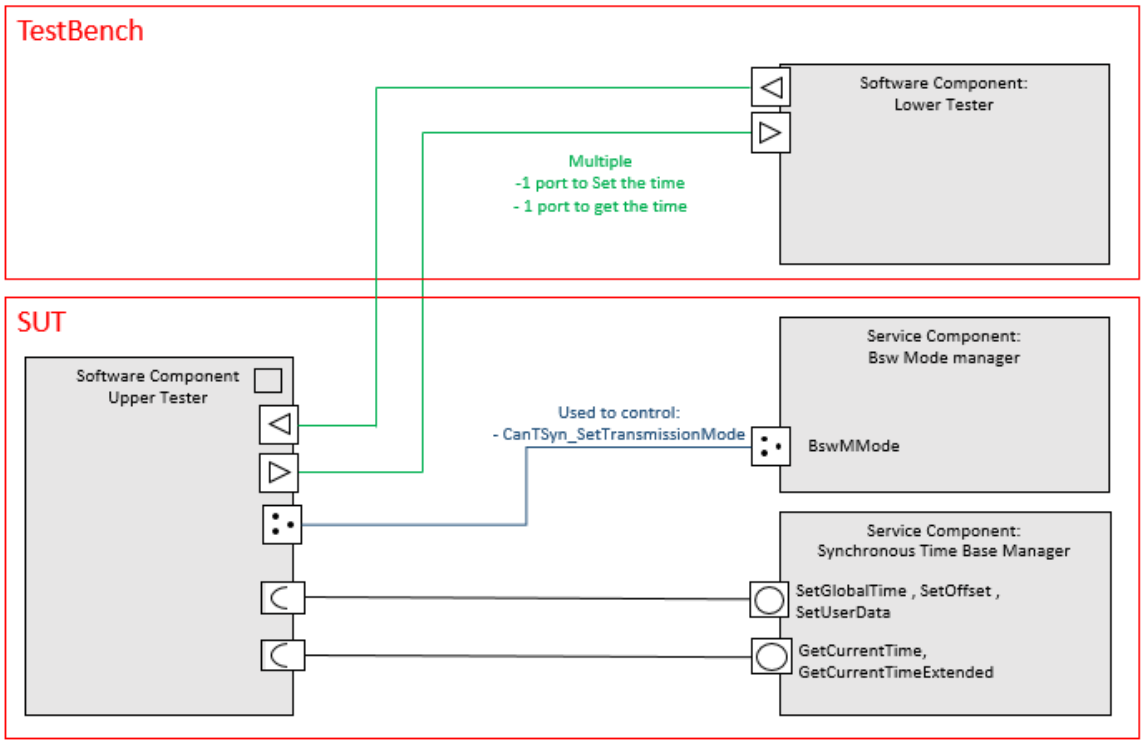


Fig: UC 01.01 - Global Time Master over CAN SWC Overview

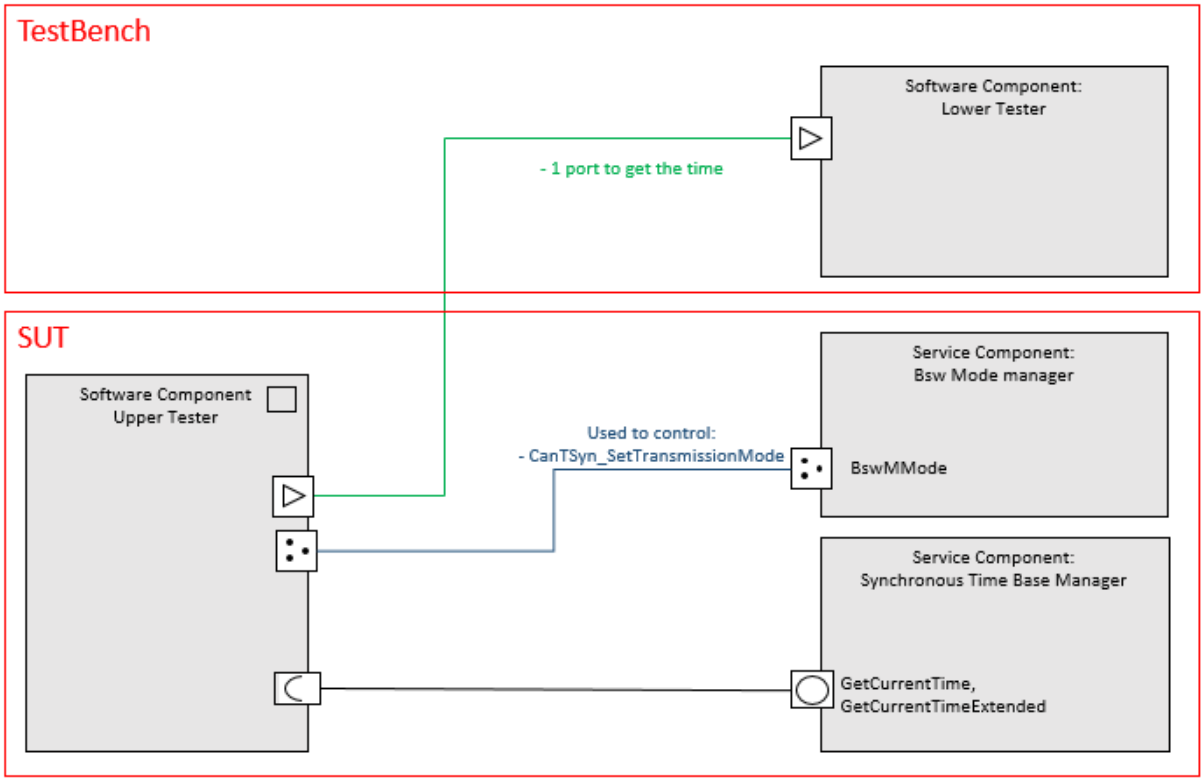


Fig: UC 01.02 - Time Slave over CAN SWC Overview

4.2.2.4.1 SWC Client GlobalTime_Provider

SWC Name	GlobalTime_Provider		
PORTS	**Name**	Client_SetGlobalTime	
	Type	RPortPrototype	
	Interface	GlobalTime_Master_Interface	
	Requirements		
	Name	Client_SetUserData	
	Type	RPortPrototype	
	Interface	GlobalTime_Master_Interface	
	Requirements		
	Name	Client_SetOffset	
	Type	RPortPrototype	
	Interface	GlobalTime_Master_Interface	
	Requirements		
RUNNABLE ENTITIES	**Name**	RUN_GlobalTimeProvider	
	Requirements	Runnable shall be invoked by TCP	
	ServerCallPoint	**Name**	sscp_GlobalTimeProvider
		Type	SynchronousServerCallPoint
		Access to	Client_SetGlobalTime (Write operation) Client_SetUserData (Write operation) Client_SetOffset (Write operation)
		Requirements	

4.2.2.4.2 SWC Client Time_User

SWC Name	Time_User	
	Name	Client_GetCurrentTime

PORTS	**Type**	RPortPrototype	
	Interface	GlobalTime_Slave_Interface	
	Requirements		
	Name	Client_GetCurrentTimeExtended	
	Type	RPortPrototype	
	Interface	GlobalTime_Slave_Interface	
	Requirements		
RUNNABLE ENTITIES	**Name**	RUN_TimeUser	
	Requirements	Runnable shall be invoked by TCP	
	ServerCallPoint	**Name**	sscp_TimeUser
		Type	SynchronousServerCallPoint
		Access to	Client_GetCurrentTime (Read operation) Client_GetCurrentTimeExtended (Read operation)
		Requirements	

4.2.2.4.3 SWC Server StbM

SWC Name	StbM	
	Name	Server_SetGlobalTime
	Type	PPortPrototype
	Interface	GlobalTime_Master_Interface
	Requirements	
	Name	Server_SetOffset
	Type	PPortPrototype
	Interface	GlobalTime_Master_Interface
	Requirements	

PORTS	**Name**	Server_SetUserData	
	Type	PPortPrototype	
	Interface	GlobalTime_Master_Interface	
	Requirements		
	Name	Server_GetCurrentTime	
	Type	PPortPrototype	
	Interface	GlobalTime_Slave_Interface	
	Requirements		
	Name	Server_GetCurrentTimeExtended	
	Type	PPortPrototype	
	Interface	GlobalTime_Slave_Interface	
	Requirements		
RUNNABLE ENTITIES	**Name**	StbM_SetGlobalTime	
	Requirements		
	Started by Event	**Name**	OIE_ SetGlobalTime
		Type	OperationInvokedEvent Port:Server_SetGlobalTime Operation: Read
		Requirements	
	Name	StbM_SetOffset	
	Requirements		
	ServerCallPoint	**Name**	OIE_SetOffset
		Type	OperationInvokedEvent Port: Server_SetOffset Operation: Read
		Requirements	
	Name	StbM_SetUserData	

	Requirements		
	Started by Event	**Name**	OIE_ SetUserData
		Type	OperationInvokedEvent Port: Server_SetUserData Operation: Read
		Requirements	
	Name	StbM_GetCurrentTime	
	Requirements		
	Started by Event	**Name**	OIE_ GetCurrentTime
		Type	OperationInvokedEvent Port: Server_GetCurrentTime Operation: Read
		Requirements	
	Name	StbM_GetCurrentTimeExtended	
	Requirements		
	Started by Event	**Name**	OIE_GetCurrentTimeExtended
		Type	OperationInvokedEvent Port: Server_GetCurrentTimeExtended Operation: Read
		Requirements	

4.3 Re-usable Test Steps

Not applicable

4.4 Test cases

4.4.1 [ATS_GTS_01228] Global Time Master: Setting of Global Time base by Active Customer and sending of SYNC frames (CRC_SUPPORTED) over CAN

Test Objective	Global Time Master: Setting of Global Time base by Active Customer and sending of SYNC frames (CRC_SUPPORTED) over CAN		
ID	ATS_GTS_01228	**AUTOSAR**	4.2.1|4.2.2

		Releases	
Affected Modules	StbM, CanTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00131 ATR: ATR_ATR_00132 ATR: ATR_ATR_00133		
Trace to SWS Item	SynchronizedTimeBaseManager: SWS_StbM_00240 TimeSyncOverCAN: SWS_CanTSyn_00011 TimeSyncOverCAN: SWS_CanTSyn_00028 TimeSyncOverCAN: SWS_CanTSyn_00031		
Requirements / Reference to Test Environment	Use Case UC01.01		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 1 CanTSyn: CanTSynGlobalTimeTxCrcSecured= CRC_SUPPORTED CanTSynGlobalTimeSyncDataIDListIndex = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} CanTSynGlobalTimeSyncDataIDListValue in ASCII = {"A", "U", "T", "O", "S", "A", "R", "A", "T", "S", "G", "T", "S", "S", "Y", "N"} CanTSynGlobalTimeFupDataIDListIndex = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} CanTSynGlobalTimeFupDataIDListValue in ASCII = {"A", "U", "T", "O", "S", "A", "R", "A", "T", "S", "G", "T", "S", "F", "U", "P"}		
Summary	Aim is to: Verify that StbM accepts the global time base from Upper Tester using client-server Interface. Verify that CanTSyn shall Transmit the global time base to time slave periodically via SYNC and FUP message with CRC secured.		
Needed Adaptation to other Releases	Not Applicable		
Pre-conditions	SUT shall be initialized.		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** Start RUN_ GlobalTimeProvider		
Step 2	**[RUN<RUN_ GlobalTimeProvider>]** Execute Rte_Call_ Client_SetGlobalTime and Rte_Call_ Client_SetUserData with below values: timeBaseId = 1 StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds =		**[RUN<RUN_ GlobalTimeProvider>]** Rte_Call returns RTE_E_OK

	0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA	
Step 3	**[CP]** Wait 100ms	
Step 4	**[CP]** Start RUN_TimeUser	
Step 5	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time received from Time user should be as mentioned below: StbM_TimeStampType.nanoseconds = 0x00000000 + <TestWaitTime = 100ms>. StbM_TimeStampType.seconds = 0x00000E10(3600d) StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA
Step 6	**[LT<CAN>]** Receives the SYNC message with CRC validation as per test co-ordination requirement for CRC	**[LT]** Receives SYNC message in the format mentioned below: Byte 0: Type = 0x20 Byte 1: CRC Byte 2: D = 0x1 SC = 0x0 Byte 3: User Byte 0 = 0xAA Byte 4-7: SyncTimeSec = StbM_TimeStampType.seconds
Step 7	**[LT<CAN>]** Receives the FUP message with CRC validation as per test co-ordination requirement for CRC	**[LT]** Receives FUP message in the format mentioned below:

		Byte 0: Type = 0x28 Byte 1: CRC Byte 2: D = 0x1 SC = 0x0 Byte 3: reserved (Bit 7 to Bit 3) = 0 SGW (Bit 2) = 0 OVS = Overflow of seconds (Bit 1 to Bit 0) Byte 4-7: SyncTimeNSec = StbM_TimeStampType.nanoseconds
Step 8	**[LT<CAN>]** Receives Global time base. Store the time as base for next periodic message processing	**[LT]** Get the time stamp values as below: StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 9	**[LT<CAN>]** Receives the next periodic SYNC and FUP message with CRC validation as per test co-ordination requirement for CRC and calculate the CRC value	**[LT]** CRC in byte 1 of the frame shall match with Calculated CRC value. Time stamp values shall be as mentioned below: StbM_TimeStampType.nanoseconds = tB.Nano StbM_TimeStampType.seconds = tB.Sec StbM_TimeStampType.secondsHi = tB.SecHi StbM_UserDataType.User Data Byte 0 = 0xAA The difference between two time base shall be tB – tA = CanTSynGlobalTimeTxPeriod (2 second) + CanTSynGlobalTimeTxFollowUpOffset.

Post-conditions	None

4.4.2 [ATS_GTS_01266] Global Time Master: Handling of SYNC message Confirmation Failures

Test Objective	Global Time Master: Handling of SYNC message Confirmation Failures		
ID	ATS_GTS_01266	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	CanTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00133		
Trace to SWS Item	TimeSyncOverCAN: SWS_CanTSyn_00033		
Requirements / Reference to Test Environment	Use Case UC01.01		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 1 CanTSyn: CanTSynGlobalTimeTxCrcSecured= CRC_NOT_SUPPORTED CanIf: CanIf.CanIfInitCfg.CanIfTxPduCfg.CanIfTxPduUserTxConfirmationUL = CDD.		
Summary	Aim is to: Verify that CanTSyn shall send the SYNC message and on confirmation timeout 'CanTSynMasterConfirmationTimeout', transmission request shall be revoked and no FUP message shall be sent.		
Needed Adaptation to other Releases	Not Applicable		
Pre-conditions	SUT shall be initialized. StbM shall be initialized with base time: StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x00000000 StbM_TimeStampType.secondsHi = 0x0000		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[SUT]** CanTSyn transmit SYNC message		**[LT<CAN>]** Receive SYNC message with format Byte 0: Type = 0x10 Byte 1: User Byte 1 = 0x00

		Byte 2: D = 0x1 SC = 0x0 Byte 3: User Byte 0 = 0x00 Byte 4-7: SyncTimeSec = StbM_TimeStampType.seconds
Step 2	**[LT<CAN>]** Waits for FUP message	**[LT<CAN>]** No FUP message received. (Since parameter CanIfTxPduUserTxConfirmationUL for CanTSyn PDU = CDD, confirmation shall not reach StbM.This causes transmit confirmation timeout at CanTSyn)
Step 3	**[SUT]** CanTSyn transmit next periodic SYNC message	**[LT<CAN>]** Receive SYNC message with format Byte 0: Type = 0x10 Byte 1: User Byte 1 = 0x00 Byte 2: D = 0x1 SC = 0x0 Byte 3: User Byte 0 = 0x00 Byte 4-7: SyncTimeSec = StbM_TimeStampType.seconds + CanTSynGlobalTimeTxPeriod (2 second) + CanTSynMasterConfirmationTimeout.
Step 4	**[LT<CAN>]** Waits for FUP message	**[LT<CAN>]** No FUP message received. (Since parameter CanIfTxPduUserTxConfirmationUL for CanTSyn PDU = CDD, confirmation shall not reach StbM.This causes transmit confirmation timeout at CanTSyn)
Post-conditions	None	

4.4.3 [ATS_GTS_01264] Global Time Master: Setting of Global Time base by Active Customer and sending of offset frames (CRC_NOT_SUPPORTED) over CAN

Test Objective	Global Time Master: Setting of Global Time base by Active Customer and sending of offset frames (CRC_NOT_SUPPORTED) over CAN		
ID	ATS_GTS_01264	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, CanTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00131 ATR: ATR_ATR_00132 ATR: ATR_ATR_00133		
Trace to SWS Item	SynchronizedTimeBaseManager: SWS_StbM_00240 TimeSyncOverCAN: SWS_CanTSyn_00038 TimeSyncOverCAN: SWS_CanTSyn_00040 TimeSyncOverCAN: SWS_CanTSyn_00041		
Requirements / Reference to Test Environment	Use Case UC01.01		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 16. CanTSyn: CanTSynGlobalTimeTxCrcSecured= CRC_NOT_SUPPORTED		
Summary	Aim is to: Verify that StbM accepts the global time base from Upper Tester using client-server Interface. Verify that CanTSyn shall Transmit the offset time base to time slave periodically via OFS and OFNS message.		
Needed Adaptation to other Releases	Not Applicable		
Pre-conditions	SUT shall be initialized.		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** Start RUN_ GlobalTimeProvider		
Step 2	**[RUN<RUN_ GlobalTimeProvider>]** Execute Rte_Call_ Client_ SetGlobalTime with below values: timeBaseId = 1 StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds =		**[RUN<RUN_ GlobalTimeProvider>]** Rte_Call returns RTE_E_OK

	0x00000E10 StbM_TimeStampType.secondsHi = 0x0000	
Step 3	**[CP]** Wait 100ms	
Step 4	**[RUN<RUN_ GlobalTimeProvider>]** Execute Rte_Call_ Client_ SetOffset with below values: timeBaseId = 16 StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x00000064 StbM_TimeStampType.secondsHi = 0x0000	**[RUN<RUN_ GlobalTimeProvider>]** Rte_Call returns RTE_E_OK
Step 5	**[CP]** Wait 100ms	
Step 6	**[CP]** Start RUN_TimeUser	
Step 7	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime Time base = 16	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time received from Time user shall be as mentioned below: StbM_TimeStampType.nanoseconds = 0x00000000 + <TestWaitTime = 100ms> + <TestWaitTime = 100ms> StbM_TimeStampType.seconds = 0x00000E10 + 0x00000064 StbM_TimeStampType.secondsHi = 0x0000
Step 8	**[LT<CAN>]** Receives the OFS message	**[LT]** Receives OFS message with format Byte 0: Type = 0x30 Byte 1: reserved = 0x00 Byte 2: D = 0x0 SC = 0x0

		Byte 3: OfsTimeSecLsbHi = <StbM_TimeStampType.secondsHi (LSB) = 0x00> Byte 4-7: OfsTimeSecLsbLo = <StbM_TimeStampType.seconds = 0x00000064>
Step 9	**[LT<CAN>]** Receives the OFNS message	**[LT]** Receives OFNS message with format Byte 0: Type = 0x38 Byte 1: reserved = 0x00 Byte 2: D = 0x0 SC = 0x0 Byte 3: OfsTimeSecMsbHi = <StbM_TimeStampType.secondsHi (MSB) = 0x00> Byte 4-7: OfsTimeNSec = <StbM_TimeStampType.nanoseconds = 0x00000000>
Post-conditions	None	

4.4.4 [ATS_GTS_01268] Global Time Master: Handling of Offset message Confirmation Failures

Test Objective	Global Time Master: Handling of Offset message Confirmation Failures		
ID	ATS_GTS_01268	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	CanTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00133		
Trace to SWS Item	TimeSyncOverCAN: SWS_CanTSyn_00042		
Requirements / Reference to Test Environment	Use Case UC01.01		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 16 CanTSyn: CanTSynGlobalTimeTxCrcSecured= CRC_NOT_SUPPORTED		

	CanIf: CanIf.CanIfInitCfg.CanIfTxPduCfg.CanIfTxPduUserTxConfirmationUL = CDD.	
Summary	Aim is to: Verify that CanTSyn shall send the OFS message and on confirmation timeout 'CanTSynMasterConfirmationTimeout', transmission request shall be revoked and no OFNS message shall be sent.	
Needed Adaptation to other Releases	Not Applicable	
Pre-conditions	SUT shall be initialized. StbM shall be initialized with base time: StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x00000000 StbM_TimeStampType.secondsHi = 0x0000	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[SUT]** CanTSyn transmit OFS message with TimeBaseId = 16 OfsTimeSecLsbLo = 0x00000002	**[LT<CAN>]** Receive OFS message with format Byte 0: Type = 0x30 Byte 1: reserved = 0x00 Byte 2: D = *0x0* SC = 0x0 Byte 3: OfsTimeSecLsbHi = 8 Bit offset time stamp (LSB) from secondsHi Byte 4-7: OfsTimeSecLsbLo = StbM_TimeStampType.seconds
Step 2	**[LT<CAN>]** Waits for OFNS message	**[LT<CAN>]** No OFNS message received. (Since parameter CanIfTxPduUserTxConfirmationUL for CanTSyn PDU = CDD, Confirmation shall not reach StbM. This cause transmit confirmation timeout at CanTSyn)
Step 3	**[SUT]** CanTSyn transmit next periodic OFS message	**[LT<CAN>]** Receive OFS message with format Byte 0: Type = 0x30 Byte 1: reserved = 0x00 Byte 2: D = 0x0

		SC = 0x1 Byte 3: OfsTimeSecLsbHi = 8 Bit offset time stamp (LSB) from secondsHi Byte 4-7: OfsTimeSecLsbLo = StbM_TimeStampType.seconds + CanTSynGlobalTimeTxPeriod (2 second) + CanTSynMasterConfirmationTimeout.
Step 4	**[LT<CAN>]** Waits for OFNS message	**[LT<CAN>]** No OFNS message received. (Since parameter CanIfTxPduUserTxConfirmationUL for CanTSyn PDU = CDD, Confirmation shall not reach StbM. This cause transmit confirmation timeout at CanTSyn)
Post-conditions	None	

4.4.5 [ATS_GTS_01238] Global Time Master: Handling of time base using NvM (Storage and Retrieve)

Test Objective	Global Time Master: Handling of time base using NvM (Storage and Retrieve)		
ID	ATS_GTS_01238	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00131 ATR: ATR_ATR_00132		
Trace to SWS Item	SynchronizedTimeBaseManager: SWS_StbM_00171 SynchronizedTimeBaseManager: SWS_StbM_00172		
Requirements / Reference to Test Environment	Use Case UC01.04		
Configuration Parameters	StbM: StbMMainFunctionPeriod 5ms StbMIsSystemWideGlobalTimeMaster = TRUE StbMStoreTimebaseNonVolatile = STORAGE_AT_SHUTDOWN		
Summary	Aim is to Verify that during Initialization StbM shall load the Time base value from NvM and store the Time base to NvM at shutdown.		
Needed Adaptation to other Releases	Not Applicable		
Pre-conditions	SUT shall be initialized.		
Main Test Execution			
Test Steps		**Pass Criteria**	

Step 1	**[CP]** Start RUN_ GlobalTimeProvider	
Step 2	**[RUN<RUN_ GlobalTimeProvider>]** Execute Rte_Call_ Client_SetGlobalTime and Rte_Call_ Client_SetUserData with below values: timeBaseId = 1 StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = xAA	**[RUN<RUN_ GlobalTimeProvider>]** Rte_Call returns RTE_E_OK
Step 3	**[CP]** Wait 100ms	
Step 4	**[CP]** Start RUN_TimeUser	
Step 5	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time received from Time user shall be as mentioned below: StbM_TimeStampType.nanoseconds = 0x00000000 + <TestWaitTime = 100ms> StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA
Step 6	**[SUT]** ECU shutdown	**[SUT]** Time base shall be stored into NvM during shutdown
Step 7	**[SUT]** Restart the ECU	**[SUT]** Time base shall be retrieved from NvM during initialization.

Step 8	**[CP]** Wait 100ms	
Step 9	**[CP]** Start RUN_TimeUser	
Step 10	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = 0x00000000 + <TestWaitTime = 100ms> + <TestWaitTime = 100ms> StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA
Post-conditions	None	

4.4.6 [ATS_GTS_01242] Time Slave: Reception of SYNC frames (CRC_IGNORED) over CAN, Synchronize Local Time Base and share the current time to active customers

Test Objective	Time Slave: Reception of SYNC frames (CRC_IGNORED) over CAN, Synchronize Local Time Base and share the current time to active customers		
ID	ATS_GTS_01242	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, CanTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00131 ATR: ATR_ATR_00132 ATR: ATR_ATR_00133		
Trace to SWS Item	SynchronizedTimeBaseManager: SWS_StbM_00247 TimeSyncOverCAN: SWS_CanTSyn_00011		
Requirements / Reference to Test Environment	Use Case UC01.02		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 1 CanTSyn: CanTSynRxCrcValidated = CRC_IGNORED		

Summary	Verify that CanTSyn shall call StbM to update global time base on reception of SYNC and FUP message even with invalid CRC when CanTSynRxCrcValidated = CRC_IGNORED.	
Needed Adaptation to other Releases	Not Applicable	
Pre-conditions	SUT shall be initialized.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[LT<CAN>]** Transmit SYNC message with timeBaseId = 1 StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB	**[SUT]** Receives SYNC message ignoring CRC value (if there) in below format Byte 0: Type = 0x10 Byte 1: User Byte 1 = 0xBB Byte 2: D = 0x1 SC = 0x0 Byte 3: User Byte 0 = 0xAA Byte 4-7: SyncTimeSec = StbM_TimeStampType.seconds
Step 2	**[LT<CAN>]** Transmit FUP message with StbM_TimeStampType.nanoseconds = t4r	**[SUT]** Receives FUP message ignoring CRC value (if there) in below format Byte 0: Type = 0x18 Byte 1: User Byte 2 = 0x00 Byte 2: D = 0x1 SC = 0x0 Byte 3: reserved (Bit 7 to Bit 3) = 0 SGW (Bit 2) = 0 OVS = Overflow of seconds (Bit 1 to Bit 0) Byte 4-7: SyncTimeNSec = StbM_TimeStampType.nanoseconds
Step 3		**[SUT]** StbM updates its time base with values StbM_TimeStampType.nanoseconds

		= tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi Time updated in StbM, tA = tA + ToleranceTime_CanTSyn. ToleranceTime_CanTSyn = Max one main function of CanTSyn as reception is asynchronous.
Step 4	**[CP]** Wait 800ms	
Step 5	**[CP]** Start RUN_TimeUser	
Step 6	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 800ms> StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Post-conditions	None	

4.4.7 [ATS_GTS_01271] Time Slave: Handling of SYNC message reception timeout (CanTSynGlobalTimeFollowUpTimeout)

Test Objective	Time Slave: Handling of SYNC message reception timeout (CanTSynGlobalTimeFollowUpTimeout)		
ID	ATS_GTS_01271	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, CanTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00131 ATR: ATR_ATR_00132 ATR: ATR_ATR_00133		
Trace to SWS Item	SynchronizedTimeBaseManager: SWS_StbM_00183 SynchronizedTimeBaseManager: SWS_StbM_00247		

	TimeSyncOverCAN: SWS_CanTSyn_00063	
Requirements / Reference to Test Environment	Use Case UC01.02	
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 1 CanTSyn: CanTSynRxCrcValidated = CRC_IGNORED	
Summary	To verify that if FUP message are not recieved within 'CanTSynGlobalTimeFollowUpTimeout' CanTSyn shall reset the sequence and wait for new SYNC message. To verify StbM shall set the bit TIMEOUT in Sync state (when UT requests for current time) if StbM_BusSetGlobalTime is not invoked within StbMSyncLossTimeout and Clear the bit on invocation of StbM_BusSetGlobalTime.	
Needed Adaptation to other Releases	Not Applicable	
Pre-conditions	SUT shall be initialized.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[LT<CAN>]** Transmit SYNC message with TimeBaseId = 1 StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB	**[SUT]** Receives SYNC message with format Byte 0: Type = 0x10 Byte 1: User Byte 1 = 0xBB Byte 2: D = 0x1 SC = 0x0 Byte 3: User Byte 0 = 0xAA Byte 4-7: SyncTimeSec = StbM_TimeStampType.seconds
Step 2	**[LT<CAN>]** Transmit FUP message with StbM_TimeStampType.nanoseconds = t4r	**[SUT]** Receives FUP message with format Byte 0: Type = 0x18 Byte 1: Reserved Byte 2: D = 0x1 SC = 0x0 Byte 3: reserved (Bit 7 to Bit 3) = 0

		SGW (Bit 2) = 0 OVS = Overflow of seconds (Bit 1 to Bit 0) Byte 4-7: SyncTimeNSec = StbM_TimeStampType.nanoseconds
Step 3		**[SUT]** CanTSyn shall update the time base of StbM StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi Time updated in StbM, tA = tA + ToleranceTime_CanTSyn. ToleranceTime_CanTSyn = Max one main function of CanTSyn as reception is asynchronous.
Step 4	**[CP]** Wait 800ms	
Step 5	**[CP]** Start RUN_TimeUser	
Step 6	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 800ms> StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 7	**[LT<CAN>]** Transmit SYNC message with TimeBaseId = 1 StbM_TimeStampType.seconds =	**[SUT]** Receives SYNC message with format Byte 0: Type = 0x10 Byte 1: User Byte 1 = 0xDD

	0x00001C20 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xCC StbM_UserDataType.User Data Byte 1 = 0xDD	Byte 2: D = 0x1 SC = 0x1 Byte 3: User Byte 0 = 0xCC Byte 4-7: SyncTimeSec = StbM_TimeStampType.seconds
Step 8	**[LT<CAN>]** Do not transmit FUP message. Reception timeout occurs after 300ms (CanTSynGlobalTimeFollowUpTimeout = 300ms) the sequence is reset and waits for a new SYNC message.	**[SUT]** Do not receives FUP message StbM_TimeBaseStatusType.TIMEOUT is set to 1. CanTSyn shall not update the time base of StbM
Step 9	**[CP]** Wait 800ms	
Step 10	**[CP]** Start RUN_TimeUser	
Step 11	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 800ms> StbM_TimeStampType.seconds = tA .Sec + CanTSynGlobalTimeTxPeriod (2 second) StbM_TimeStampType.secondsHi = tA .SecHi
Post-conditions	None	

4.4.8 [ATS_GTS_01273] Time Slave: Handling of SYNC message Sequence Mismatch failures

Test Objective	Time Slave: Handling of SYNC message Sequence Mismatch failures		
ID	ATS_GTS_01273	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, CanTSyn	**State**	reviewed

Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00131 ATR: ATR_ATR_00132 ATR: ATR_ATR_00133
Trace to SWS Item	SynchronizedTimeBaseManager: SWS_StbM_00247 TimeSyncOverCAN: SWS_CanTSyn_00076 TimeSyncOverCAN: SWS_CanTSyn_00078
Requirements / Reference to Test Environment	Use Case UC01.02
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 1 CanTSyn: CanTSynRxCrcValidated = CRC_IGNORED
Summary	To verify that if Sequence Counter of SYNC and FUP message are not matching, CanTSyn shall discard the received SYNC message and shall ignore the received FUP message. To verify Sequence Counter Jump Width between two SYNC messages greater than CanTSynGlobalTimeSequenceCounterJumpWidth, the messages will be ignored.
Needed Adaptation to other Releases	Not Applicable
Pre-conditions	SUT shall be initialized.

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[LT<CAN>]** Transmit SYNC message with TimeBaseId = 1 StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB	**[SUT]** Receives SYNC message with format Byte 0: Type = 0x10 Byte 1: User Byte 1 = 0xBB Byte 2: D = 0x1 SC = 0x0 Byte 3: User Byte 0 = 0xAA Byte 4-7: SyncTimeSec = StbM_TimeStampType.seconds
Step 2	**[LT<CAN>]** Transmit FUP message with StbM_TimeStampType.nanosecon	**[SUT]** Receives FUP message with format

	ds = t4r	Byte 0: Type = 0x18 Byte 1: Reserved Byte 2: D = 0x1 SC = 0x0 Byte 3: reserved (Bit 7 to Bit 3) = 0 SGW (Bit 2) = 0 OVS = Overflow of seconds (Bit 1 to Bit 0) Byte 4-7: SyncTimeNSec = StbM_TimeStampType.nanoseconds
Step 3		**[SUT]** CanTSyn shall update the time base of StbM StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi Time updated in StbM, tA = tA + ToleranceTime_CanTSyn. ToleranceTime_CanTSyn = Max one main function of CanTSyn as reception is asynchronous.
Step 4	**[CP]** Wait 800ms	
Step 5	**[CP]** Start RUN_TimeUser	
Step 6	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 800ms>. StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi StbM_TimeBaseStatusType.TIMEOUT is set to 0.
Step 7	**[LT<CAN>]** Transmit next periodic SYNC	**[SUT]**

	message with TimeBaseId = 1 StbM_TimeStampType.seconds = 0x00001C20 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xCC StbM_UserDataType.User Data Byte 1 = 0xDD	Receives SYNC message with format Byte 0: Type = 0x10 Byte 1: User Byte 1 = 0xDD Byte 2: D = 0x1 SC = 0x1 Byte 3: User Byte 0 = 0xCC Byte 4-7: SyncTimeSec = StbM_TimeStampType.seconds
Step 8	**[LT<CAN>]** Transmit FUP message with: StbM_TimeStampType.nanoseconds = t9r Different sequence number	**[SUT]** Receives FUP message with format Byte 0: Type = 0x18 Byte 1: Reserved Byte 2: D = 0x1 SC = 0x2 Byte 3: reserved (Bit 7 to Bit 3) = 0 SGW (Bit 2) = 0 OVS = Overflow of seconds (Bit 1 to Bit 0) Byte 4-7: SyncTimeNSec = StbM_TimeStampType.nanoseconds CanTSyn shall not update the time base of StbM
Step 9	**[CP]** Wait 800ms	
Step 10	**[CP]** Start RUN_TimeUser	
Step 11	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 800ms>. StbM_TimeStampType.seconds = tA .Sec +

		CanTSynGlobalTimeTxPeriod (2 second) StbM_TimeStampType.secondsHi = tA .SecHi StbM_TimeBaseStatusType.TIMEOUT is set to 1.
Step 12	**[LT<CAN>]** Transmit next periodic SYNC message with timeBaseId = 1 StbM_TimeStampType.seconds = 0x00002328 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xEE StbM_UserDataType.User Data Byte 1 = 0xFF Sequence Counter = 3	**[SUT]** Time base is not updated as per incoming time base as sequence counter value is greater than CanTSynGlobalTimeSequenceCounterJumpWidth.
Step 13	**[CP]** Wait 800ms	
Step 14	**[CP]** Start RUN_TimeUser	
Step 15	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 800ms> + <TestWaitTime = 800ms>. StbM_TimeStampType.seconds = tA .Sec + CanTSynGlobalTimeTxPeriod (2 second) + CanTSynGlobalTimeTxPeriod (2 second). StbM_TimeStampType.secondsHi = tA .SecHi StbM_TimeBaseStatusType.TIMEOUT is set to 1.
Post-conditions	None	

4.4.9 [ATS_GTS_01286] Time Slave: Reception of Offset frames (CRC_VALIDATED) over CAN, Synchronize Local Time Base and share the current time to active customers.

Test Objective	Time Slave: Reception of Offset frames (CRC_VALIDATED) over CAN, Synchronize Local Time Base and share the current time to active customers.		
ID	ATS_GTS_01286	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, CanTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00131 ATR: ATR_ATR_00132 ATR: ATR_ATR_00133		
Trace to SWS Item	SynchronizedTimeBaseManager: SWS_StbM_00247 TimeSyncOverCAN: SWS_CanTSyn_00080		
Requirements / Reference to Test Environment	Use Case UC01.02		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 17. CanTSyn: CanTSynRxCrcValidated = CRC_VALIDATED CanTSynGlobalTimeOfsDataIDListIndex = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} CanTSynGlobalTimeOfsDataIDListValue in ASCII = {"A", "U", "T", "O", "S", "A", "R", "A", "T", "S", "G", "T", "S", "O", "F", "S"} CanTSynGlobalTimeOfnsDataIDListIndex = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} CanTSynGlobalTimeOfnsDataIDListValue in ASCII = {"A", "U", "T", "O", "S", "A", "R", "A", "T", "S", "G", "T", "O", "F", "N", "S"}		
Summary	Aim is to: Verify that CanTSyn shall Receive the offset time periodically via OFS and OFNS message respectively with CRC validation. Verify that StbM shall synchronize its local offset time base on reception of Time Base from CanTSyn Verify that UT shall get the valid current time, offset time, current time in extended format, user data using Client-Server Interface.		
Needed Adaptation to other Releases	Not Applicable		
Pre-conditions	SUT shall be initialized.		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[LT<CAN>]** Transmit SYNC and FUP message with: TimeBaseId = 1 StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds =		**[SUT]** Get the time stamp values with CRC validation as below: StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA

	0x00000E10 (3600d) StbM_TimeStampType.secondsHi = 0x0000	.Sec StbM_TimeStampType.secondsHi = tA .SecHi Time updated in StbM, tA = tA + ToleranceTime_CanTSyn. ToleranceTime_CanTSyn = Max one main function of CanTSyn as reception is asynchronous.
Step 2	**[CP]** Wait 800ms	
Step 3	**[LT<CAN>]** Transmit OFS and OFNS message with: TimeBaseId = 17 StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x00000010 (16d) StbM_TimeStampType.secondsHi = 0x0000	**[SUT]** Get the offset time stamp values with CRC validation as below: StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 4	**[CP]** Wait 800ms	
Step 5	**[CP]** Start RUN_TimeUser	
Step 6	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 800ms> + <TestWaitTime = 800ms> StbM_TimeStampType.seconds = tA .Sec + <offset time = 0x00000010> StbM_TimeStampType.secondsHi = tA .SecHi
Post-conditions	None	

4.4.10 [ATS_GTS_01244] Time Slave: Reception of Offset frames (CRC_NOT_VALIDATED) over CAN, Synchronize Local Time Base and share the current time to active customer

Test Objective	Time Slave: Reception of Offset frames (CRC_NOT_VALIDATED) over CAN, Synchronize Local Time Base and share the current time to active customer		
ID	ATS_GTS_01244	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, CanTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00131 ATR: ATR_ATR_00132 ATR: ATR_ATR_00133		
Trace to SWS Item	SynchronizedTimeBaseManager: SWS_StbM_00247 TimeSyncOverCAN: SWS_CanTSyn_00072		
Requirements / Reference to Test Environment	Use Case UC01.02		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 16 CanTSyn: CanTSynRxCrcValidated = CRC_NOT_VALIDATED		
Summary	Aim is to: Verify that CanTSyn shall Recieve the offset time periodically via OFS and OFNS message respectively. Verify that StbM shall synchronize its local offset time base on receiption of Time Base from CanTSyn Verify that UT shall shall get the valid current time, offset time, current time in extended format, user data using Client-Server Interface.		
Needed Adaptation to other Releases	Not Applicable		
Pre-conditions	SUT shall be initialized.		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[LT<CAN>]** Transmit SYNC and FUP message with: TimeBaseId = 1 StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x00000E10 (3600d) StbM_TimeStampType.secondsHi = 0x0000		**[SUT]** Get the time stamp values as below: StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi Time updated in StbM, tA = tA +

		ToleranceTime_CanTSyn. ToleranceTime_CanTSyn = Max one main function of CanTSyn as reception is asynchronous.
Step 2	**[CP]** Wait 800ms	
Step 3	**[LT<CAN>]** Transmit OFS and OFNS message with: TimeBaseId = 16 StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x0000000A (10d) StbM_TimeStampType.secondsHi = 0x0000	**[SUT]** Get the offset time stamp values as below: StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 4	**[CP]** Wait 800ms	
Step 5	**[CP]** Start RUN_TimeUser	
Step 6	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 800ms> + <TestWaitTime = 800ms> StbM_TimeStampType.seconds = tA .Sec + <offset time = 0x0000000A> StbM_TimeStampType.secondsHi = tA .SecHi
Post-conditions	None	

4.4.11 [ATS_GTS_01272] Time Slave: Handling of Offset message reception timeout (CanTSynGlobalTimeFollowUpTimeout)

Test Objective	Time Slave: Handling of Offset message reception timeout (CanTSynGlobalTimeFollowUpTimeout)		
ID	ATS_GTS_01272	**AUTOSAR Releases**	4.2.1|4.2.2

Affected Modules	StbM, CanTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00131 ATR: ATR_ATR_00132 ATR: ATR_ATR_00133		
Trace to SWS Item	SynchronizedTimeBaseManager: SWS_StbM_00247 TimeSyncOverCAN: SWS_CanTSyn_00071		
Requirements / Reference to Test Environment	Use Case UC01.02		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 16 CanTSyn: CanTSynRxCrcValidated = CRC_IGNORED		
Summary	To verify that if OFNS message are not recieved within 'CanTSynGlobalTimeFollowUpTimeout' CanTSyn shall reset the sequence and wait for new OFS message. To verify StbM shall set the bit TIMEOUT in Sync state (when UT requests for current time) if StbM_BusSetGlobalTime is not invoked within StbMSyncLossTimeout and Clear the bit on invocation of StbM_BusSetGlobalTime.		
Needed Adaptation to other Releases	Not Applicable		
Pre-conditions	SUT shall be initialized. StbM shall be initialized with base time: StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x00000000 StbM_TimeStampType.secondsHi = 0x0000		

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[LT<CAN>]** Transmit OFS message with TimeBaseId = 16 StbM_TimeStampType.seconds =0x00000014 StbM_TimeStampType.secondsHi = 0x0000	**[SUT]** Receives OFS message with format Byte 0: Type = 0x30 Byte 1: Reserved Byte 2: D = 0x0 SC = 0x0 Byte 3: OfsTimeSecLsbHi = 8 Bit offset time stamp (LSB) from secondsHi Byte 4-7: OfsTimeSecLsbLo = StbM_TimeStampType.seconds
Step 2	**[LT<CAN>]**	**[SUT]**

	Transmit OFNS message with StbM_TimeStampType.nanoseconds = t4r	Receives OFNS message with format Byte 0: Type = 0x38 Byte 1: Reserved Byte 2: D = 0x0 SC = 0x0 Byte 3: OfsTimeSecMsbHi = 8 Bit offset time stamp (MSB) from secondsHi Byte 4-7: OfsTimeNSec = StbM_TimeStampType.nanoseconds
Step 3		**[SUT]** CanTSyn shall update the time base of StbM StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 4	**[CP]** Wait 800ms	
Step 5	**[CP]** Start RUN_TimeUser	
Step 6	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 800ms> StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 7	**[LT]** Transmit OFS message with	**[SUT]** Receives OFS message with format

	TimeBaseId = 16 StbM_TimeStampType.seconds =0x00000032 StbM_TimeStampType.secondsHi = 0x0000	Byte 0: Type = 0x30 Byte 1: Reserved Byte 2: D = 0x0 SC = 0x1 Byte 3: OfsTimeSecLsbHi = 8 Bit offset time stamp (LSB) from secondsHi Byte 4-7: OfsTimeSecLsbLo = StbM_TimeStampType.seconds
Step 8	**[LT]** Do not transmit OFNS message Reception timeout occurs after 300ms (CanTSynGlobalTimeFollowUpTimeout = 300ms) the sequence is reset and waits for a new OFS message.	**[SUT]** No OFNS message received StbM_TimeBaseStatusType.TIMEOUT is set to 1. CanTSyn shall not update the time base of StbM
Step 9	**[CP]** Wait 800ms	
Step 10	**[CP]** Start RUN_TimeUser	
Step 11	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 800ms> StbM_TimeStampType.seconds = tA .Sec + CanTSynGlobalTimeTxPeriod (2 second) StbM_TimeStampType.secondsHi = tA .SecHi
Post-conditions	None	

4.4.12 [ATS_GTS_01274] Time Slave: Handling of Offset message Sequence Mismatch failure

Test Objective	Time Slave: Handling of Offset message Sequence Mismatch failure

ID	ATS_GTS_01274	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, CanTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00131 ATR: ATR_ATR_00132 ATR: ATR_ATR_00133		
Trace to SWS Item	SynchronizedTimeBaseManager: SWS_StbM_00247 TimeSyncOverCAN: SWS_CanTSyn_00077 TimeSyncOverCAN: SWS_CanTSyn_00078		
Requirements / Reference to Test Environment	Use Case UC01.02		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 16 CanTSyn: CanTSynRxCrcValidated = CRC_IGNORED		
Summary	To verify that if Sequence Counter of OFS and OFNS message are not matching, CanTSyn shall discard the received OFS message and shall ignore the received OFNS message. To verify Sequence Counter Jump Width between two OFS messages greater than CanTSynGlobalTimeSequenceCounterJumpWidth, the messages will be ignored.		
Needed Adaptation to other Releases	Not Applicable		
Pre-conditions	SUT shall be initialized. StbM shall be initialized with base time: StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x00000000 StbM_TimeStampType.secondsHi = 0x0000		

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[LT<CAN>]** Transmit OFS message with TimeBaseId = 16 StbM_TimeStampType.seconds = 0x00000014 StbM_TimeStampType.secondsHi = 0x0000	**[SUT]** Receives OFS message with format Byte 0: Type = 0x30 Byte 1: Reserved Byte 2: D = 0x0 SC = 0x0 Byte 3: OfsTimeSecLsbHi = 8 Bit offset time stamp (LSB) from secondsHi

		Byte 4-7: OfsTimeSecLsbLo = StbM_TimeStampType.seconds
Step 2	**[LT<CAN>]** Transmit OFNS message with StbM_TimeStampType.nanoseconds = t4r	**[SUT]** Receives OFNS message with format Byte 0: Type = 0x38 Byte 1: Reserved Byte 2: D = 0x0 SC = 0x0 Byte 3: OfsTimeSecMsbHi = 8 Bit offset time stamp (MSB) from secondsHi Byte 4-7: OfsTimeNSec = StbM_TimeStampType.nanoseconds
Step 3		**[SUT]** CanTSyn shall update the offset time base of StbM StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 4	**[CP]** Wait 800ms	
Step 5	**[CP]** Start RUN_TimeUser	
Step 6	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 800ms>. StbM_TimeStampType.seconds = tA .Sec StbM_TimeBaseStatusType.TIMEOUT is set to 0. StbM_TimeStampType.secondsHi = tA .SecHi
Step 7	**[LT<CAN>]** Transmit next periodic OFS message	**[SUT]** Receives OFS message with format

	with TimeBaseId = 16 StbM_TimeStampType.seconds = 0x00000032 StbM_TimeStampType.secondsHi = 0x0000	Byte 0: Type = 0x30 Byte 1: Reserved Byte 2: D = 0x0 SC = 0x1 Byte 3: OfsTimeSecLsbHi = 8 Bit offset time stamp (LSB) from secondsHi StbM_TimeStampType.seconds
Step 8	**[LT<CAN>]** Transmit OFNS message with different sequence number	**[SUT]** Receives OFNS message with format Byte 0: Type = 0x38 Byte 1: Reserved Byte 2: D = 0x0 SC = 0x2 Byte 3: OfsTimeSecMsbHi = 8 Bit offset time stamp (MSB) from secondsHi Byte 4-7: OfsTimeNSec = t4 CanTSyn shall not update the offset time base of StbM
Step 9	**[CP]** Wait 800ms	
Step 10	**[CP]** Start RUN_TimeUser	
Step 11	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 800ms>. StbM_TimeStampType.seconds = tA .Sec + CanTSynGlobalTimeTxPeriod (2 second) StbM_TimeStampType.secondsHi = tA .SecHi StbM_TimeBaseStatusType.TIMEOUT is set to 1.

Step 12	**[LT]** Transmit next periodic OFS message with TimeBaseId = 16 StbM_TimeStampType.seconds = 0x00000050 StbM_TimeStampType.secondsHi = 0x0000 Sequence Counter = 3	**[SUT]** Time base is not updated as per new incoming time base as sequence counter value is greater than CanTSynGlobalTimeSequenceCounterJumpWidth
Step 13	**[CP]** Wait 800ms	
Step 14	**[CP]** Start RUN_TimeUser	
Step 15	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 800ms> + <TestWaitTime = 800ms>. StbM_TimeStampType.seconds = tA .Sec + CanTSynGlobalTimeTxPeriod (2 second) + CanTSynGlobalTimeTxPeriod (2 second StbM_TimeStampType.secondsHi = tA .SecHi StbM_TimeBaseStatusType.TIMEOUT is set to 1.
Post-conditions	None	

4.4.13 [ATS_GTS_01270] Time Slave: Synchronize Runnable entities to time base

Test Objective	Time Slave: Synchronize Runnable entities to time base		
ID	ATS_GTS_01270	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, CanTSyn	**State**	reviewed
Trace to Requirement on Acceptance	ATR: ATR_ATR_00132		

Test Document		
Trace to SWS Item	SynchronizedTimeBaseManager: SWS_StbM_00020 SynchronizedTimeBaseManager: SWS_StbM_00247	
Requirements / Reference to Test Environment	Use Case UC01.03	
Configuration Parameters	StbM: StbMMainFunctionPeriod 5ms StbMSynchronizedTimeBaseIdentifier = 1 StbMTriggeredCustomerPeriod 10ms StbMOSScheduleTableRef = Reference to OS ScheduleTable StbMSynchronizedTimeBaseRef = Reference to StbMSynchronizedTimeBase CanTSyn: CanTSynRxCrcValidated = CRC_IGNORED OS: ScheduleTableID = 1	
Summary	Aim is to verify that StbM shall synchronize the OS schedule tables (StbMOSScheduleTableRef) with updated time base as received from time master.	
Needed Adaptation to other Releases	Not Applicable	
Pre-conditions	SUT shall be initialized.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[LT]** Transmit SYNC message with TimeBaseId = 1 StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB	**[SUT]** Receives SYNC message with format Byte 0: Type = 0x10 Byte 1: User Byte 1 = 0xBB Byte 2: D = 0x1 SC = 0x0 Byte 3: User Byte 0 = 0xAA Byte 4-7: SyncTimeSec = StbM_TimeStampType.seconds
Step 2	**[LT]** Transmit FUP message with StbM_TimeStampType.nanoseconds = 0x00000000	**[SUT]** Receives FUP message with format Byte 0: Type = 0x18 Byte 1: Reserved Byte 2: D = 0x1 SC = 0x0

		Byte 3: reserved (Bit 7 to Bit 3) = 0 SGW (Bit 2) = 0 OVS = Overflow of seconds (Bit 1 to Bit 0) Byte 4-7: SyncTimeNSec = t4r
Step 3		**[SUT]** StbM updates its time base in 500ms with values StbM_TimeStampType.nanoseconds = tA.Nano + 500ms StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi Time updated in StbM, tA = tA + ToleranceTime_CanTSyn. ToleranceTime_CanTSyn = Max one main function of CanTSyn as reception is asynchronous.
Step 4	**[SUT]** StbM synchronizes OS schedule tables	**[SUT]** StbM shall invoke SyncScheduleTable(<ScheduleTableID = 1>, <Tick_value>)
Post-conditions	None	

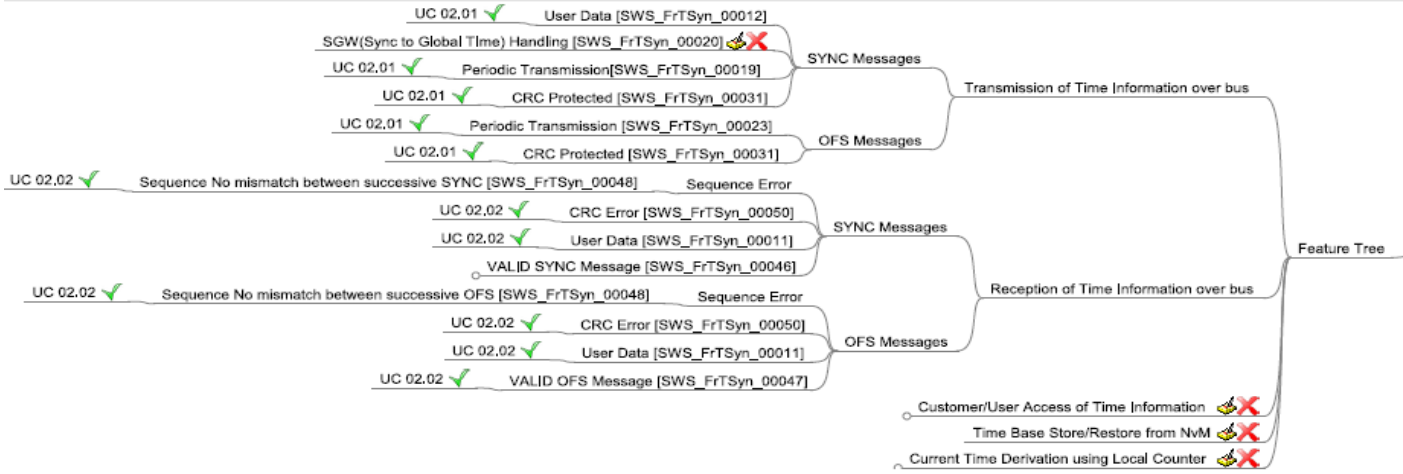
5 RS_BRF_01660 - Global Time Synchronization over FlexRay

5.1 General Test Objective and Approach

This Test Specification intends to cover the Global Time Synchronization feature of StbM and FrTSyn as described in the AUTOSAR Feature [RS_BRF_01660].

The tests use a test bench environment and Embedded Software Components that use the feature.

This test case document has been established to cover the following features:



Below Features are not tested in this test suite:

- SGW(Sync to Global TIme) Handling [SWS_FrTSyn_00020]: Feature is tested in the test suite 'Global Time Synchronization over Multiple Bus'.
- Customer/User Access of Time Information, Time Base Store/Restore from NvM, Current Time Derivation using Local Counter: These features are tested in test suite 'Global Time Synchronization over CAN'.

This specification gives the description of required tests environments (test bench, uses case, configuration files) and detailed tests cases for executing tests.

5.1.1 Test System

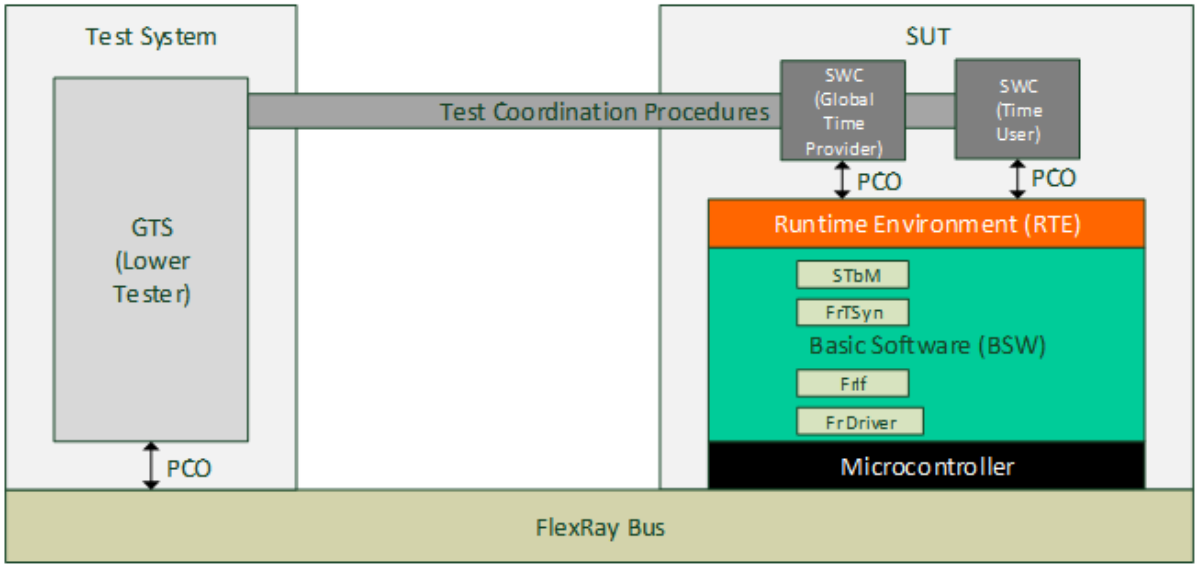
5.1.1.1 Overview on Architecture

In order to cover the required features / sub-features coverage, the environment has been separated in several use cases.

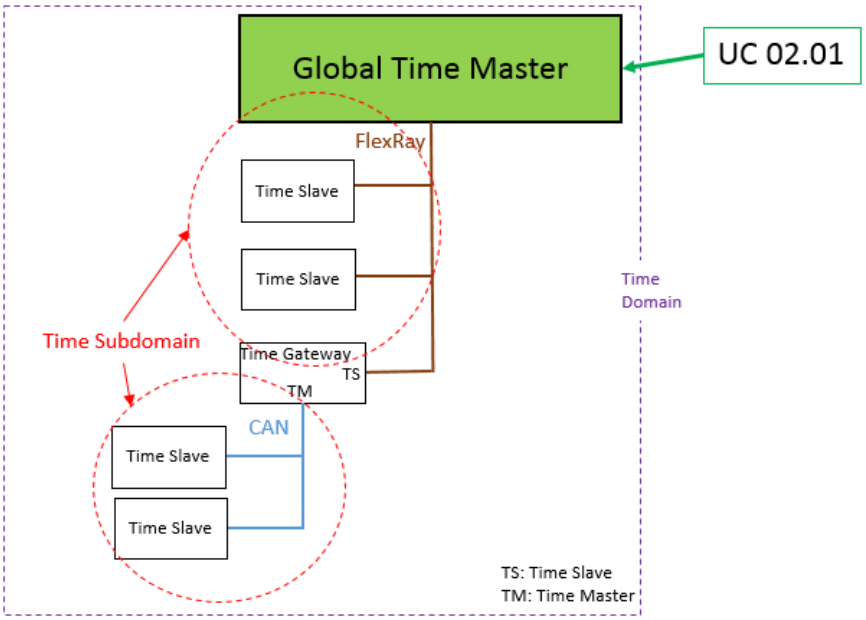
Test Cases are derived based on below use cases

5.1.1.1.1 UC 02.01: Global Time Master over FlexRay

SUT acts as Global time Master and Sets time base, offset time base, User Data and Trigger for transmission of Synchronization over FlexRay bus.

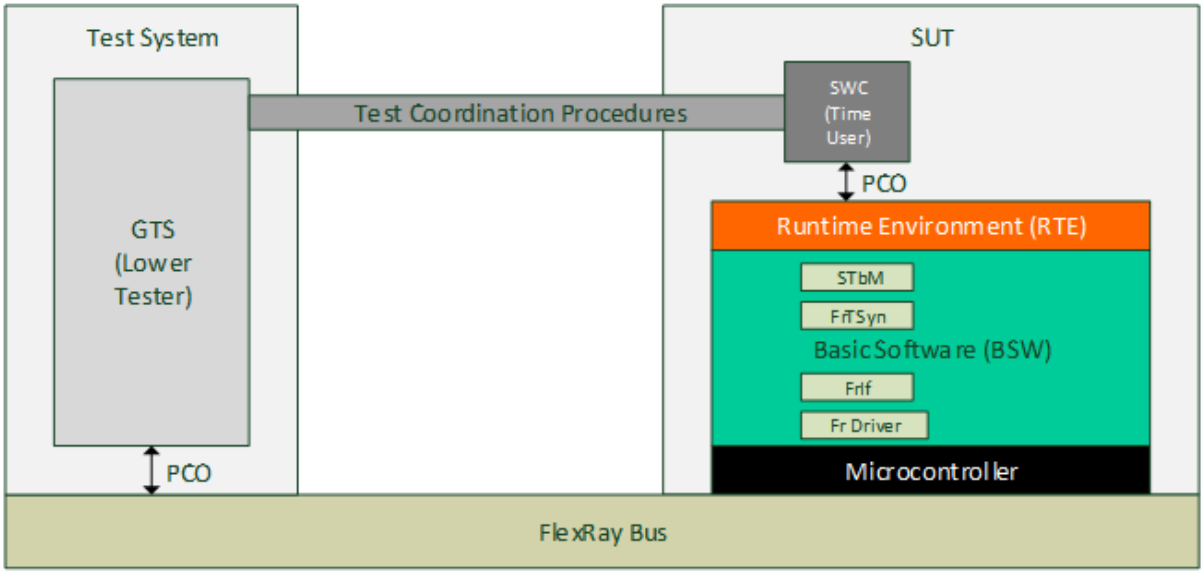


As shown in below figure, Functionalities of Global Time master of a time domain are tested over FlexRay in the use case 02.01

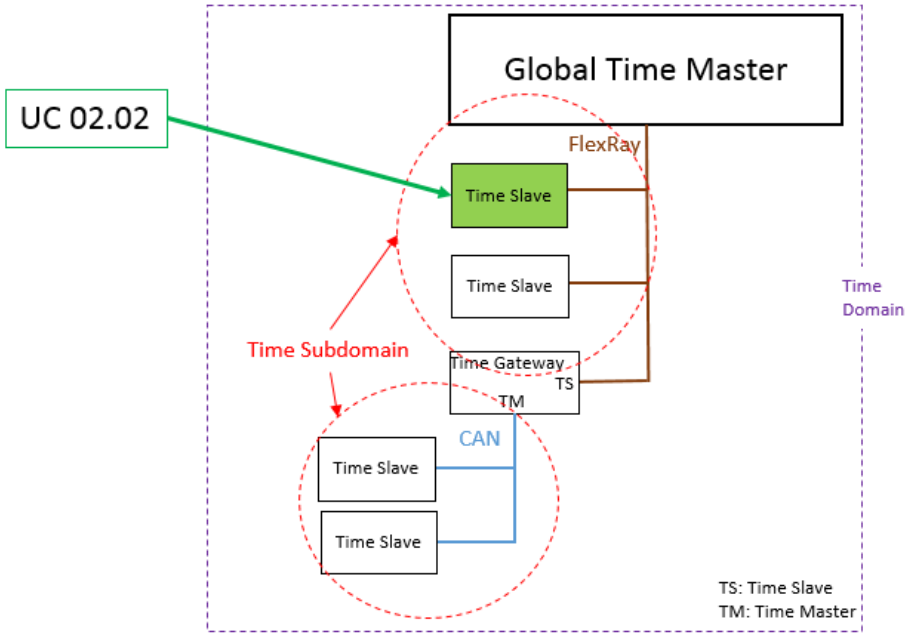


5.1.1.1.2 UC 02.02: Time Slave over FlexRay

SUT acts as Time Slave and Gets time base, offset time base and Synchronizes Local time to Global Time base.



As shown in below figure, Functionalities of Time Slave of a time domain are tested over FlexRay in the use case 02.02



5.1.1.2 Specific Requirements

Not Applicable.

5.1.1.3 Test Coordination Requirements

UC 01.01: Global Time Master over FlexRay

- Test System (LT <FlexRay>) shall read the FrTSyn FlexRay Frames and decode the same as per Frame Format provided in AUTOSAR_SWS_TimeSyncOverFlexRay.

UC 01.02: Time Slave over FlexRay

- Test System (LT <FlexRay>) shall encode the FrTSyn FlexRay Frames as per Frame Format provided in AUTOSAR_SWS_TimeSyncOver FlexRay and transmit over bus.

Requirements for CRC Calculation

- Test System (LT < FlexRay >) shall use the Crc_CalculateCRC8H2F() (Refer AUTOSAR Specification of CRC Routines AUTOSAR_SWS_CRCLibrary.pdf) to calculate the CRC of the Frame. Below are the parameters used for CRC calculation:
  The CRC start value shall be 0xFF.
  The CRC final XOR-value shall be 0xFF.
  The CRC polynomial shall be 0x2F.
  The DataIDList shall be same as provided in FrTSyn Configuration.

5.1.2 Test Case Design

Below diagrams explain test design for different use cases

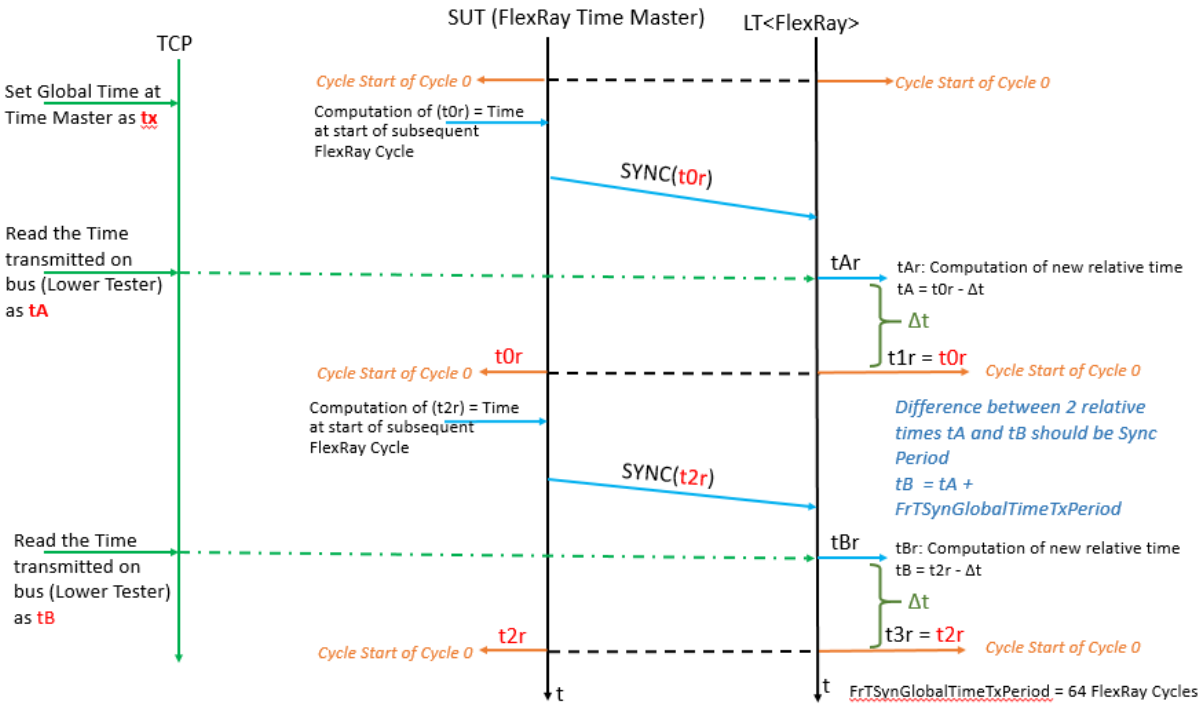


Fig: UC 02.01 - Global Time Master over FlexRay Test Design

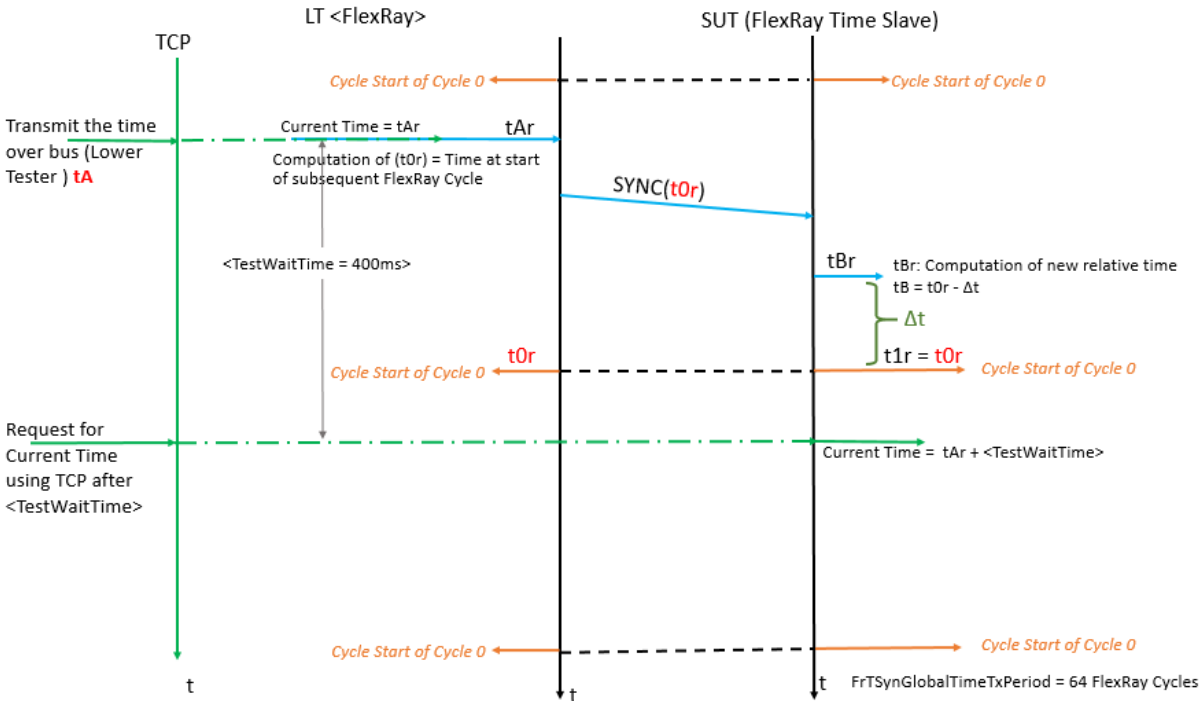


Fig: UC 02.02 - Time Slave over FlexRay Test Design

5.2 Configuration requirements

The configuration can be divided into two separate parts. The *test configuration* describes variables used to parameterize the test case. The *static configuration* describes the necessary settings of the DUT in order to allow a test case to perform.

5.2.1 Test Configuration

Communication data base for FrTSyn is depicted below

test configuration parameters	
Parameter	Value
FrIf Tx Pdu	AT_201_Ipdu201
FrIfTxPdu. FrIfImmediate	FALSE
FrIfFrameTriggering.FrIfBaseCycle	0
FrIfFrameTriggering.FrIfCycleRepetition	64
FrIfFrameTriggering.FrIfSlotId	2
FrIf Rx Pdu	AT_202_Ipdu202

5.2.2 Static Configuration

5.2.2.1 Static Configuration Groups

SCG_ATS_GlobalTimeSync_Time Master	
System Description Parameters	
StbM	
SystemTemplate::GlobalTime::GlobalTimeMaster.isSystemWideGlobalTimeMaster	TRUE

FrTSyn	
SystemTemplate::GlobalTime::GlobalTimeDomain.domainId	1
SystemTemplate::GlobalTime::GlobalTimeMaster.syncPeriod	320ms
SystemTemplate::GlobalTime::GlobalTimeDomain.subDomain	Ref. To PDU
ECU Configuration Parameters	
StbM	
StbM.StbMGeneral.StbMMainFunctionPeriod	5ms
StbM.StbMSynchronizedTimeBase.StbMLocalTimeRef	Ref. to OSCounter
FrTSyn	
FrTSyn.FrTSynGeneral.FrTSynMainFunctionPeriod	5ms
FrTSyn.FrTSynGlobalTimeDomain. FrTSynSynchronizedTimeBaseRef	Ref. to StbM time base
FrTSyn.CanTSynGlobalTimeDomain.FrTSynGlobalTimeMaster. FrTSynGlobalTimeMasterPdu. FrTSynGlobalTimeMasterConfirmationHandleId	0
Test Cases	
UC 02.01	

SCG_ATS_GlobalTimeSync_Time Slave	
System Description Parameters	
StbM	
SystemTemplate::GlobalTime::GlobalTimeMaster.isSystemWideGlobalTimeMaster	FALSE
FrTSyn	
SystemTemplate::GlobalTime::GlobalTimeDomain.subDomain	Ref. To PDU
ECU Configuration Parameters	
StbM	
StbM.StbMGeneral.StbMMainFunctionPeriod	5ms
StbM.StbMSynchronizedTimeBase.StbMLocalTimeRef	Ref. to OSCounter
FrTSyn	
FrTSyn.FrTSynGeneral.FrTSynMainFunctionPeriod	5ms
FrTSyn.FrTSynGlobalTimeDomain.FrTSynGlobalTimeDomainId	1
FrTSyn.FrTSynGlobalTimeDomain. FrTSynGlobalTimeSequenceCounterJumpWidth	1
FrTSyn.FrTSynGlobalTimeDomain. FrTSynSynchronizedTimeBaseRef	Ref. to StbM time base
FrTSyn.CanTSynGlobalTimeDomain.FrTSynGlobalTimeSlave. FrTSynGlobalTimeSlavePdu.FrTSynGlobalTimeSlaveHandleId	1
Test Cases	

UC 02.02	

5.2.2.2 Required System Description

Refer Section 4.2.2.1

5.2.2.3 Required ECU Configuration

Refer Section 4.2.2.1

5.2.2.4 Required Software Components

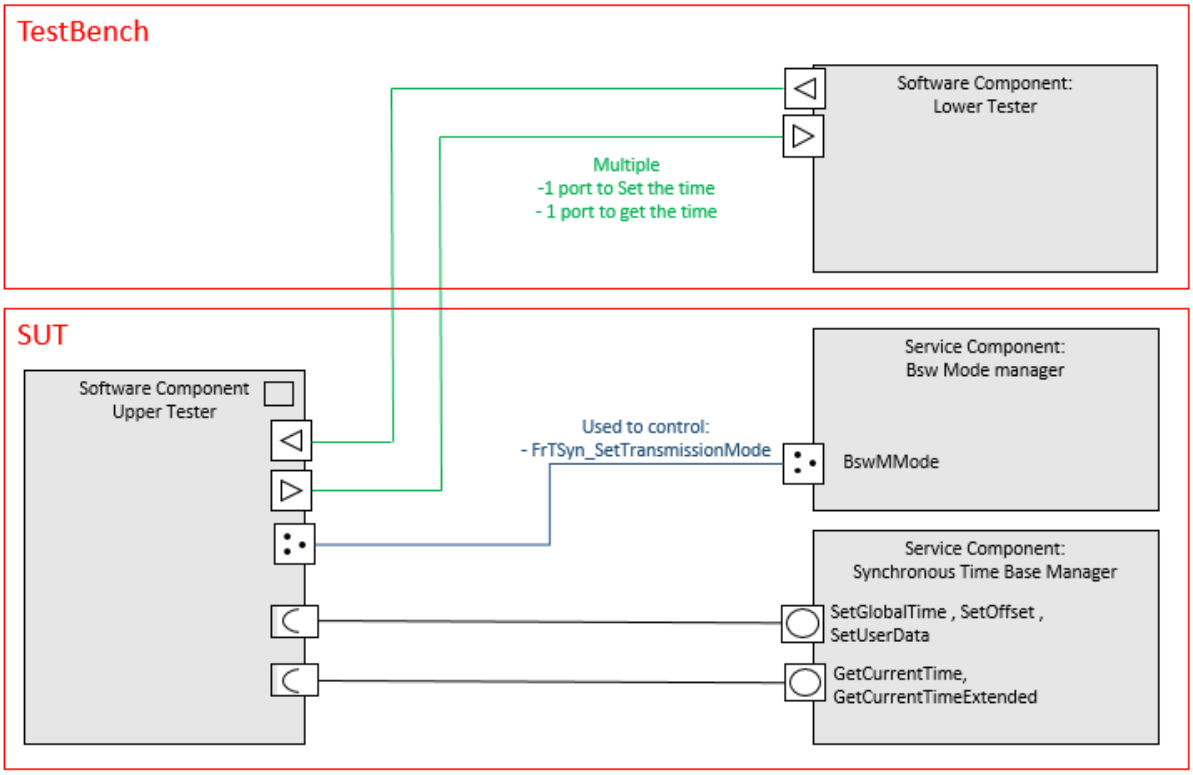


Fig: UC 02.01 - Global Time Master over FlexRay SWC Overview

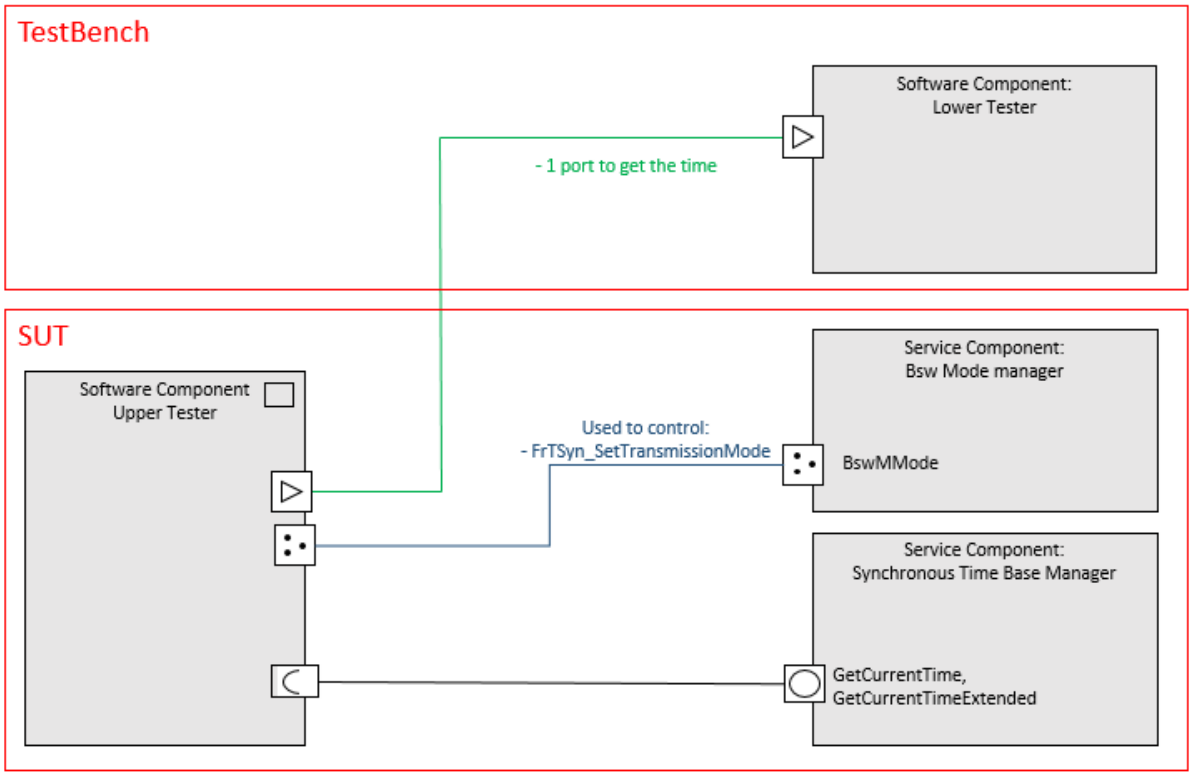


Fig: UC 02.02 - Time Slave over FlexRay SWC Overview

5.2.2.4.1 SWC Client GlobalTime_Provider

SWC Name	GlobalTime_Provider	
PORTS	**Name**	Client_SetGlobalTime
	Type	RPortPrototype
	Interface	GlobalTime_Master_Interface
	Requirements	
	Name	Client_SetUserData
	Type	RPortPrototype
	Interface	GlobalTime_Master_Interface
	Requirements	
	Name	Client_SetOffset
	Type	RPortPrototype
	Interface	GlobalTime_Master_Interface
	Requirements	

RUNNABLE ENTITIES	**Name**	RUN_GlobalTimeProvider	
	Requirements	Runnable shall be invoked by TCP	
	ServerCallPoint	**Name**	sscp_GlobalTimeProvider
		Type	SynchronousServerCallPoint
		Access to	Client_SetGlobalTime (Write operation) Client_SetUserData (Write operation) Client_SetOffset (Write operation)
		Requirements	

5.2.2.4.2 SWC Client Time_User

SWC Name	Time_User		
PORTS	**Name**	Client_GetCurrentTime	
	Type	RPortPrototype	
	Interface	GlobalTime_Slave_Interface	
	Requirements		
	Name	Client_GetCurrentTimeExtended	
	Type	RPortPrototype	
	Interface	GlobalTime_Slave_Interface	
	Requirements		
RUNNABLE ENTITIES	**Name**	RUN_TimeUser	
	Requirements	Runnable shall be invoked by TCP	
	ServerCallPoint	**Name**	sscp_TimeUser
		Type	SynchronousServerCallPoint
		Access to	Client_GetCurrentTime (Read operation) Client_GetCurrentTimeExtended (Read operation)

		Requirements	

5.2.2.4.3 SWC Server StbM

SWC Name	StbM		
PORTS	**Name**	Server_SetGlobalTime	
	Type	PPortPrototype	
	Interface	GlobalTime_Master_Interface	
	Requirements		
	Name	Server_SetOffset	
	Type	PPortPrototype	
	Interface	GlobalTime_Master_Interface	
	Requirements		
	Name	Server_SetUserData	
	Type	PPortPrototype	
	Interface	GlobalTime_Master_Interface	
	Requirements		
	Name	Server_GetCurrentTime	
	Type	PPortPrototype	
	Interface	GlobalTime_Slave_Interface	
	Requirements		
	Name	Server_GetCurrentTimeExtended	
	Type	PPortPrototype	
	Interface	GlobalTime_Slave_Interface	
	Requirements		
	Name	StbM_SetGlobalTime	
	Requirements		
		Name	OIE_ SetGlobalTime

RUNNABLE ENTITIES	**Started by Event**	**Type**	OperationInvokedEvent Port:Server_SetGlobalTime Operation: Read
		Requirements	
	Name	StbM_SetOffset	
	Requirements		
	ServerCallPoint	**Name**	OIE_SetOffset
		Type	OperationInvokedEvent Port: Server_SetOffset Operation: Read
		Requirements	
	Name	StbM_SetUserData	
	Requirements		
	Started by Event	**Name**	OIE_ SetUserData
		Type	OperationInvokedEvent Port: Server_SetUserData Operation: Read
		Requirements	
	Name	StbM_GetCurrentTime	
	Requirements		
	Started by Event	**Name**	OIE_ GetCurrentTime
		Type	OperationInvokedEvent Port: Server_GetCurrentTime Operation: Read
		Requirements	
	Name	StbM_GetCurrentTimeExtended	
	Requirements		
		Name	OIE_GetCurrentTimeExtended

	Started by Event		
		Type	OperationInvokedEvent Port: Server_GetCurrentTimeExtended Operation: Read
		Requirements	

5.3 Re-usable Test Steps

Not Applicable

5.4 Test Cases

5.4.1 [ATS_GTS_01275] Global Time Master: Setting of Global Time base and user data by Active Customer and sending of SYNC frames (CRC_NOT_SUPPORTED) over FlexRay

Test Objective	Global Time Master: Setting of Global Time base and user data by Active Customer and sending of SYNC frames (CRC_NOT_SUPPORTED) over FlexRay		
ID	ATS_GTS_01275	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, FrTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00134		
Trace to SWS Item	TimeSyncOverFlexRay: SWS_FrTSyn_00023		
Requirements / Reference to Test Environment	Use Case UC02.01		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 1 FrTSyn: FrTSynGlobalTimeTxCrcSecured = CRC_NOT_SUPPORTED CanTSynGlobalTimeSyncDataIDListIndex = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} CanTSynGlobalTimeSyncDataIDListValue in ASCII = {"A", "U", "T", "O", "S", "A", "R", "A", "T", "S", "G", "T", "S", "S", "Y", "N"} CanTSynGlobalTimeFupDataIDListIndex = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} CanTSynGlobalTimeFupDataIDListValue in ASCII = {"A", "U", "T", "O", "S", "A", "R", "A", "T", "S", "G", "T", "S", "F", "U", "P"}		
Summary	Aim is to: Verify that StbM accepts the global time base and user data from SWC Global Time Provider using client-server Interface. Verify that FrTSyn shall transmit the global time base to time slave periodically via SYNC message		
Needed Adaptation to	Not Applicable		

other Releases		
Pre-conditions	SUT shall be initialized.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Start RUN_ GlobalTimeProvider.	
Step 2	**[RUN<RUN_ GlobalTimeProvider>]** Execute Rte_Call_ Client_SetGlobalTime and Rte_Call_ Client_SetUserData with below values: timeBaseId = 1 StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB StbM_UserDataType.User Data Byte 2 = 0xCC	**[RUN<RUN_ GlobalTimeProvider>]** Rte_Call returns RTE_E_OK.
Step 3	**[CP]** Wait 100ms.	
Step 4	**[CP]** Start RUN_TimeUser.	
Step 5	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime.	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time received from Time user shall be as mentioned below: StbM_TimeStampType.nanoseconds = 0x00000000 + <TestWaitTime = 100ms> StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000

		StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB StbM_UserDataType.User Data Byte 2 = 0xCC
Step 6	**[LT<FLEXRAY>]** Receives the SYNC message without CRC validation.	**[LT]** Receives frame format as mentioned below: Byte 0: Type = 0x10 Byte 1: UserByte 2 Byte 2: D = 0x1 SC = 0x0 Byte 3: FCNT= FlexRay Cycle Counter 0 (Bit 7 to Bit 2) SGW (Bit 1) SyncToSubDomain = 1 reserved (Bit 0), default: 0 Byte 4: User Byte 0 Byte 5: User Byte 1 Byte 6-11: SyncTimeSec = 48 Bit time stamp in seconds Byte 12-15: SyncTimeNSec = 32 Bit time stamp in nanoseconds
Step 7	**[LT<FLEXRAY>]** Receives Global time base. Store the time as base for next periodic message processing	**[LT]** Get the time stamp values as below: StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 8	**[LT<FLEXRAY>]** Receives the next periodic SYNC message without CRC validation	**[LT]** Time stamp values shall be as

		mentioned below: StbM_TimeStampType.nanoseconds = tB.Nano StbM_TimeStampType.seconds = tB.Sec StbM_TimeStampType.secondsHi = tB.SecHi StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB StbM_UserDataType.User Data Byte 2 = 0xCC The difference between two time base shall be tB – tA = FrTSynGlobalTimeTxPeriod (320ms).
Post-conditions	None	

5.4.2 [ATS_GTS_01287] Global Time Master: Setting of Offset Time base by Active Customer and sending Offset frames (CRC_SUPPORTED) over FlexRay

Test Objective	Global Time Master: Setting of Offset Time base by Active Customer and sending Offset frames (CRC_SUPPORTED) over FlexRay		
ID	ATS_GTS_01287	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, FrTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00134		
Trace to SWS Item	TimeSyncOverFlexRay: SWS_FrTSyn_00031		
Requirements / Reference to Test Environment	Use Case UC02.01		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 17 FrTSyn: FrTSynGlobalTimeTxCrcSecured = CRC_SUPPORTED		

Summary	Aim is to: Verify that StbM accepts the global time base from Upper Tester using client-server Interface. Verify that FrTSyn shall transmit the offset time base to time slave periodically via OFS message with CRC secured.	
Needed Adaptation to other Releases	Not Applicable	
Pre-conditions	SUT shall be initialized.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Start RUN_ GlobalTimeProvider	
Step 2	**[RUN<RUN_ GlobalTimeProvider>]** Execute Rte_Call_ Client_ SetGlobalTime with below values: TimeBaseId = 1 StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x00000E10 (3600d) StbM_TimeStampType.secondsHi = 0x0000	**[RUN<RUN_ GlobalTimeProvider>]** Rte_Call returns RTE_E_OK
Step 3	**[CP]** Wait 100ms	
Step 4	**[RUN<RUN_ GlobalTimeProvider>]** Execute Rte_Call_ Client_ SetOffset with below values: timeBaseId = 17 StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x00000064 StbM_TimeStampType.secondsHi = 0x0000	**[RUN<RUN_ GlobalTimeProvider>]** Rte_Call returns RTE_E_OK
Step 5	**[CP]** Wait 100ms	
Step 6	**[CP]** Start RUN_TimeUser	

Step 7	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime Time base = 17	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time received from Time user shall be as mentioned below: StbM_TimeStampType.nanoseconds = 0x00000000 + <TestWaitTime = 100ms> + <TestWaitTime = 100ms> StbM_TimeStampType.seconds = 0x00000E10 + 0x00000064 StbM_TimeStampType.secondsHi = 0x0000
Step 8	**[LT<FLEXRAY>]** Receives the OFS message with CRC as per test co-ordination requirement for CRC.	**[LT]** Receives OFS message with format Byte 0: Type = 0x40 Byte 1: CRC Byte 2: D = Time Domain 16 (Bit 7 to Bit 4) SC = <Sequence Counter 0x01>(Bit 3 to Bit 0) Byte 3: User Byte 0 Byte 4: User Byte 1 Byte 5: User Byte 2 Byte 6-11: OfsTimeSec = <StbM_TimeStampType.seconds 0x000000000064> Byte 12-15: OfsTimeNSec = <StbM_TimeStampType.nanoseconds = 0x00000000>
Post-conditions	None	

5.4.3 [ATS_GTS_01249] Time Slave: Reception of SYNC frames (CRC_VALIDATED) over FlexRay, Synchronize Local Time Base and share the current time to active customers

Test Objective	Time Slave: Reception of SYNC frames (CRC_VALIDATED) over FlexRay, Synchronize Local Time Base and share the current time to active customers		
ID	ATS_GTS_01249	**AUTOSAR**	4.2.1|4.2.2

		Releases	
Affected Modules	StbM, FrTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00134		
Trace to SWS Item	TimeSyncOverFlexRay: SWS_FrTSyn_00050		
Requirements / Reference to Test Environment	Use Case UC02.02		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 1 FrTSyn: FrTSynRxCrcValidated = CRC_VALIDATED		
Summary	Verify that FrTSyn shall receive Synchronization frames after validating CRC. Verify that StbM shall synchronize its local time base on reception of Time Base from FrTSyn. Verify that Upper Tester shall get the valid current time, current time in extended format, user data using Client-Server Interface.		
Needed Adaptation to other Releases	Not Applicable		
Pre-conditions	SUT shall be initialized.		
Main Test Execution			
Test Steps		**Pass Criteria**	
Step 1	**[LT<FLEXRAY>]** Transmit SYNC message with timeBaseId = 1 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB Byte CRC = Calculated CRC And time value as T0 Where T0 = TSYNC + (MacroticksPerCycle * (64 - currentCycle) - currentMacroticks) * MacrotickDuration And TSYNC value as given below: StbM_TimeStampType.seconds =	**[SUT]** Receives SYNC message with CRC value in below format Byte 0: Type = 0x20 Byte 1: CRC Byte 2: D = <Time Domain = 1> (Bit 7 to Bit 4) SC = <Sequence Counter = 0> (Bit 3 to Bit 0) Byte 3: FCNT= <FlexRay Cycle Counter = 0> (Bit 7 to Bit 2) SGW (Bit 1) SyncToSubDomain = 1 reserved (Bit 0)	

	0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_TimeStampType.nanoseconds = 0x00000000	Byte 4: User Byte 0 Byte 5: User Byte 1 Byte 6-11: SyncTimeSec = T0.secondsHi and T0.seconds Byte 12-15: SyncTimeNSec = T0.nanoseconds
Step 2		**[SUT]** StbM updates its time base with values StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi Time updated in StbM, tA = tA + ToleranceTime_FrTSyn. ToleranceTime_FrTSyn = Max one main function of FrTSyn as reception is asynchronous.
Step 3	**[CP]** Wait 200ms	
Step 4	**[CP]** Start RUN_TimeUser	
Step 5	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 200ms> StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Post-conditions	None	

5.4.4 [ATS_GTS_01250] Time Slave: Reception of SYNC frames (CRC_IGNORED) over FlexRay, Synchronize Local Time Base and share the current time to active customers.

Test Objective	Time Slave: Reception of SYNC frames (CRC_IGNORED) over FlexRay, Synchronize Local Time Base and share the current time to active customers.		
ID	ATS_GTS_01250	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, FrTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00134		
Trace to SWS Item	TimeSyncOverFlexRay: SWS_FrTSyn_00046		
Requirements / Reference to Test Environment	Use Case UC02.02		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 1 FrTSyn: FrTSynRxCrcValidated = CRC_IGNORED		
Summary	Aim is to Verify that FrTSyn shall call StbM to update time base on reception of SYNC message even with invalid CRC when FrTSynRxCrcValidated = CRC_IGNORED.		
Needed Adaptation to other Releases	Not Applicable		
Pre-conditions	SUT shall be initialized.		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[LT<FLEXRAY>]** Transmit SYNC message with timeBaseId = 1 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB And time value as T0 Where T0 = TSYNC + (MacroticksPerCycle * (64 - currentCycle) - currentMacroticks) * MacrotickDuration And TSYNC value as given below: StbM_TimeStampType.seconds =		**[SUT]** Receives SYNC message ignoring CRC value (if there) in below format Byte 0: Type = 0x10 Byte 1: User Byte 2 Byte 2: D = <Time Domain = 1> (Bit 7 to Bit 4) SC = <Sequence Counter = 0> (Bit 3 to Bit 0) Byte 3: FCNT= <FlexRay Cycle Counter = 0> (Bit 7 to Bit 2) SGW (Bit 1)

	0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_TimeStampType.nanoseconds = 0x00000000	SyncToSubDomain = 1 reserved (Bit 0) Byte 4: User Byte 0 Byte 5: User Byte 1 Byte 6-11: SyncTimeSec = T0.secondsHi and T0.seconds Byte 12-15: SyncTimeNSec = T0.nanoseconds
Step 2		**[SUT]** StbM updates its time base with values StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi Time updated in StbM, tA = tA + ToleranceTime_FrTSyn. ToleranceTime_FrTSyn = Max one main function of FrTSyn as reception is asynchronous.
Step 3	**[CP]** Wait 200ms	
Step 4	**[CP]** Start RUN_TimeUser	
Step 5	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 200ms> StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Post-	None	

conditions	

5.4.5 [ATS_GTS_01256] Time Slave: Handling of SYNC message Sequence Mismatch failures

Test Objective	Time Slave: Handling of SYNC message Sequence Mismatch failures		
ID	ATS_GTS_01256	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, FrTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00134		
Trace to SWS Item	TimeSyncOverFlexRay: SWS_FrTSyn_00048		
Requirements / Reference to Test Environment	Use Case UC02.02		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 1 FrTSyn: FrTSynRxCrcValidated = CRC_IGNORED		
Summary	Aim is to verify that if Sequence Counter Jump Width between two SYNC messages is greater than FrTSynGlobalTimeSequenceCounterJumpWidth, then messages will be ignored.		
Needed Adaptation to other Releases	Not Applicable		
Pre-conditions	SUT shall be initialized.		
Main Test Execution			
Test Steps		**Pass Criteria**	
Step 1	**[LT<FLEXRAY>]** Transmit SYNC message with TimeBaseId = 1 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB	**[SUT]** Receives SYNC message ignoring CRC value in below format Byte 0: Type = 0x10 Byte 1: User Byte 2 Byte 2: D = <Time Domain = 1> (Bit 7 to Bit 4) SC = <Sequence Counter = 0> (Bit 3 to Bit 0)	

	And time value as T0a Where T0a = TSYNC + (MacroticksPerCycle * (64 - currentCycle) - currentMacroticks) * MacrotickDuration And TSYNC value as given below: StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_TimeStampType.nanoseconds = 0x00000000	Byte 3: FCNT= <FlexRay Cycle Counter = 0> (Bit 7 to Bit 2) SGW (Bit 1) SyncToSubDomain = 1 reserved (Bit 0) Byte 4: User Byte 0 Byte 5: User Byte 1 Byte 6-11: SyncTimeSec = T0a.secondsHi and T0a.seconds Byte 12-15: SyncTimeNSec = T0a.nanoseconds
Step 2		**[SUT]** FrTSyn shall update the time base of StbM StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi Time updated in StbM, tA = tA + ToleranceTime_FrTSyn. ToleranceTime_FrTSyn = Max one main function of FrTSyn as reception is asynchronous.
Step 3	**[CP]** Wait 200ms	
Step 4	**[CP]** Start RUN_TimeUser	
Step 5	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 200ms>. StbM_TimeStampType.seconds = tA .Sec

		StbM_TimeStampType.secondsHi = tA .SecHi StbM_TimeBaseStatusType.TIMEOUT is set to 0.
Step 6	**[LT<FLEXRAY>]** Transmit next periodic SYNC message with timeBaseId = 1 StbM_UserDataType.User Data Byte 0 = 0xEE StbM_UserDataType.User Data Byte 1 = 0xFF And time value as T0b Where T0b = TSYNC + (MacroticksPerCycle * (64 - currentCycle) - currentMacroticks) * MacrotickDuration And TSYNC value as given below: StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_TimeStampType.nanoseconds = 0x00000000 Sequence counter = 3	**[SUT]** Time base is not updated as per incoming time base as sequence counter value is greater than FrTSynGlobalTimeSequenceCounterJumpWidth
Step 7	**[CP]** Wait 200ms	
Step 8	**[CP]** Start RUN_TimeUser	
Step 9	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 200ms> + <TestWaitTime = 200ms>. StbM_TimeStampType.seconds = tA .Sec + FrTSynGlobalTimeTxPeriod (320ms)

		StbM_TimeStampType.secondsHi = tA .SecHi StbM_TimeBaseStatusType.TIMEOUT is set to 1.
Post-conditions	None	

5.4.6 [ATS_GTS_01277] Time Slave: Reception of Offset frames (CRC_NOT_VALIDATED) over FlexRay, Synchronize Local Time Base and share the current time to active customers.

Test Objective	Time Slave: Reception of Offset frames (CRC_NOT_VALIDATED) over FlexRay, Synchronize Local Time Base and share the current time to active customers.		
ID	ATS_GTS_01277	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, FrTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00134		
Trace to SWS Item	TimeSyncOverFlexRay: SWS_FrTSyn_00011 TimeSyncOverFlexRay: SWS_FrTSyn_00047		
Requirements / Reference to Test Environment	Use Case UC02.02		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 16 FrTSyn: FrTSynRxCrcValidated = CRC_NOT_VALIDATED CanTSynGlobalTimeOfsDataIDListIndex = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} CanTSynGlobalTimeOfsDataIDListValue in ASCII = {"A", "U", "T", "O", "S", "A", "R", "A", "T", "S", "G", "T", "S", "O", "F", "S"} CanTSynGlobalTimeOfnsDataIDListIndex = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} CanTSynGlobalTimeOfnsDataIDListValue in ASCII = {"A", "U", "T", "O", "S", "A", "R", "A", "T", "S", "G", "T", "O", "F", "N", "S"}		
Summary	Aim is to: Verify that FrTSyn shall Recieve the offset time base periodically via offset message. Verify that StbM shall synchronize its local offset time base on receiption of Time Base from FrTSyn.		
Needed Adaptation to other Releases	Not Applicable		
Pre-conditions	SUT shall be initialized.		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[LT<FLEXRAY>]**		**[SUT]**

	Transmit SYNC message with: TimeBaseId = 1 And time value as T0 Where T0 = TSYNC + (MacroticksPerCycle * (64 - currentCycle) - currentMacroticks) * MacrotickDuration And TSYNC value as given below: StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_TimeStampType.nanoseconds = 0x00000000	Get the time stamp values as below: StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi Time updated in StbM, tA = tA + ToleranceTime_FrTSyn. ToleranceTime_FrTSyn = Max one main function of FrTSyn as reception is asynchronous.
Step 2	**[CP]** Wait 200ms	
Step 3	**[LT<FLEXRAY>]** Transmit OFS message with: TimeBaseId = 16 StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x0000000A (10d) StbM_TimeStampType.secondsHi = 0x0000	**[SUT]** Get the offset time stamp values as below: StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 4	**[CP]** Wait 200ms	
Step 5	**[CP]** Start RUN_TimeUser	
Step 6	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 200ms> + <TestWaitTime = 200ms> StbM_TimeStampType.seconds = tA .Sec + <offset time = 0x0000000A>

		StbM_TimeStampType.secondsHi = tA .SecHi
Post-conditions	None	

5.4.7 [ATS_GTS_01280] Time Slave: Handling of Offset message Sequence Mismatch failures

Test Objective	Time Slave: Handling of Offset message Sequence Mismatch failures		
ID	ATS_GTS_01280	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, FrTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00134		
Trace to SWS Item	TimeSyncOverFlexRay: SWS_FrTSyn_00048		
Requirements / Reference to Test Environment	Use Case UC02.02		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 16 FrTSyn: FrTSynRxCrcValidated = CRC_IGNORED		
Summary	Aim is to verify that if Sequence Counter Jump Width between two OFS messages is greater than FrTSynGlobalTimeSequenceCounterJumpWidth, then messages will be ignored.		
Needed Adaptation to other Releases	Not Applicable		
Pre-conditions	SUT shall be initialized. StbM shall be initialized with base time: StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x00000000 StbM_TimeStampType.secondsHi = 0x0000		
Main Test Execution			
Test Steps		**Pass Criteria**	
Step 1	**[LT<FLEXRAY>]** Transmit OFS message with TimeBaseId = 16 StbM_TimeStampType.nanoseconds	**[SUT]** Receives OFS message with format Byte 0: Type = 0x30	

	= 0x00000000 StbM_TimeStampType.seconds = 0x00000014 StbM_TimeStampType.secondsHi = 0x0000	Byte 1: User Byte 3 Byte 2: D = <Time Domain = 16> (Bit 7 to Bit 4) SC = <Sequence Counter = 0> (Bit 3 to Bit 0) Byte 3: User Byte 0 Byte 4: User Byte 1 Byte 5: User Byte 2 Byte 6-11: OfsTimeSec = 0x000000000014 Byte 12-15: OfsTimeNSec = 0x00000000
Step 2		**[SUT]** FrTSyn shall update the offset time base of StbM StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 3	**[CP]** Wait 200ms	
Step 4	**[CP]** Start RUN_TimeUser	
Step 5	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime.	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 200ms>. StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi StbM_TimeBaseStatusType.TIMEOUT is set to 0.
Step 6	**[LT]** Transmit next periodic OFS message with TimeBaseId = 16 StbM_TimeStampType.seconds =	**[SUT]** Time base is not updated as per new incoming time base as sequence counter value is greater than FrTSynGlobalTimeSequenceCounterJumpWidth.

	0x00000050 StbM_TimeStampType.secondsHi = 0x0000 Sequence number = 3	
Step 7	**[CP]** Wait 200ms.	
Step 8	**[CP]** Start RUN_TimeUser	
Step 9	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 200ms> + <TestWaitTime = 200ms>. StbM_TimeStampType.seconds = tA .Sec + FrTSynGlobalTimeTxPeriod (320ms) + FrTSynGlobalTimeTxPeriod (320ms) StbM_TimeStampType.secondsHi = tA .SecHi StbM_TimeBaseStatusType.TIMEOUT is set to 1.
Post-conditions	None	

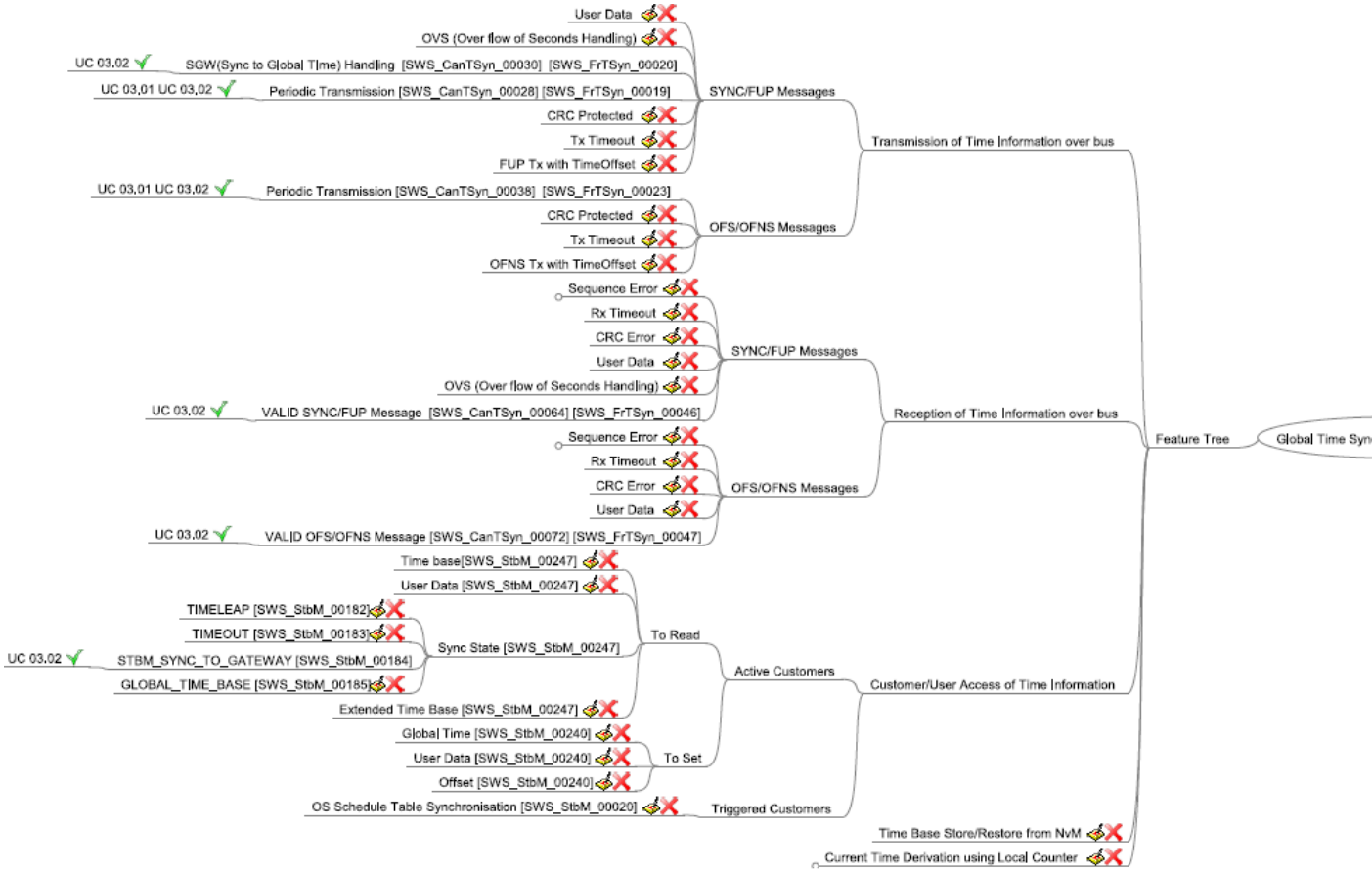
6 RS_BRF_01660 - Global Time Synchronization over Multiple Bus

6.1 General Test Objective and Approach

This Test Specification intends to cover the Global Time Synchronization feature of StbM CanTSync, and FrTSync as described in the AUTOSAR Feature [RS_BRF_01660].

The tests use a test bench environment and Embedded Software Components that use the feature.

This test case document has been established to cover the following features:



Features not tested in this test suite are either tested in 'Global Time Synchronization over CAN' or 'Global Time Synchronization over FlexRay'.

This specification gives the description of required tests environments (test bench, uses case, configuration files) and detailed tests cases for executing tests.

6.1.1 Test System

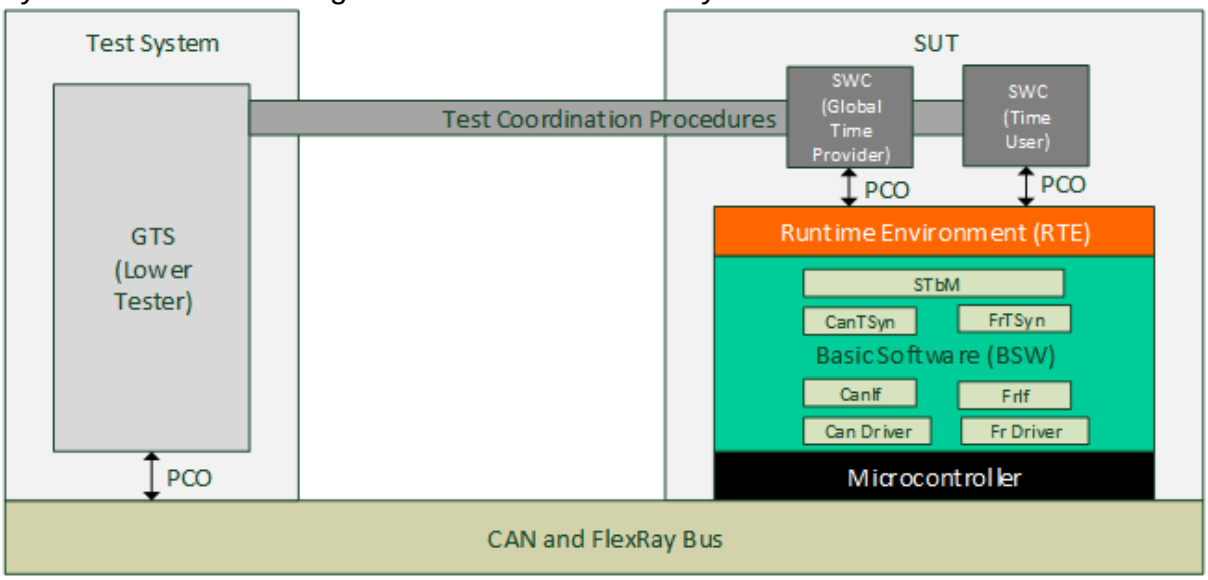
6.1.1.1 Overview on Architecture

In order to cover the required features / sub-features coverage, the environment has been separated in several use cases.

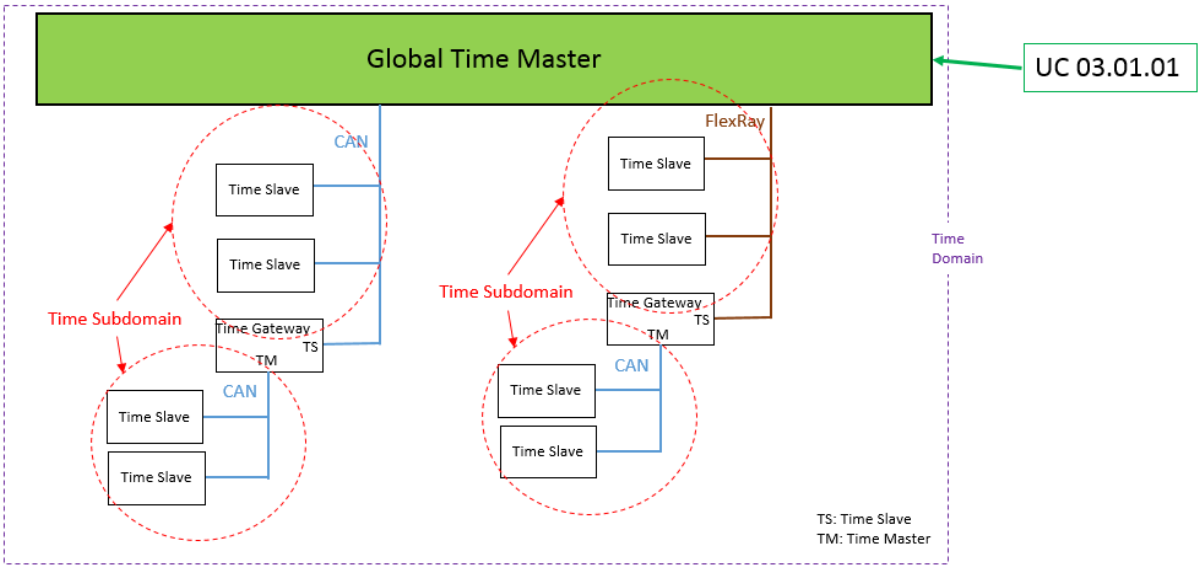
Test Cases are derived based on below use cases

6.1.1.1.1 UC 03.01.01 Global Time Master (Single Time Domain) over both CAN and FlexRay

SUT acts as Global time Master over Single time domain and Transmits Time Synchronization messages over CAN and FlexRay bus.

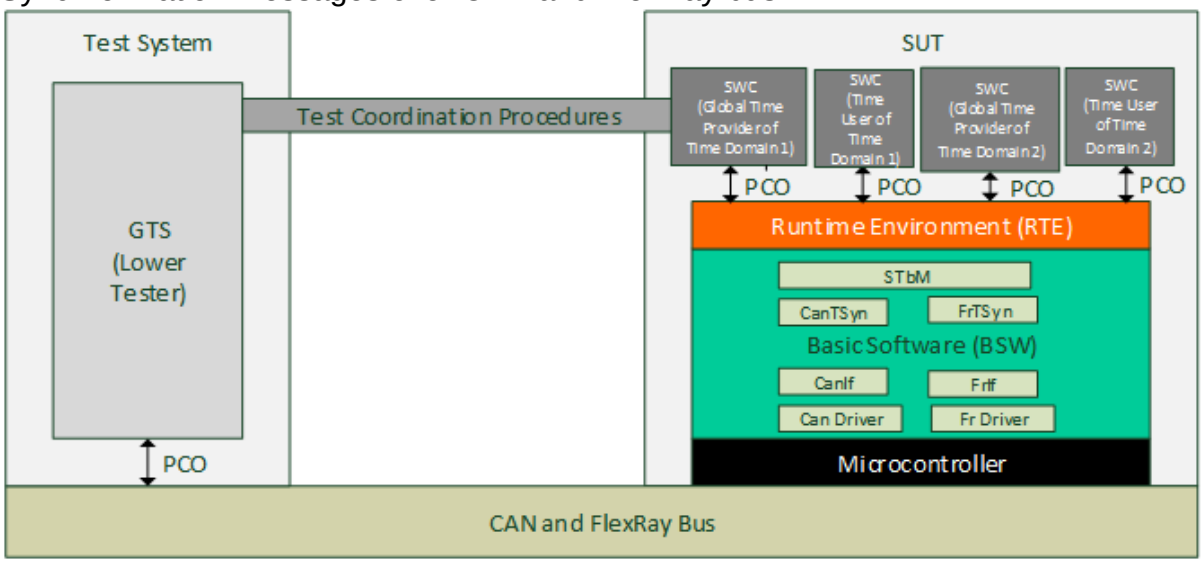


As shown in below figure, Functionalities of Global Time master for single time domain are tested over CAN and FlexRay in the use case 03.01.01

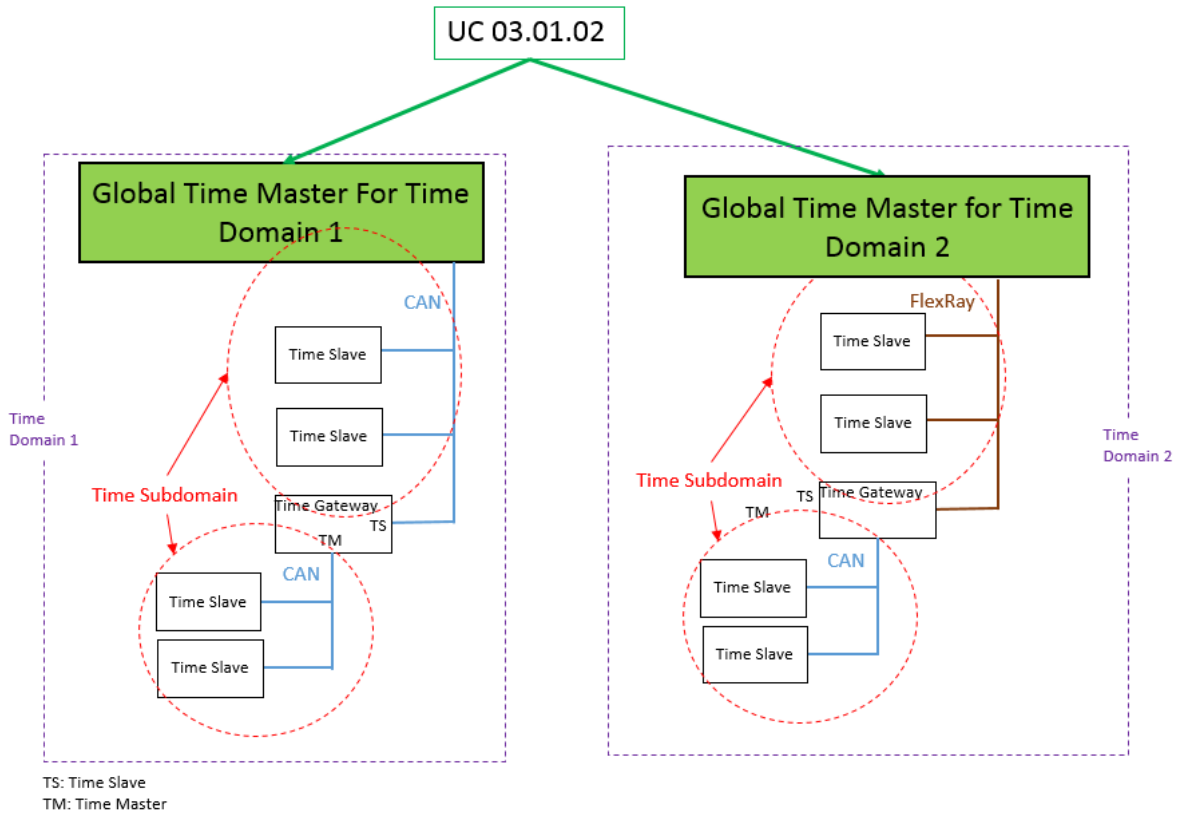


6.1.1.1.2 UC 03.01.02 Global Time Master (Multiple Time Domain) over both CAN and FlexRay

SUT acts as Global time Master over multiple time domain and Transmits Time Synchronization messages over CAN and FlexRay bus.

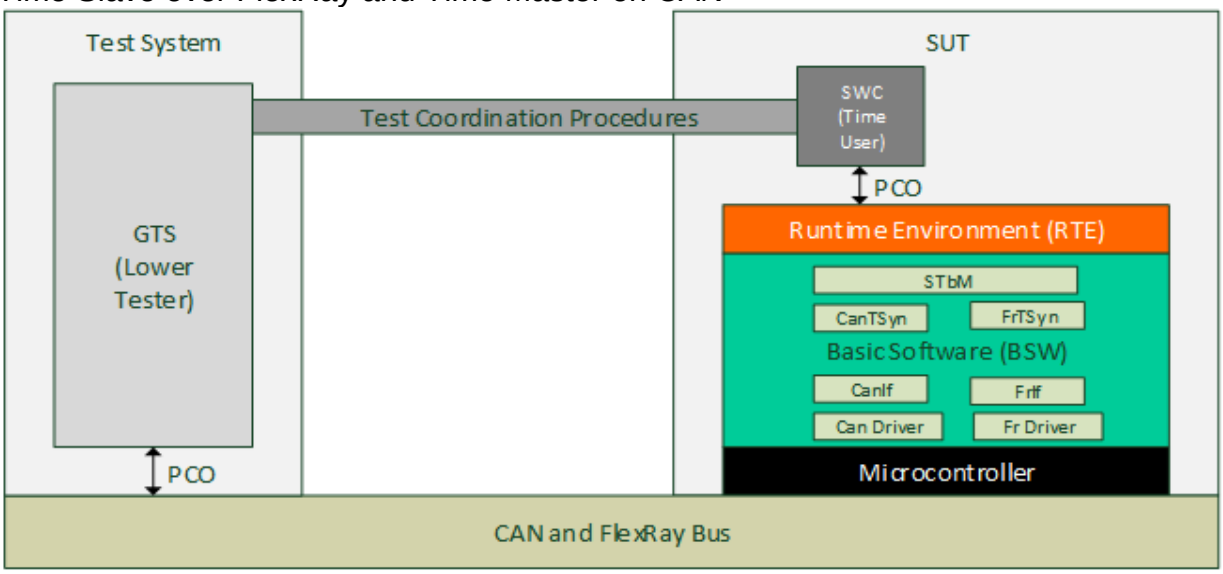


As shown in below figure, Functionalities of Global Time master for multiple time domain are tested over CAN and FlexRay in the use case 03.01.02

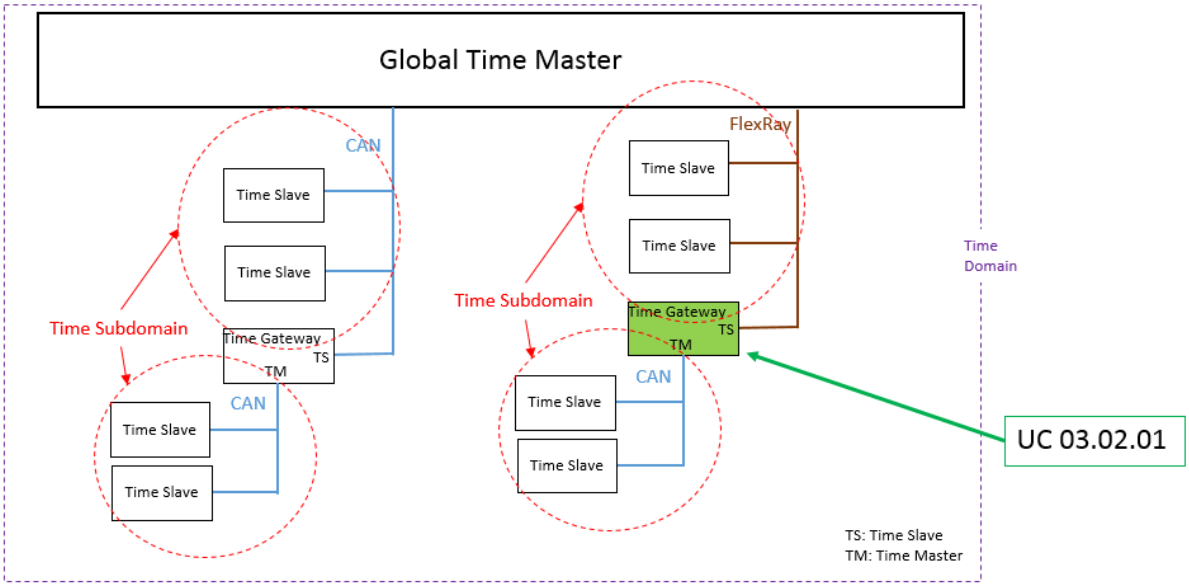


6.1.1.1.3 UC 03.02.01 Time Gateway: Time Slave over FlexRay and time master over CAN

SUT acts as Time Gateway. It receives time base, offset time base and Synchronizes Local time to Global Time base and transmits the received time base to its slaves.
Time Slave over FlexRay and Time Master on CAN

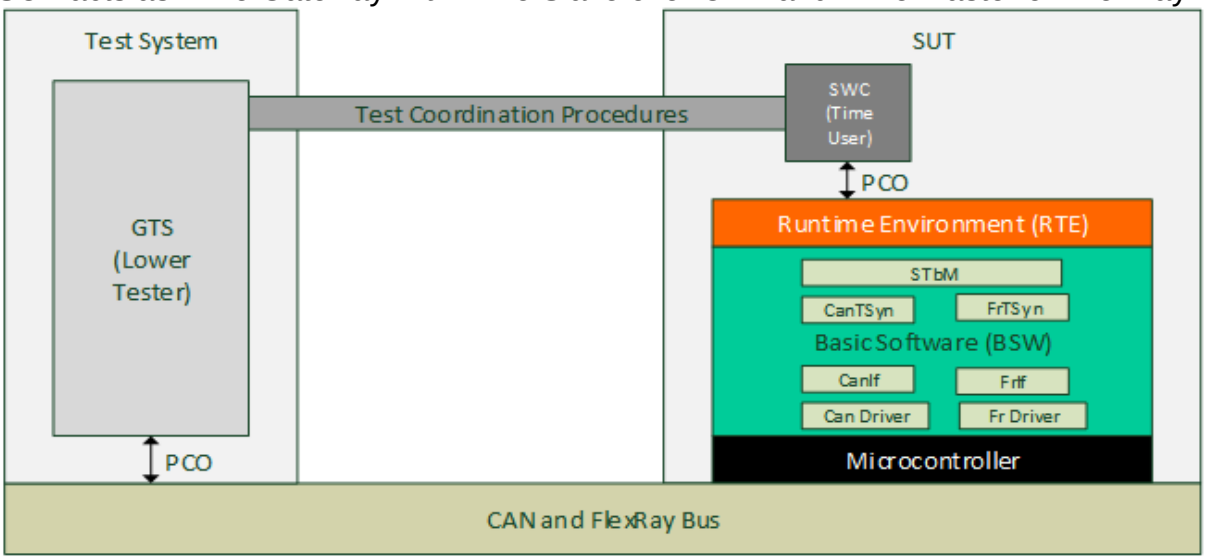


As shown in below figure, Functionalities of Time Gateway - Time Slave over FlexRay and Time Master over CAN in the use case 03.02.01

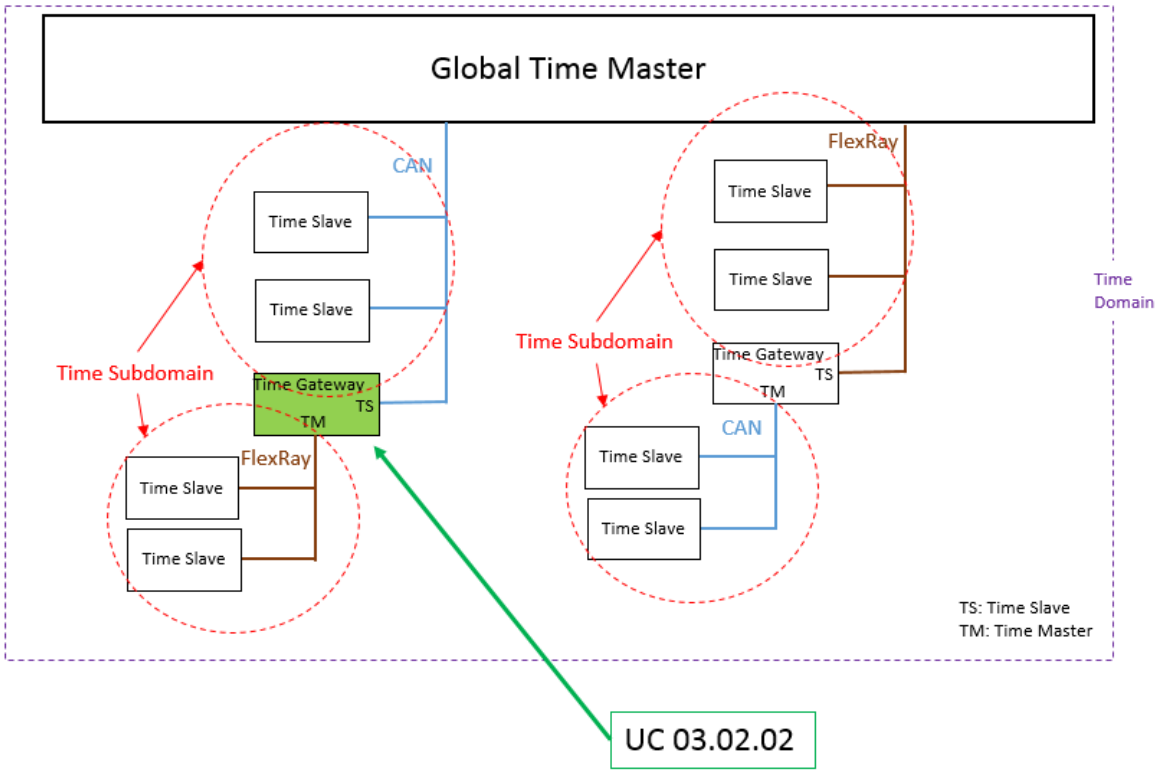


6.1.1.1.4 UC 03.02.02 Time Gateway: Time Slave over CAN and time master over FlexRay

SUT acts as Time Gateway with Time Slave over CAN and Time Master on FlexRay

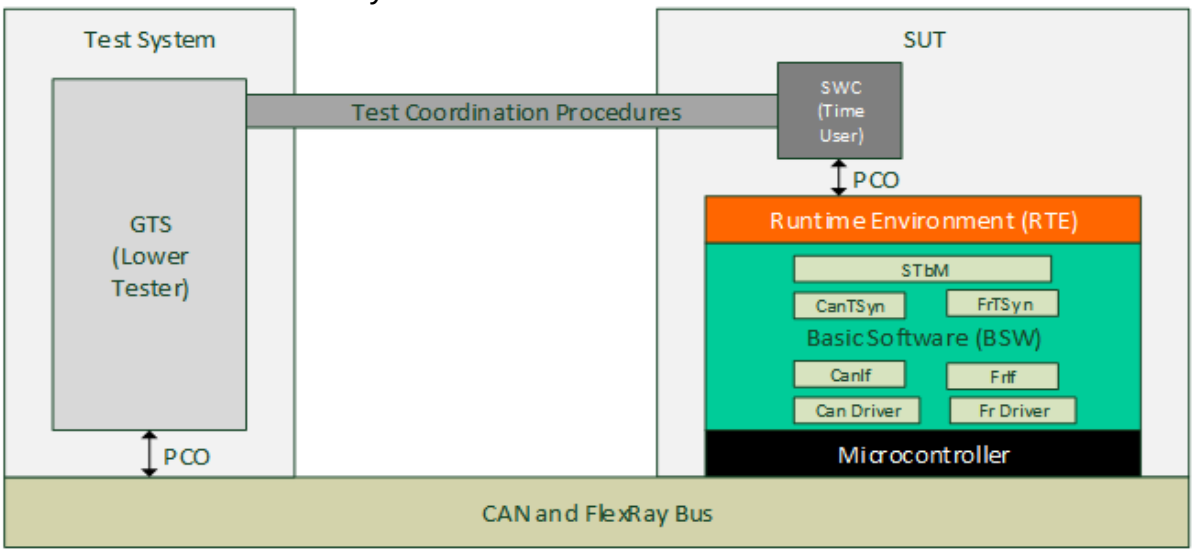


As shown in below figure, Functionalities of Time Gateway - Time Slave over CAN and Time Master over FlexRay in the use case 03.02.02

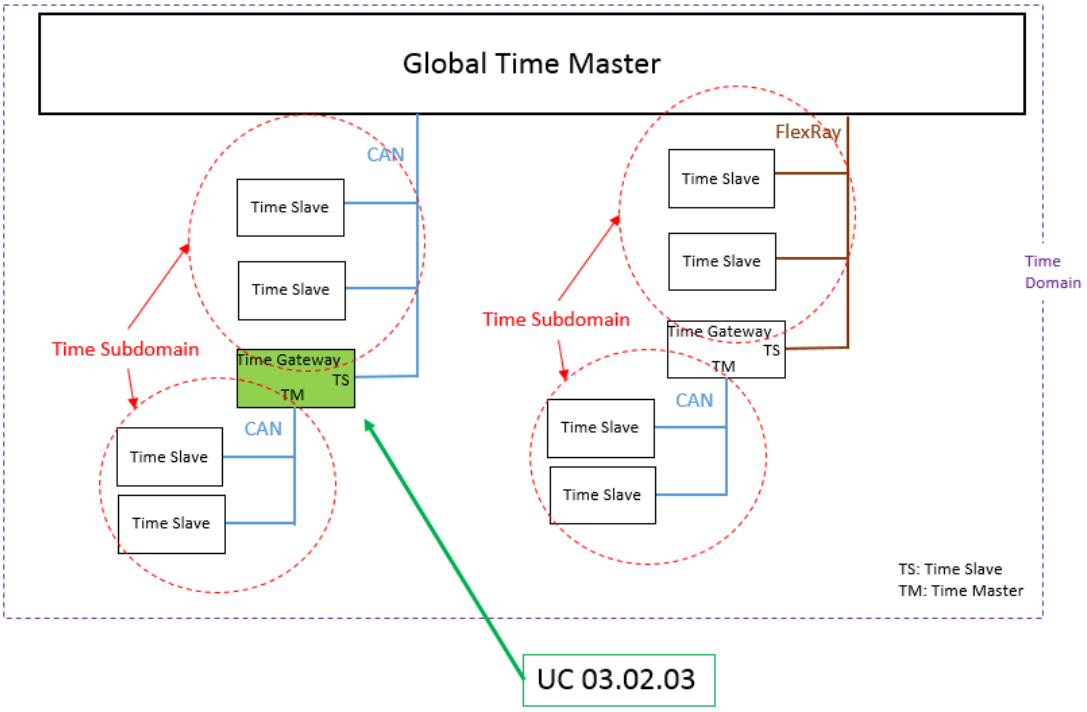


6.1.1.1.5 UC 03.02.03 Time Gateway: Time Slave over CAN (Network 1) and time master over CAN (Network 2)

SUT acts as Time Gateway with Time Slave over CAN1 and Time Master on CAN2.



As shown in below figure, Functionalities of Time Gateway - Time Slave over CAN1 and Time Master over CAN2 in the use case 03.02.03



6.1.1.2 Specific Requirements

Not Applicable.

6.1.1.3 Test Coordination Requirements

UC 03.01, UC 03.02: Time Master over CAN and/or FlexRay

- Test System (LT <CAN>) shall read the CanTSyn CAN Frames and decode the same as per Frame Format provided in AUTOSAR_SWS_TimeSyncOverCAN.
- Test System (LT <FlexRay>) shall read the FrTSyn FlexRay Frames and decode the same as per Frame Format provided in AUTOSAR_SWS_TimeSyncOverFlexRay.

UC 03.01, UC 03.02: Time Slave over CAN and/or FlexRay

- Test System (LT <CAN>) shall encode the CanTSyn CAN Frames as per Frame Format provided in AUTOSAR_SWS_TimeSyncOverCAN and transmit over bus.
- Test System (LT <FlexRay>) shall encode the FrTSyn FlexRay Frames as per Frame Format provided in AUTOSAR_SWS_TimeSyncOver FlexRay and transmit over bus.

Requirements for CRC Calculation

- Test System (LT <CAN>) shall use the Crc_CalculateCRC8H2F() (Refer AUTOSAR Specification of CRC Routines AUTOSAR_SWS_CRCLibrary.pdf) to calculate the CRC of the Frame. Below are the parameters used for CRC calculation:
  The CRC start value shall be 0xFF.
  The CRC final XOR-value shall be 0xFF.
  The CRC polynomial shall be 0x2F.
  The DataIDList shall be same as provided in CanTSyn Static Configuration.

6.1.2 Test Case Design

- Global Time Master over Multiple Bus(Single Time Domain)
  The Time domain for both CAN and FlexRay will be same. And Synchronization frames shall be sent over bus as per respective TSyn module configuration
- Global Time Master over Multiple Bus(Single Time Domain)
  The Time domain for both CAN and FlexRay will be different. And Synchronization frames shall be sent over bus with respective time domain information and respective TSyn module configuration

  Below figures provides detailed info about how the time base information is validated by the tester.

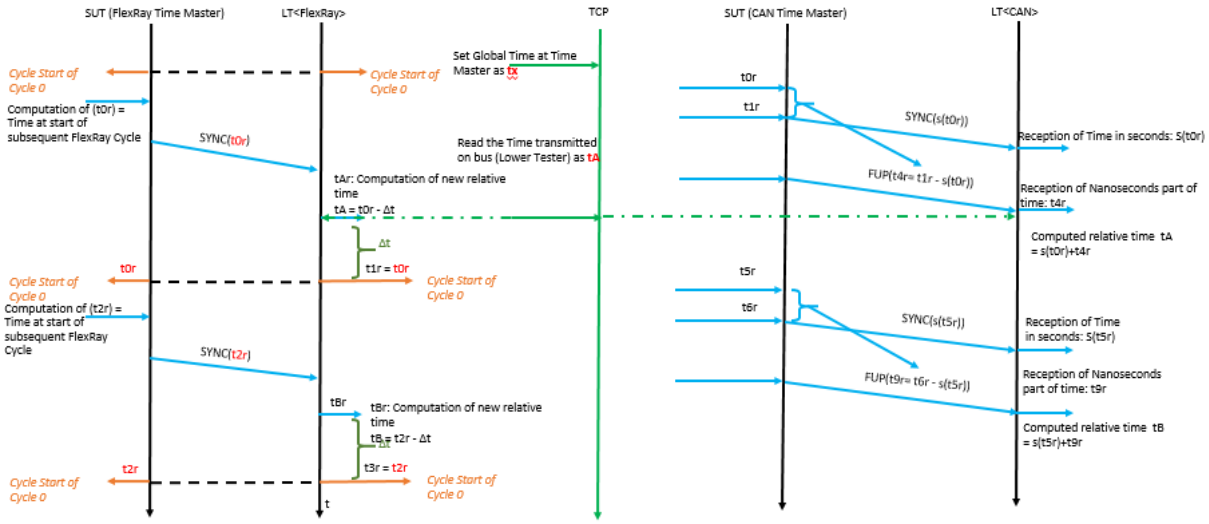


Fig: Global Time Master over Multiple Bus (Single/Multiple Time Domain) Test Design

- Time Gateway - Time Slave on FlexRay and Time Master on CAN

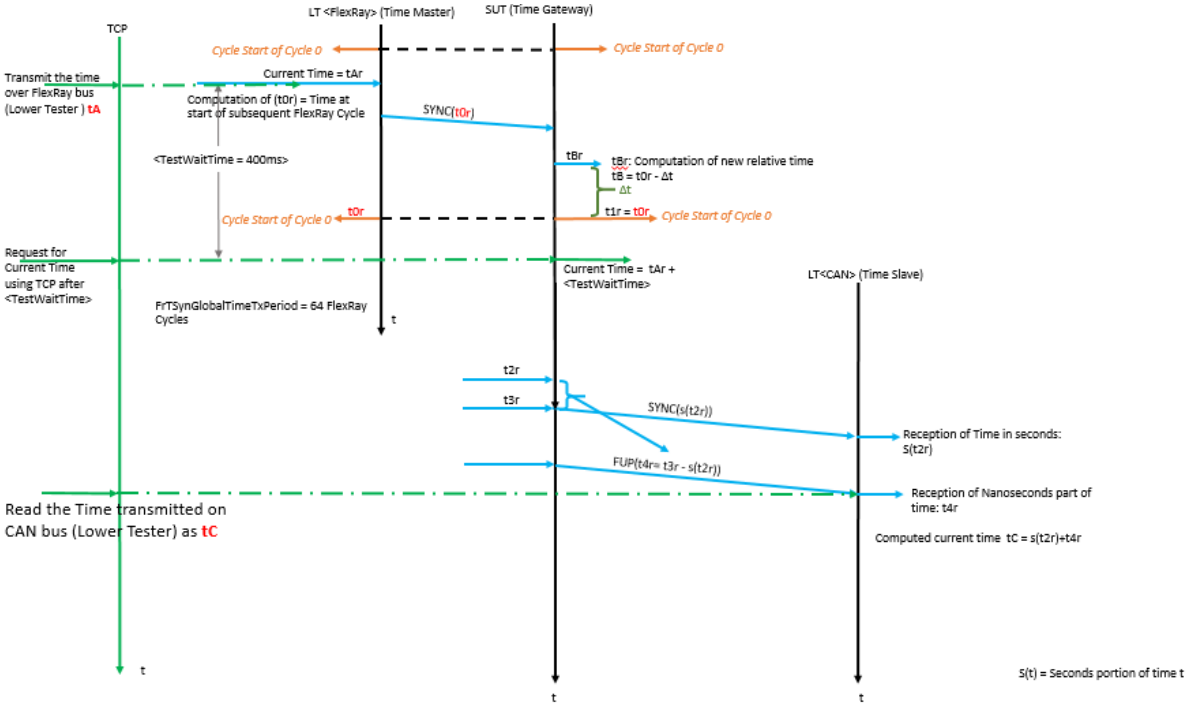


Fig: Time Gateway - Time Slave on FlexRay and Time Master on CAN Test Design

- Time Gateway - Time Slave on CAN and Time Master on FlexRay

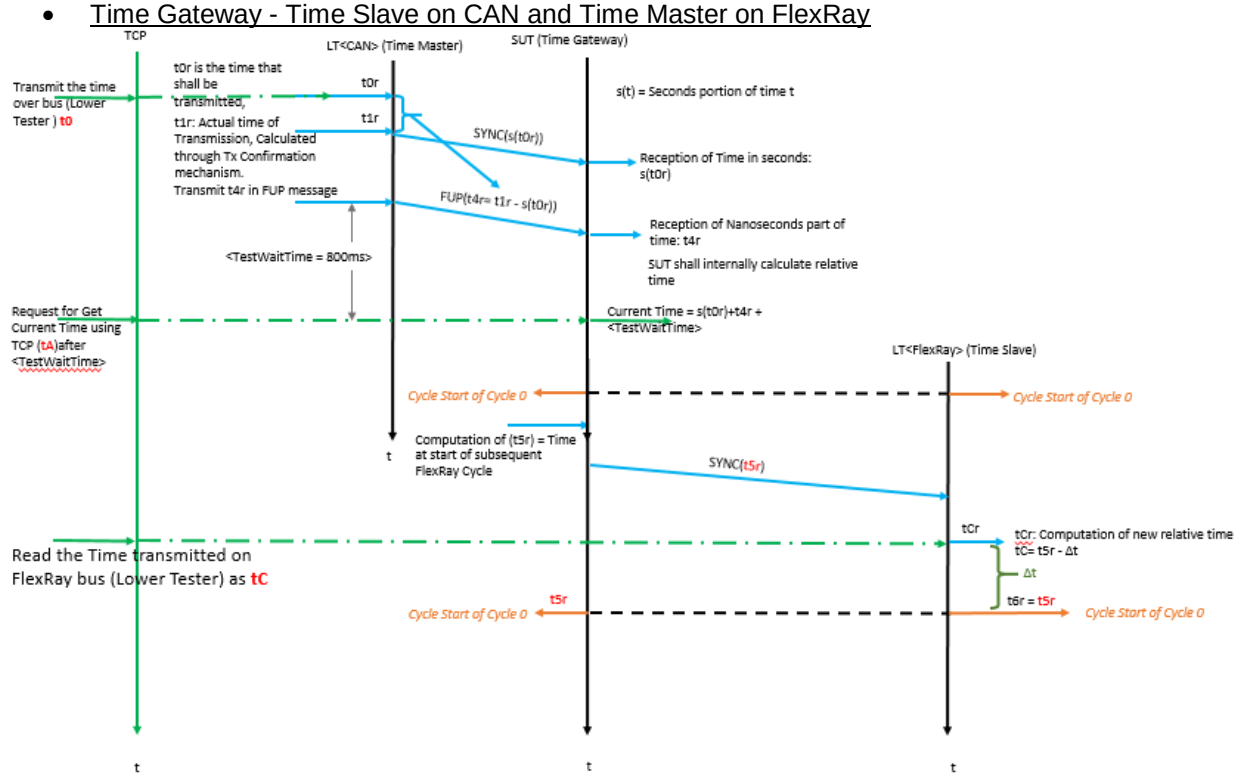


Fig: Time Gateway - Time Slave on CAN and Time Master on FlexRay Test Design

- Time Gateway - Time Slave on CAN (Network 1) and Time Master on CAN (Network 2)

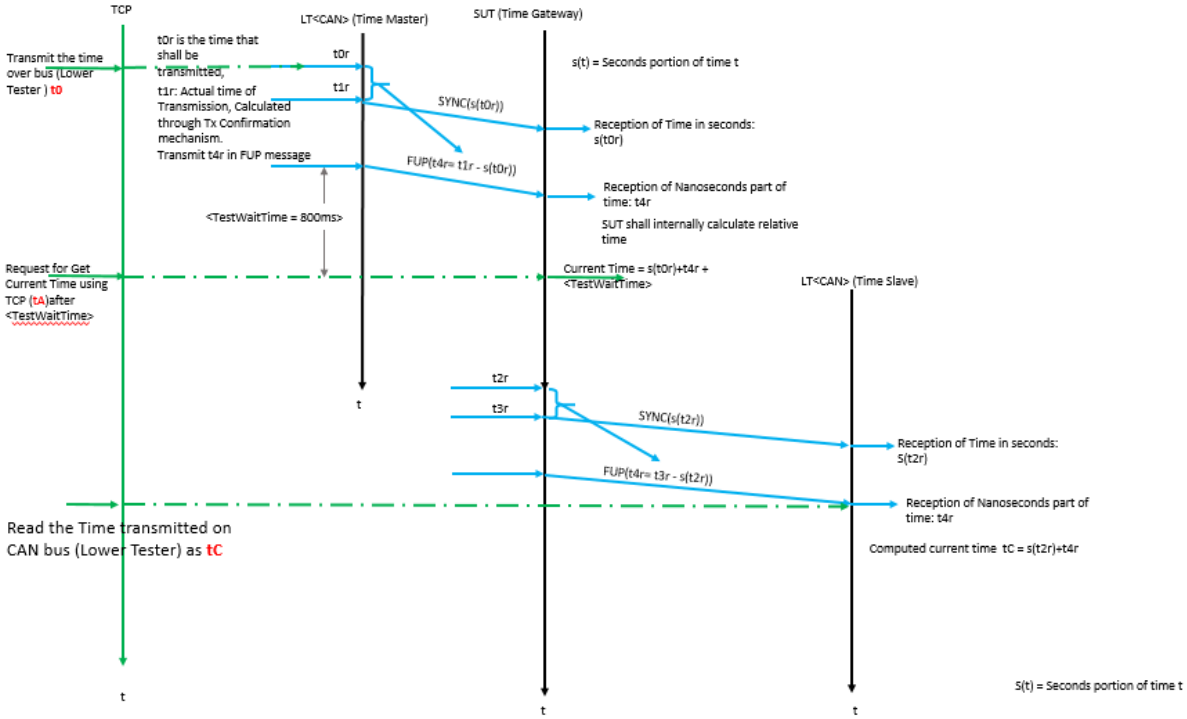


Fig: Time Gateway - Time Slave on CAN Network 1 and Time Master on CAN network 2 Test Design

6.2 Configuration requirements

The configuration can be divided into two separate parts. The *test configuration* describes variables used to parameterize the test case. The *static configuration* describes the necessary settings of the DUT in order to allow a test case to perform.

6.2.1 Test Configuration

Communication data base for CanTSyn is depicted below

test configuration parameters			
I-Pdu	CAN ID	Tx ECU	Rx ECU
AT_101_Ipdu	101	SUT	Test Bench
AT_102_Ipdu	102	Test Bench	SUT

Communication data base for FrTSyn is depicted below

test configuration parameters	
Parameter	Value
FrIf Tx Pdu	AT_201_Ipdu201
FrIfTxPdu. FrIfImmediate	FALSE
FrIfFrameTriggering.FrIfBaseCycle	0
FrIfFrameTriggering.FrIfCycleRepetition	64
FrIfFrameTriggering.FrIfSlotId	2
FrIf Rx Pdu	AT_202_Ipdu202

6.2.2 Static Configuration

6.2.2.1 Static Configuration Groups

SCG_ATS_GlobalTimeSync_Single and Multiple time Domain	
System Description Parameters	
StbM	
SystemTemplate::GlobalTime::GlobalTimeMaster.isSystemWideGlobalTimeMaster	TRUE
CanTSyn	
SystemTemplate::GlobalTime::GlobalTimeMaster.syncPeriod	2000ms
SystemTemplate::GlobalTime::CAN::GlobalTimeCanMaster.syncConfirmationTimeout	80ms
SystemTemplate::GlobalTime::GlobalTimeDomain.globalTimePdu	Ref. To PDU
FrTSyn	
SystemTemplate::GlobalTime::GlobalTimeDomain.domainId	1
SystemTemplate::GlobalTime::GlobalTimeMaster.syncPeriod	2000ms
SystemTemplate::GlobalTime::GlobalTimeDomain.subDomain	Ref. To PDU
ECU Configuration Parameters	
StbM	
StbM.StbMGeneral.StbMMainFunctionPeriod	5ms
StbM.StbMSynchronizedTimeBase.StbMLocalTimeRef	Ref. to OSCounter
CanTSyn	
CanTSyn.CanTSynGeneral.CanTSynMainFunctionPeriod	5ms
CanTSyn.CanTSynGlobalTimeDomain. CanTSynGlobalTimeDomainId	1
CanTSyn.CanTSynGlobalTimeDomain. CanTSynSynchronizedTimeBaseRef	Ref. to StbM time base
CanTSyn.CanTSynGlobalTimeDomain. CanTSynGlobalTimeMaster. CanTSynGlobalTimeTxFollowUpOffset	100ms
CanTSyn.CanTSynGlobalTimeDomain. CanTSynGlobalTimeMaster. CanTSynGlobalTimeMasterPdu. CanTSynGlobalTimeMasterConfirmationHandleId	1
FrTSyn	
FrTSyn.FrTSynGeneral.FrTSynMainFunctionPeriod	5ms
FrTSyn.FrTSynGlobalTimeDomain. FrTSynSynchronizedTimeBaseRef	Ref. to StbM time base
FrTSyn.CanTSynGlobalTimeDomain.FrTSynGlobalTimeMaster. FrTSynGlobalTimeMasterPdu.	0
Test Cases	
UC 03.01	

SCG_ATS_GlobalTimeSync_Time Gateway	
System Description Parameters	
StbM	
SystemTemplate::GlobalTime::GlobalTimeMaster.isSystemWideGlobalTimeMaster	FALSE
CanTSyn	
SystemTemplate::GlobalTime::GlobalTimeDomain.globalTimePdu	Ref. To PDU
FrTSyn	
SystemTemplate::GlobalTime::GlobalTimeDomain.subDomain	Ref. To PDU
ECU Configuration Parameters	
StbM	
StbM.StbMGeneral.StbMMainFunctionPeriod	5ms
StbM.StbMSynchronizedTimeBase.StbMLocalTimeRef	Ref. to OSCounter
CanTSyn	
CanTSyn.CanTSynGeneral.CanTSynMainFunctionPeriod	5ms
CanTSyn.CanTSynGlobalTimeDomain. CanTSynGlobalTimeMaster.CanTSynGlobalTimeMasterPdu. CanTSynGlobalTimeMasterConfirmationHandleId	0
FrTSyn	
FrTSyn.FrTSynGeneral.FrTSynMainFunctionPeriod	5ms
FrTSyn.CanTSynGlobalTimeDomain.FrTSynGlobalTimeMaster. FrTSynGlobalTimeMasterPdu. FrTSynGlobalTimeMasterConfirmationHandleId	0
Test Cases	
UC 03.02	

6.2.2.2 Required System Description

Refer Section 5.2.2.1

6.2.2.3 Required ECU Configuration

Refer Section 5.2.2.1

6.2.2.4 Required Software Components

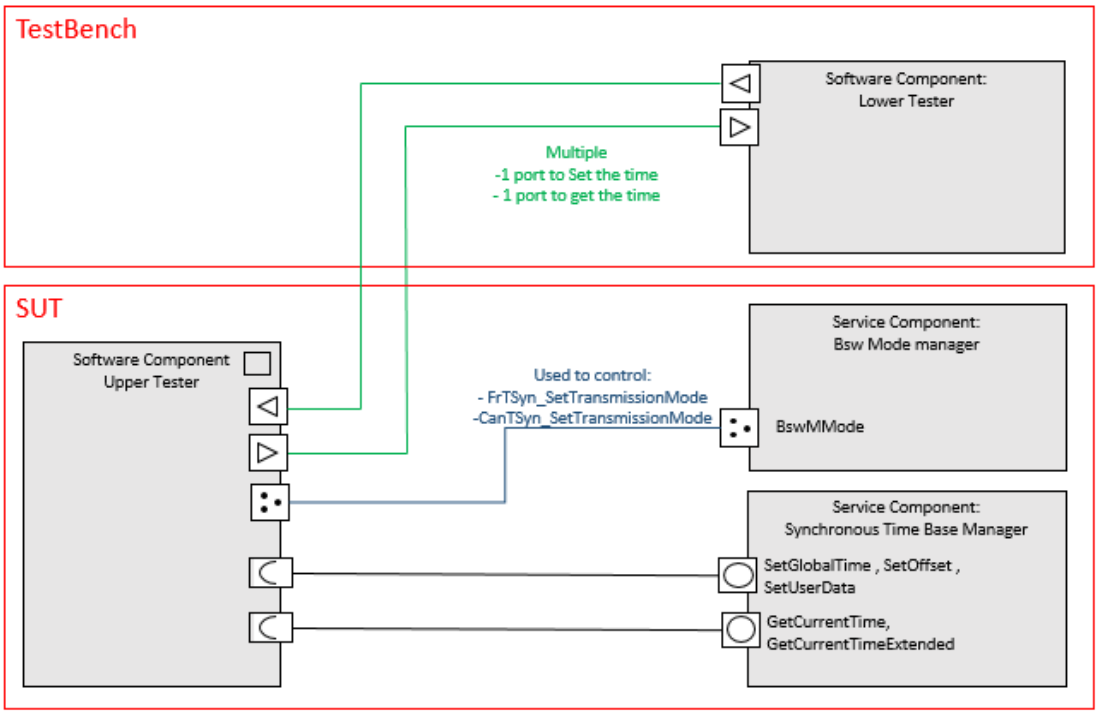


Fig: UC 03.01.01 - Global Time Master over Multiple Bus(Single Time Domain) SWC Overview

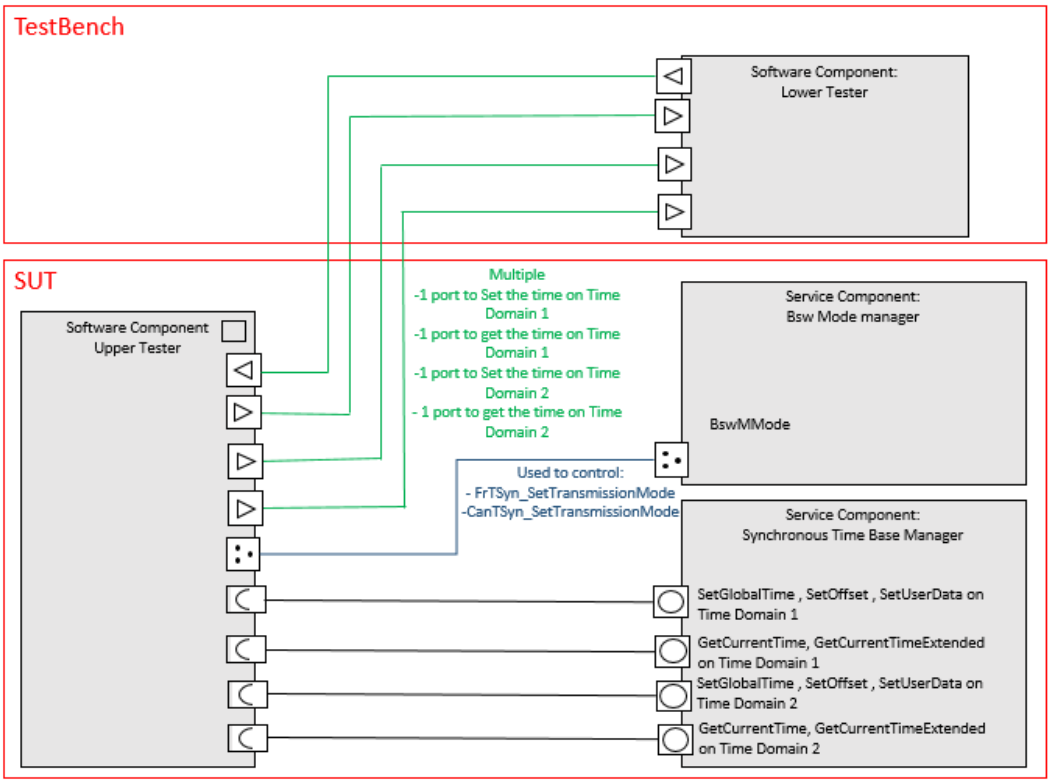


Fig: UC 03.01.02 - Global Time Master over Multiple Bus (Multiple Time Domain) SWC Overview

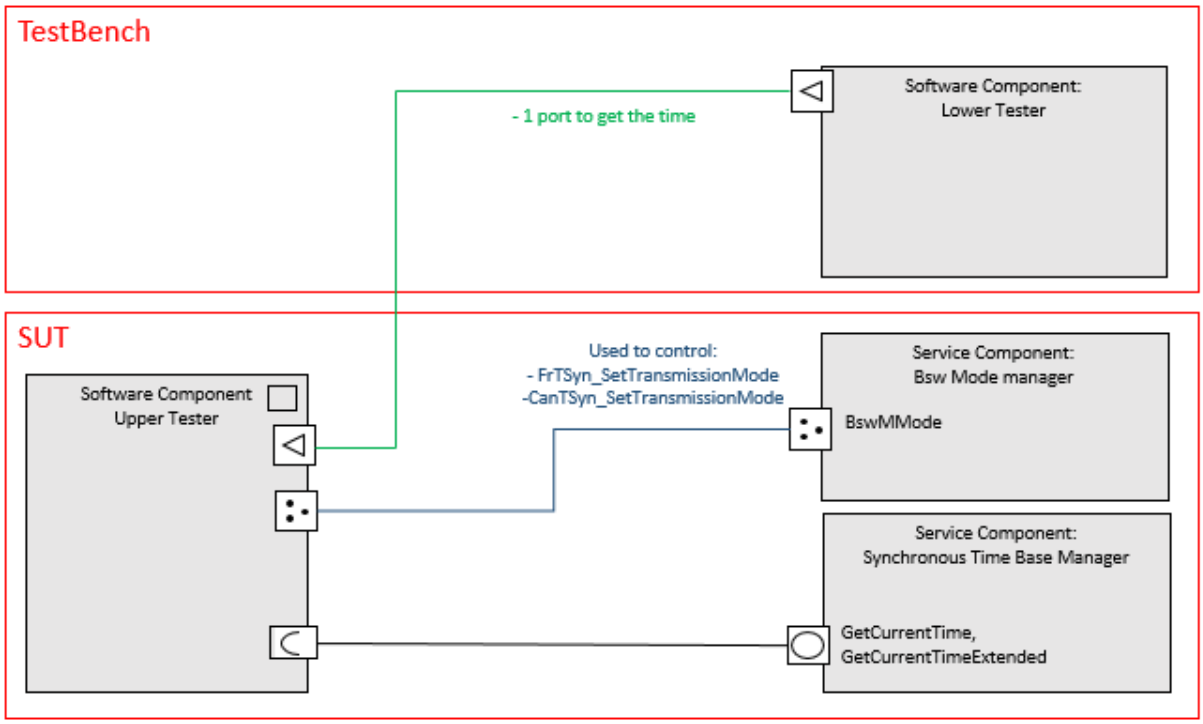


Fig: UC 03.02.01, UC 03.02.02 and UC 03.02.03 – Time Gateway SWC Overview

6.2.2.4.1 SWC Client GlobalTime_Provider

SWC Name	GlobalTime_Provider	
PORTS	**Name**	Client_SetGlobalTime
	Type	RPortPrototype
	Interface	GlobalTime_Master_Interface
	Requirements	
	Name	Client_SetUserData
	Type	RPortPrototype
	Interface	GlobalTime_Master_Interface
	Requirements	
	Name	Client_SetOffset
	Type	RPortPrototype
	Interface	GlobalTime_Master_Interface
	Requirements	

RUNNABLE ENTITIES	**Name**	RUN_GlobalTimeProvider	
	Requirements	Runnable shall be invoked by TCP	
	ServerCallPoint	**Name**	sscp_GlobalTimeProvider
		Type	SynchronousServerCallPoint
		Access to	Client_SetGlobalTime (Write operation) Client_SetUserData (Write operation) Client_SetOffset (Write operation)
		Requirements	

6.2.2.4.2 SWC Client Time_User

SWC Name	Time_User		
PORTS	**Name**	Client_GetCurrentTime	
	Type	RPortPrototype	
	Interface	GlobalTime_Slave_Interface	
	Requirements		
	Name	Client_GetCurrentTimeExtended	
	Type	RPortPrototype	
	Interface	GlobalTime_Slave_Interface	
	Requirements		
RUNNABLE ENTITIES	**Name**	RUN_TimeUser	
	Requirements	Runnable shall be invoked by TCP	
	ServerCallPoint	**Name**	sscp_TimeUser
		Type	SynchronousServerCallPoint
		Access to	Client_GetCurrentTime (Read operation) Client_GetCurrentTimeExtended

			(Read operation)
		Requirements	

6.2.2.4.3 SWC Server StbM

SWC Name	StbM	
PORTS	**Name**	Server_SetGlobalTime
	Type	PPortPrototype
	Interface	GlobalTime_Master_Interface
	Requirements	
	Name	Server_SetOffset
	Type	PPortPrototype
	Interface	GlobalTime_Master_Interface
	Requirements	
	Name	Server_SetUserData
	Type	PPortPrototype
	Interface	GlobalTime_Master_Interface
	Requirements	
	Name	Server_GetCurrentTime
	Type	PPortPrototype
	Interface	GlobalTime_Slave_Interface
	Requirements	
	Name	Server_GetCurrentTimeExtended
	Type	PPortPrototype
	Interface	GlobalTime_Slave_Interface
	Requirements	
	Name	StbM_SetGlobalTime
	Requirements	

RUNNABLE ENTITIES	**Started by Event**	**Name**	OIE_ SetGlobalTime
		Type	OperationInvokedEvent Port:Server_SetGlobalTime Operation: Read
		Requirements	
	Name	StbM_SetOffset	
	Requirements		
	ServerCallPoint	**Name**	OIE_SetOffset
		Type	OperationInvokedEvent Port: Server_SetOffset Operation: Read
		Requirements	
	Name	StbM_SetUserData	
	Requirements		
	Started by Event	**Name**	OIE_ SetUserData
		Type	OperationInvokedEvent Port: Server_SetUserData Operation: Read
		Requirements	
	Name	StbM_GetCurrentTime	
	Requirements		
	Started by Event	**Name**	OIE_ GetCurrentTime
		Type	OperationInvokedEvent Port: Server_GetCurrentTime Operation: Read
		Requirements	
	Name	StbM_GetCurrentTimeExtended	
	Requirements		

	Started by Event	**Name**	OIE_GetCurrentTimeExtended
		Type	OperationInvokedEvent Port: Server_GetCurrentTimeExtended Operation: Read
		Requirements	

6.3 Re-usable Test Steps

Not Applicable

6.4 Test Cases

6.4.1 [ATS_GTS_01281] Global Time Master over Multiple Bus(Single Time Domain): Setting of Global Time base and user data and sending of SYNC frame over CAN and FlexRay

Test Objective	Global Time Master over Multiple Bus(Single Time Domain): Setting of Global Time base and user data and sending of SYNC frame over CAN and FlexRay		
ID	ATS_GTS_01281	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, CanTSyn, FrTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00133 ATR: ATR_ATR_00134		
Trace to SWS Item	TimeSyncOverCAN: SWS_CanTSyn_00028 TimeSyncOverCAN: SWS_CanTSyn_00030 TimeSyncOverFlexRay: SWS_FrTSyn_00019 TimeSyncOverFlexRay: SWS_FrTSyn_00020		
Requirements / Reference to Test Environment	Use Case UC03.01		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 1 CanTSyn: CanTSynGlobalTimeTxCrcSecured= CRC_NOT_SUPPORTED FrTSyn: FrTSynGlobalTimeTxCrcSecured = CRC_NOT_SUPPORTED		
Summary	Aim is to test the functionality of global time master a time domain and Transmission of Synchronization message over FlexRay and CAN Bus. Verify that StbM accepts the global time base from Upper Tester using client-server		

	Interface. Verify that CanTSyn shall transmit the global time base to time slave periodically via SYNC and FUP message. Verify that FrTSyn shall transmit the global time base to time slave periodically via SYNC message.	
Needed Adaptation to other Releases	Not Applicable.	
Pre-conditions	SUT shall be initialized.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** Start RUN_ GlobalTimeProvider.	
Step 2	**[RUN<RUN_ GlobalTimeProvider>]** Execute Rte_Call_ Client_SetGlobalTime and Rte_Call_ Client_SetUserData with below values: timeBaseId = 1 StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB StbM_UserDataType.User Data Byte 2 = 0xCC	**[RUN<RUN_ GlobalTimeProvider>]** Rte_Call returns RTE_E_OK.
Step 3	**[CP]** Wait 100ms.	
Step 4	**[CP]** Start RUN_TimeUser.	
Step 5	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime.	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time received from Time user shall be

		as mentioned below: StbM_TimeStampType.nanoseconds = 0x00000000 + <TestWaitTime = 100ms> StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB StbM_UserDataType.User Data Byte 2 = 0xCC
Step 6	**[LT<CAN>]** Receives the SYNC message.	**[LT]** Receives frame format as mentioned below: Byte 0: Type = 0x10 Byte 1: User Byte 1 = 0xBB Byte 2: D = 0x1 SC = 0x0 Byte 3: User Byte 0 = 0xAA Byte 4-7: SyncTimeSec = StbM_TimeStampType.seconds
Step 7	**[LT<CAN>]** Receives the FUP message	**[LT]** Receives frame format mentioned below: Byte 0: Type = 0x18 Byte 1: User Byte 2 = 0xCC Byte 2: D = 0x1 SC = 0x0 Byte 3: reserved (Bit 7 to Bit 3) = 0 SGW (Bit 2) = 0 OVS = Overflow of seconds (Bit 1 to Bit

		0) Byte 4-7: SyncTimeNSec = StbM_TimeStampType.nanoseconds + <Test Wait Time = 100ms>
Step 8	**[LT<CAN>]** Receives Global time base. Store the time as base for next periodic message processing	**[LT]** Get the time stamp values as below: StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 9	**[LT<CAN>]** Receives the next periodic SYNC and FUP message	**[LT]** Time stamp values shall be as mentioned below: StbM_TimeStampType.nanoseconds = tB.Nano StbM_TimeStampType.seconds = tB.Sec StbM_TimeStampType.secondsHi = tB.SecHi StbM_UserDataType.User Data Byte 0 = 0xAA The difference between two time base shall be tB – tA = CanTSynGlobalTimeTxPeriod (2 second) + CanTSynGlobalTimeTxFollowUpOffset.
Step 10	**[LT<FLEXRAY>]** Receives the SYNC message	**[LT]** Receives frame format as mentioned below: Byte 0: Type = 0x10 Byte 1: UserByte 2 = 0xCC Byte 2: D = 0x1 SC = 0x0 Byte 3: FCNT= FlexRay Cycle Counter

		0 (Bit 7 to Bit 2) SGW (Bit 1) SyncToGTM = 0 reserved (Bit 0), default: 0 Byte 4: User Byte 0 = 0xAA Byte 5: User Byte 1 = 0xBB Byte 6-11: SyncTimeSec = 48 Bit time stamp in seconds Byte 12-15: SyncTimeNSec = 32 Bit time stamp in nanoseconds
Step 11	**[LT<FLEXRAY>]** Receives Global time base. Store the time as base for next periodic message processing	**[LT]** Get the time stamp values as below: StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 12	**[LT<FLEXRAY>]** Receives the next periodic SYNC message	**[LT]** Time stamp values shall be as mentioned below: StbM_TimeStampType.nanoseconds = tB.Nano StbM_TimeStampType.seconds = tB.Sec StbM_TimeStampType.secondsHi = tB.SecHi StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB StbM_UserDataType.User Data Byte 2 = 0xCC The difference between two time base shall be

		tB – tA = FrTSynGlobalTimeTxPeriod (320ms).
Post-conditions	None	

6.4.2 [ATS_GTS_01257] Global Time Master over Multiple Bus(MultipleTime Domain): Setting of Global Time base and user data and sending of SYNC frame over CAN and FlexRay

Test Objective	Global Time Master over Multiple Bus(MultipleTime Domain): Setting of Global Time base and user data and sending of SYNC frame over CAN and FlexRay		
ID	ATS_GTS_01257	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, CanTSyn, FrTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00133 ATR: ATR_ATR_00134		
Trace to SWS Item	TimeSyncOverCAN: SWS_CanTSyn_00038 TimeSyncOverFlexRay: SWS_FrTSyn_00023		
Requirements / Reference to Test Environment	Use Case UC03.01		
Configuration Parameters	StbM: StbMSynchronizedTimeBaseIdentifier = 1, 2 CanTSyn: CanTSynGlobalTimeTxCrcSecured= CRC_NOT_SUPPORTED FrTSyn: FrTSynGlobalTimeTxCrcSecured = CRC_NOT_SUPPORTED		
Summary	Aim is to test the functionality of global time master on Multiple Time Domain and Transmission of Synchronization Messages over Multiple Bus Verify that StbM accepts the global time base from Upper Tester using client-server Interface for Time Domain 1 and Time Domain 2 Verify that CanTSyn as Global time master shall transmit the Global time base to time slaves periodically via SYNC and FUP message. Verify that FrTSyn as Global time master shall transmit the Global Time Base to time slave periodically via SYNC message.		
Needed Adaptation to other Releases	Not Applicable.		
Pre-conditions	SUT shall be initialized.		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** Start RUN_ GlobalTimeProvider.		

Step 2	**[RUN<RUN_ GlobalTimeProvider>]** Execute Rte_Call_ Client_SetGlobalTime and Rte_Call_ Client_SetUserData with below values: timeBaseId = 1 StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB	**[RUN<RUN_ GlobalTimeProvider>]** Rte_Call returns RTE_E_OK.
Step 3	**[CP]** Wait 100ms.	
Step 4	**[CP]** Start RUN_TimeUser.	
Step 5	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime.	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time received from Time user shall be as mentioned below: StbM_TimeStampType.nanoseconds = 0x00000000 + <TestWaitTime = 100ms> StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB
Step 6	**[LT<CAN>]** Receives the SYNC message	**[LT]** Receives frame format as mentioned below:

		Byte 0: Type = 0x10 Byte 1: User Byte 1 = 0xBB Byte 2: D = 0x1 SC = 0x0 Byte 3: User Byte 0 = 0xAA Byte 4-7: SyncTimeSec = StbM_TimeStampType.seconds
Step 7	**[LT<CAN>]** Receives the FUP message	**[LT]** Receives frame format mentioned below: Byte 0: Type = 0x18 Byte 1: User Byte 2 = 0x00 Byte 2: D = 0x1 SC = 0x0 Byte 3: reserved (Bit 7 to Bit 3) = 0 SGW (Bit 2) = 0 OVS = Overflow of seconds (Bit 1 to Bit 0) Byte 4-7: SyncTimeNSec = StbM_TimeStampType.nanoseconds + <Test Wait Time = 100ms>
Step 8	**[LT<CAN>]** Receives Global time base. Store the time as base for next periodic message processing	**[LT]** Get the time stamp values as below: StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 9	**[LT<CAN>]** Receives the next periodic SYNC and FUP message	**[LT]** Time stamp values shall be as mentioned below: StbM_TimeStampType.nanoseconds = tB.Nano

		StbM_TimeStampType.seconds = tB.Sec StbM_TimeStampType.secondsHi = tB.SecHi StbM_UserDataType.User Data Byte 0 = 0xAA The difference between two time base shall be tB – tA = CanTSynGlobalTimeTxPeriod (2 second) + CanTSynGlobalTimeTxFollowUpOffset.
Step 10	**[CP]** Start RUN_ GlobalTimeProvider.	
Step 11	**[RUN<RUN_ GlobalTimeProvider>]** Execute Rte_Call_ Client_SetGlobalTime and Rte_Call_ Client_SetUserData with below values: timeBaseId = 2 StbM_TimeStampType.nanoseconds = 0x00000000 StbM_TimeStampType.seconds = 0x00001C20 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB StbM_UserDataType.User Data Byte 2 = 0xCC	**[RUN<RUN_ GlobalTimeProvider>]** Rte_Call returns RTE_E_OK.
Step 12	**[CP]** Wait 100ms.	
Step 13	**[CP]** Start RUN_TimeUser.	
Step 14	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime for timeBaseId = 2	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time received from Time user shall be

		as mentioned below: StbM_TimeStampType.nanoseconds = 0x00000000 + <TestWaitTime = 100ms> StbM_TimeStampType.seconds = 0x00001C20 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB StbM_UserDataType.User Data Byte 2 = 0xCC
Step 15	**[LT<FLEXRAY>]** Receives the SYNC message	**[LT]** Receives frame format as mentioned below: Byte 0: Type = 0x10 Byte 1: UserByte 2 = 0xCC Byte 2: D = 0x1 SC = 0x0 Byte 3: FCNT= FlexRay Cycle Counter 0 (Bit 7 to Bit 2) SGW (Bit 1) SyncToGTM = 0 reserved (Bit 0), default: 0 Byte 4: User Byte 0 = 0xAA Byte 5: User Byte 1 = 0xBB Byte 6-11: SyncTimeSec = 48 Bit time stamp in seconds Byte 12-15: SyncTimeNSec = 32 Bit time stamp in nanoseconds + <Test Wait Time = 100ms>
Step 16	**[LT<FLEXRAY>]**	**[LT]**

	Receives Global time base. Store the time as base for next periodic message processing	Get the time stamp values as below: StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 17	**[LT<FLEXRAY>]** Receives the next periodic SYNC message	**[LT]** Time stamp values shall be as mentioned below: StbM_TimeStampType.nanoseconds = tB.Nano StbM_TimeStampType.seconds = tB.Sec StbM_TimeStampType.secondsHi = tB.SecHi StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB StbM_UserDataType.User Data Byte 2 = 0xCC The difference between two time base shall be tB – tA = FrTSynGlobalTimeTxPeriod (320ms).
Post-conditions	None	

6.4.3 [ATS_GTS_01258] Time Gateway- Time Slave on FlexRay and Time Master on CAN

Test Objective	Time Gateway- Time Slave on FlexRay and Time Master on CAN		
ID	ATS_GTS_01258	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, CanTSyn, FrTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00133 ATR: ATR_ATR_00134		

Trace to SWS Item	TimeSyncOverCAN: SWS_CanTSyn_00028 TimeSyncOverCAN: SWS_CanTSyn_00030 TimeSyncOverFlexRay: SWS_FrTSyn_00046
Requirements / Reference to Test Environment	Use Case UC03.02
Configuration Parameters	StbM: StbMIsSystemWideGlobalTimeMaster = FALSE StbMSynchronizedTimeBaseIdentifier = 1 CanTSyn: CanTSynGlobalTimeDomainId = 1 CanTSynSynchronizedTimeBaseRef = Reference to StbMSynchronizedTimeBase CanTSynGlobalTimeTxCrcSecured= CRC_NOT_SUPPORTED CanTSynGlobalTimeTxFollowUpOffset = 100ms CanTSynGlobalTimeTxPeriod = 2000ms CanTSynMasterConfirmationTimeout = 80ms FrTSyn: FrTSynRxCrcValidated = CRC_NOT_VALIDATED FrTSynGlobalTimeDomainId = 1
Summary	Aim is to verify the functionality of time gateway Verify that FrTSyn as time slave shall receive the global time base periodically via SYNC message. Verify that StbM synchronizes the local time as per the received global time base from FrTSyn (using API StbM_BusSetGlobalTime) Verify that CanTSyn as time master shall transmit the global time base received from global time domain to time subdomain periodically via SYNC and FUP message respectively. Verify that UT shall get the valid current time using Client-Server Interface.
Needed Adaptation to other Releases	Not Applicable.
Pre-conditions	SUT shall be initialized.

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[LT<FLEXRAY>]** Transmit SYNC message with timeBaseId = 1 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB And time value as T0 Where T0 = TSYNC + (MacroticksPerCycle *	**[SUT]** Receives SYNC message ignoring CRC value(if there) in below format Byte 0: Type = 0x10 Byte 1: User Byte 2 Byte 2: D = <Time Domain = 1> (Bit 7 to Bit 4) SC = <Sequence Counter = 0> (Bit 3 to Bit 0)

	(64 - currentCycle) - currentMacroticks) * MacrotickDuration And TSYNC value as given below: StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_TimeStampType.nanoseconds = 0x00000000	Byte 3: FCNT= <FlexRay Cycle Counter = 0> (Bit 7 to Bit 2) SGW (Bit 1) SyncToSubDomain = 1 reserved (Bit 0) Byte 4: User Byte 0 Byte 5: User Byte 1 Byte 6-11: SyncTimeSec = T0.secondsHi and T0.seconds Byte 12-15: SyncTimeNSec = T0.nanoseconds
Step 2		**[SUT]** StbM updates its time base with values StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 3	**[CP]** Wait 200ms	
Step 4	**[CP]** Start RUN_TimeUser	
Step 5	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 200ms> StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 6	**[LT<CAN>]** Receives the SYNC message	**[LT]** Receives SYNC message in the

		format mentioned below: Byte 0: Type = 0x10 Byte 1: User Byte 1 = 0xBB Byte 2: D = 0x1 SC = 0x0 Byte 3: User Byte 0 = 0xAA Byte 4-7: SyncTimeSec = StbM_TimeStampType.seconds
Step 7	**[LT<CAN>]** Receives the FUP message	**[LT]** Receives FUP message in the format mentioned below: Byte 0: Type = 0x18 Byte 1: User Byte 2 = 0x00 Byte 2: D = 0x1 SC = 0x0 Byte 3: reserved (Bit 7 to Bit 3) = 0 SGW (Bit 2) = 0 OVS = Overflow of seconds (Bit 1 to Bit 0) Byte 4-7: SyncTimeNSec = StbM_TimeStampType.nanoseconds
Step 8	**[LT<CAN>]** Receives Global time base. Store the time as base for next periodic message processing	**[LT]** Get the time stamp values as below: StbM_TimeStampType.nanoseconds = tC.Nano StbM_TimeStampType.seconds = tC .Sec StbM_TimeStampType.secondsHi = tC .SecHi Time Received by LT<CAN> should be tC (Time base transmitted by LT<FlexRay> + ToleranceTime_FrTSyn + ToleranceTime_CanTSyn)

		ToleranceTime_FrTSyn = Max one main function of FrTSyn as reception is asynchronous ToleranceTime_CanTSyn = Max one main function of CanTSyn as gateway is asynchronous.
Post-conditions	None	

6.4.4 [ATS_GTS_01259] Time Gateway - Time Slave on CAN and Time Master on FlexRay

Test Objective	Time Gateway - Time Slave on CAN and Time Master on FlexRay		
ID	ATS_GTS_01259	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, CanTSyn, FrTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00133 ATR: ATR_ATR_00134		
Trace to SWS Item	SynchronizedTimeBaseManager: SWS_StbM_00184 TimeSyncOverCAN: SWS_CanTSyn_00064 TimeSyncOverCAN: SWS_CanTSyn_00072 TimeSyncOverFlexRay: SWS_FrTSyn_00023		
Requirements / Reference to Test Environment	Use Case UC03.02		
Configuration Parameters	StbM: StbMIsSystemWideGlobalTimeMaster = FALSE StbMSynchronizedTimeBaseIdentifier = 1 CanTSyn: CanTSynGlobalTimeDomainId = 1 CanTSynGlobalTimeFollowUpTimeout = 300ms CanTSynGlobalTimeSequenceCounterJumpWidth = 2 CanTSynSynchronizedTimeBaseRef = Reference to StbMSynchronizedTimeBase CanTSynRxCrcValidated = CRC_IGNORED CanTSynGlobalTimeSlaveHandleId = 0 FrTSyn: FrTSynGlobalTimeDomainId = 1 FrTSynGlobalTimeSequenceCounterJumpWidth = 2 FrTSynSynchronizedTimeBaseRef = Reference to StbMSynchronizedTimeBase FrTSynGlobalTimeTxCrcSecured = CRC_NOT_SUPPORTED FrTSynGlobalTimeTxPeriod = 320ms FrTSynGlobalTimePduRef Reference to Pdu		
Summary	Aim is to verify the functionality of time gateway Verify that CanTSyn as time slave shall receive the global offset time periodically via SYNCand FUP message.		

	Verify that StbM synchronizes the offset time as per the received global time base from CanTSyn (using API StbM_SetOffset) Verify that FrTSyn as time master shall transmit the offset time base received from global time domain to time sub-domain periodically via OFS message respectively. Verify that UT shall get the valid current time using Client-Server Interface.	
Needed Adaptation to other Releases	Not Applicable.	
Pre-conditions	SUT shall be initialized.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[LT<CAN>]** Transmit SYNC message with timeBaseId = 1 StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 = 0xBB	**[SUT]** Receives SYNC message ignoring CRC value in below format Byte 0: Type = 0x10 Byte 1: User Byte 1 = 0xBB Byte 2: D = 0x1 SC = 0x0 Byte 3: User Byte 0 = 0xAA Byte 4-7: SyncTimeSec = StbM_TimeStampType.seconds
Step 2	**[LT<CAN>]** Transmit FUP message with StbM_TimeStampType.nanoseconds = t4r	**[SUT]** Receives FUP message in below format Byte 0: Type = 0x18 Byte 1: User Byte 2 = 0x00 Byte 2: D = 0x1 SC = 0x0 Byte 3: reserved (Bit 7 to Bit 3) = 0 SGW (Bit 2) = 0 OVS = Overflow of seconds (Bit 1 to Bit 0) Byte 4-7: SyncTimeNSec = StbM_TimeStampType.nanoseconds
Step 3		**[SUT]**

		StbM updates its time base with values StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 4	**[CP]** Wait 800ms	
Step 5	**[CP]** Start RUN_TimeUser	
Step 6	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 800ms> StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 7	**[LT<FLEXRAY>]** Receives the SYNC message without CRC validation.	**[LT]** Receives frame format as mentioned below: Byte 0: Type = 0x10 Byte 1: UserByte 2 Byte 2: D = 0x1 SC = 0x0 Byte 3: FCNT= FlexRay Cycle Counter 0 (Bit 7 to Bit 2) SGW (Bit 1) SyncToSubDomain = 1 reserved (Bit 0), default: 0

		Byte 4: User Byte 0 Byte 5: User Byte 1 Byte 6-11: SyncTimeSec = 48 Bit time stamp in seconds Byte 12-15: SyncTimeNSec = 32 Bit time stamp in nanoseconds
Step 8	**[LT<FLEXRAY>]** Receives Global time base. Store the time as base for next periodic message processing	**[LT]** Get the time stamp values as below: StbM_TimeStampType.nanoseconds = tC.Nano StbM_TimeStampType.seconds = tC .Sec StbM_TimeStampType.secondsHi = tC .SecHi Time Received by LT<FlexRay> should be tC (Time base transmitted by LT<CAN> + ToleranceTime_CanTSyn + ToleranceTime_FrTSyn) ToleranceTime_CanTSyn = Max one main function of CanTSyn as reception is asynchronous ToleranceTime_FrTSyn = Max one main function of FrTSyn as gateway is asynchronous
Post-conditions	None	

6.4.5 [ATS_GTS_01260] Time Gateway - Time Slave on CAN (Network 1) and Time Master on CAN (Network 2)

Test Objective	Time Gateway - Time Slave on CAN (Network 1) and Time Master on CAN (Network 2)		
ID	ATS_GTS_01260	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	StbM, CanTSyn	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00133		
Trace to SWS Item	SynchronizedTimeBaseManager: SWS_StbM_00184 TimeSyncOverCAN: SWS_CanTSyn_00028		

	TimeSyncOverCAN: SWS_CanTSyn_00030 TimeSyncOverCAN: SWS_CanTSyn_00064
Requirements / Reference to Test Environment	Use Case UC03.02
Configuration Parameters	StbM: StbMIsSystemWideGlobalTimeMaster = FALSE StbMSynchronizedTimeBaseIdentifier = 1 CanTSyn for Master: CanTSynGlobalTimeDomainId = 1 CanTSynGlobalTimeSequenceCounterJumpWidth = 2 CanTSynSynchronizedTimeBaseRef = Reference to StbMSynchronizedTimeBase CanTSynGlobalTimeTxCrcSecured= CRC_NOT_SUPPORTED CanTSynGlobalTimeTxFollowUpOffset = 100ms CanTSynGlobalTimeTxPeriod = 2000ms CanTSynMasterConfirmationTimeout = 80ms CanTSyn for slave: CanTSynGlobalTimeFollowUpTimeout = 300ms CanTSynRxCrcValidated = CRC_IGNORED
Summary	Aim is to verify the functionality of time gateway Verify that CanTSyn as time slave(CAN 1) shall receive the global time base periodically via SYNC and FUP message. Verify that StbM synchronizes the local time as per the received global time base from CanTSyn (using API StbM_BusSetGlobalTime) Verify that CanTSyn as time master(CAN 2) shall transmit the global time base received from global time domain to time sub-domain periodically via SYNC and FUP message. Verify that UT shall get the valid current time using Client-Server Interface.
Needed Adaptation to other Releases	Not Applicable.
Pre-conditions	SUT shall be initialized.

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[LT<CAN>]** Transmit SYNC message with timeBaseId = 1 StbM_TimeStampType.seconds = 0x00000E10 StbM_TimeStampType.secondsHi = 0x0000 StbM_UserDataType.User Data Byte 0 = 0xAA StbM_UserDataType.User Data Byte 1 =	**[SUT]** Receives SYNC message ignoring CRC value in below format Byte 0: Type = 0x10 Byte 1: User Byte 1 = 0xBB Byte 2: D = 0x1 SC = 0x0 Byte 3: User Byte 0 = 0xAA

	0xBB	Byte 4-7: SyncTimeSec = StbM_TimeStampType.seconds
Step 2	**[LT<CAN>]** Transmit FUP message with StbM_TimeStampType.nanoseconds = t4r Byte CRC = Invalid CRC	**[SUT]** Receives FUP message ignoring CRC value in below format Byte 0: Type = 0x18 Byte 1: User Byte 2 =0x00 Byte 2: D = 0x1 SC = 0x0 Byte 3: reserved (Bit 7 to Bit 3) = 0 SGW (Bit 2) = 0 OVS = Overflow of seconds (Bit 1 to Bit 0) Byte 4-7: SyncTimeNSec = StbM_TimeStampType.nanoseconds
Step 3		**[SUT]** StbM updates its time base with values StbM_TimeStampType.nanoseconds = tA.Nano StbM_TimeStampType.seconds = tA .Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 4	**[CP]** Wait 800ms	
Step 5	**[CP]** Start RUN_TimeUser	
Step 6	**[RUN<RUN_TimeUser>]** Execute Rte_Call_ Client_GetCurrentTime	**[RUN<RUN_TimeUser>]** Rte_Call returns RTE_E_OK. Time read at Time user shall be: StbM_TimeStampType.nanoseconds = tA.Nano + <TestWaitTime = 800ms> StbM_TimeStampType.seconds = tA

		.Sec StbM_TimeStampType.secondsHi = tA .SecHi
Step 7	**[LT<CAN>]** Receives the SYNC message	**[LT]** Receives SYNC message in the format mentioned below: Byte 0: Type = 0x10 Byte 1: User Byte 1 = 0xBB Byte 2: D = 0x1 SC = 0x0 Byte 3: User Byte 0 = 0xAA Byte 4-7: SyncTimeSec = StbM_TimeStampType.seconds
Step 8	**[LT<CAN>]** Receives the FUP message	**[LT]** Receives FUP message in the format mentioned below: Byte 0: Type = 0x18 Byte 1: User Byte 2 = 0x00 Byte 2: D = 0x1 SC = 0x0 Byte 3: reserved (Bit 7 to Bit 3) = 0 SGW (Bit 2) = 0 OVS = Overflow of seconds (Bit 1 to Bit 0) Byte 4-7: SyncTimeNSec = StbM_TimeStampType.nanoseconds
Step 9	**[LT<CAN>]** Receives Global time base. Store the time as base for next periodic message processing	**[LT]** Get the time stamp values as below: StbM_TimeStampType.nanoseconds = tC.Nano StbM_TimeStampType.seconds = tC .Sec StbM_TimeStampType.secondsHi =

		tC .SecHi Time Received by LT<CAN> should be tC (Time base transmitted by LT<FlexRay> + ToleranceTime_CanTSyn + ToleranceTime_CanTSyn). ToleranceTime_CanTSyn = Max one main function of CanTSyn as reception is asynchronous. ToleranceTime_CanTSyn = Max one main function of CanTSyn as gateway is asynchronous.
Post-conditions	None	