Document Title	Acceptance Test Specification of Ecu Mode Management
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	665
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	• Added additional test cases for testing EcuM and BswM • Checked and adapted to Classic Platform Release 4.2.2
2015-10-31	1.1.0	AUTOSAR Release Management	• Some test steps were split to ensure that atomic test steps have a single PCO for their execution and their pass criteria • Checked and adapted to Classic Platform Release 4.2.1 • Formalization of point of control and observation
2014-07-30	1.0.0	AUTOSAR Release Management	• Initial release

Disclaimer

Advice for users

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

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

Table of Contents

1 Acronyms and abbreviations

Abbreviation / Acronym:	*Description:*
AT	Acceptance Test
CAN	Controller Area Network
ECU	Electronic Control Unit
LT	Lower Tester
NM	Network Management
PCO	Point of Control and Observation
PDU	Protocol Data Unit
RfC	Request for Change
Rx	Reception
SUT	System Under Test
DUT	Device Under Test
SWC	Software Component
TCP	Test Coordination Procedures
Tx	Transmission
UT	Upper Tester

2 Scope

The following test cases are used to verify the correct behavior of all the ECU mode management 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.

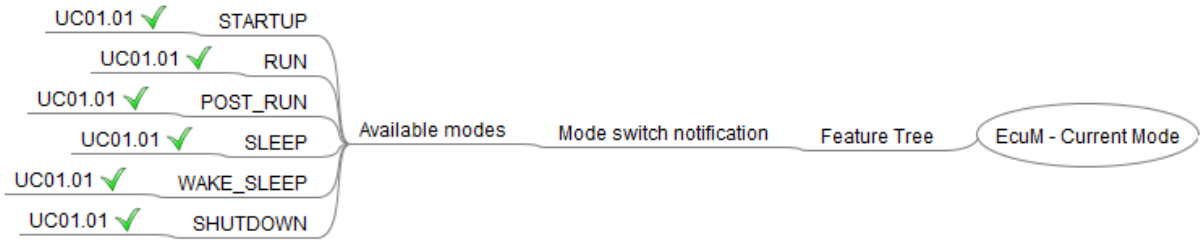
3 RS_BRF_01488 – EcuM Current Mode

3.1 General Test Objective and Approach

This Test Specification intends to cover the Current Mode feature of the EcuM as described in the AUTOSAR Feature [RS_BRF_01488].

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:

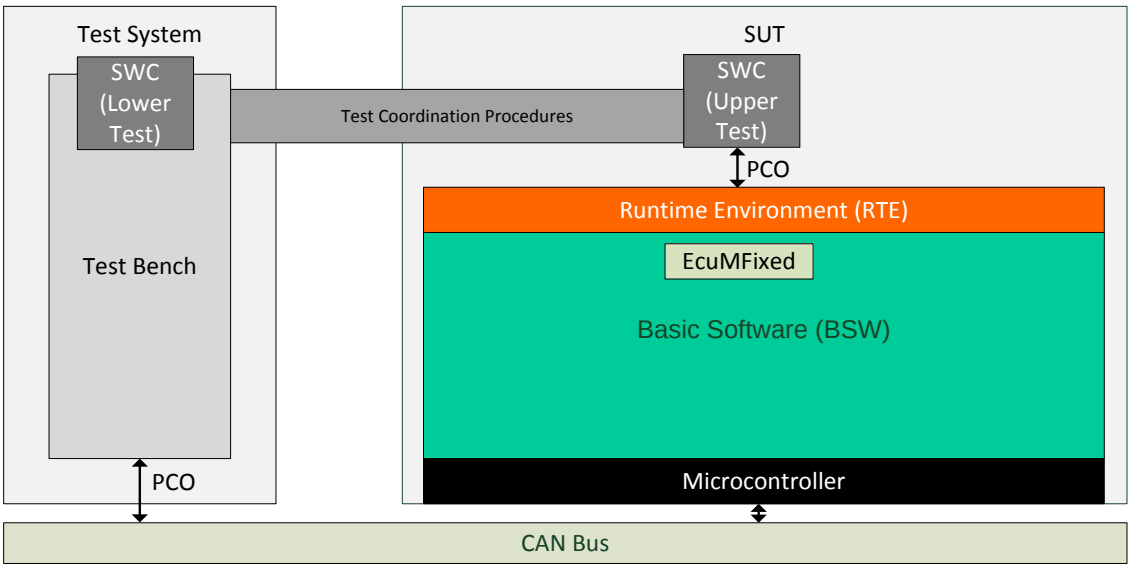


This specification gives the description of required tests environments (test bench, uses case, arxml files) and detailed tests cases for executing tests.

3.1.1 Test System

3.1.1.1 Overview on Architecture

The aim of this use case is to test the current mode feature of the EcuMFixed module. Each mode of the EcuM will be tested.



The test system architecture consists of Test Bench that executes only test sequencer and gives actions request through Test coordination Procedures to embedded SWC.

3.1.1.2 Specific Requirements

Not Applicable.

3.1.1.3 Test Coordination Requirements

Not Applicable.

3.1.2 Test Configuration

This section describes sets of requirements on configuration.

These sets are later referenced by test cases.

No configuration files are provided, they need to be developed when the test suites is implemented.

3.1.2.1 Required ECU Extract of System Description Files

For the EcuM tests cases on Current Mode feature, only one user is needed.

3.1.2.2 Required ECU Configuration Description Files

The section describes the common EcuC parameters between test cases that are required by the implementer of the test cases.

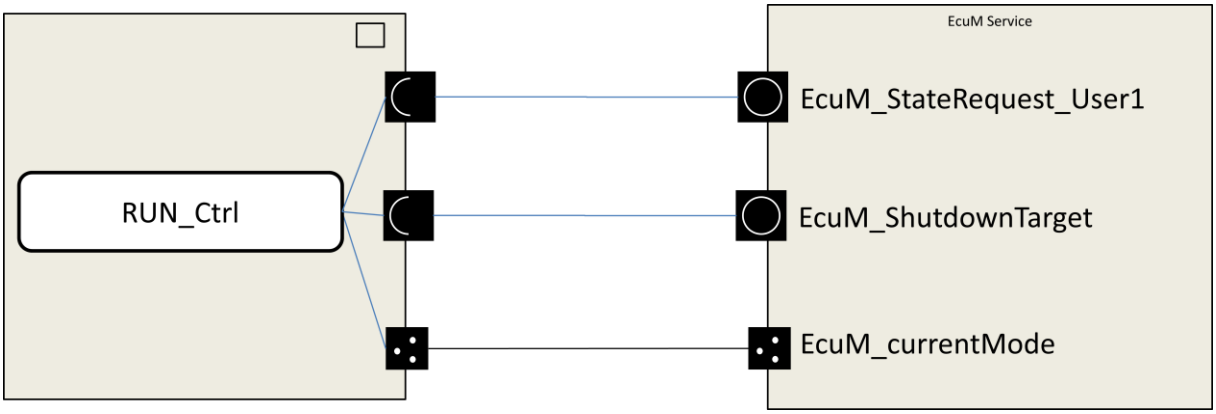
Use Case UC01.01:

- ➔ EcuMFixed Bsw component
- ➔ ECUMDefaultState = EcuMStateSleep
- ➔ EcuMRunMinimumDuration = 5 seconds

3.1.2.3 Required Software Component Description Files

The section describes the SWC-D that are required by the implementer of the test cases.

The SWC description is defined below:



3.1.2.4 Mandatory vs. Customizable Parts

Mandatory parameters are listed in Tests Cases (see chapter 3.3 Test Cases).

Customizable parameters are (these values are test case independent):

- Dem configuration
- Initialization list of the BSW
- The different sleep modes
- The wakeup sources

3.1.3 Test Case Design

Not Applicable

3.2 Re-usable Test Steps

Not Applicable

3.3 Test Cases

3.3.1 [ATS_ECUM_00113] Getting the current mode of EcuMFixed module

Test Objective	Getting the current mode of EcuMFixed module		
ID	ATS_ECUM_00113	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00037		
Trace to SWS Item	ECUStateManagerFixed: SWS_EcuM_00749 ECUStateManagerFixed: SWS_EcuM_00750 ECUStateManagerFixed: SWS_EcuM_00752 ECUStateManagerFixed: SWS_EcuMf_0031		
Requirements / Reference to Test Environment	Configuration use case : UC01.01		
Configuration Parameters	1 SWC user connected to EcuM_StateRequest interface and EcuM_currentMode interface One way to wakeup uses TTII configuration. This mode should be configured to allow entering WAKE_SLEEP state. Add a second wakeup source able to enter in RUN mode Configure a way for the LT to make sure that the ECU went to shutdown (e.g. Nm messages, periodic messages from COM ... etc).		
Summary	The aim of this test is to test the mode switch notification and the availability of the EcuM state through the service EcuM_CurrentMode. Here are the main steps of this test : 1. Start the SUT o Awaiting result : Mode notification must indicate a change in STARTUP mode		

	2. Request the RUN state ○ Awaiting result : Mode notification must indicate a change in RUN mode 3. Request the POSTRUN state 4. Release the RUN state ○ Awaiting result : Mode notification must indicate a change in POSTRUN mode 5. Release the POSTRUN state ○ Awaiting result : Mode notification must indicate a change in SLEEP mode 6. Wake up the SUT ○ Awaiting result : Mode notification must indicate a change in WAKE_SLEEP mode 7. Select the shutdown target OFF, and wait 5 seconds ○ Awaiting result : Mode notification must indicate a change in SHUTDOWN mode
Needed Adaptation to other Releases	**Needed Adaptation for any Release earlier than [4.2.1]** Configuration: [low] Test Steps: [low] `Names of shutdown targets differ in releases earlier than R4.2.1`
Pre-conditions	At Ecu Startup, the BswM activates the Com Channel used by ATF.

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** restart SUT	**[SWC]** Mode notification must indicate a change in STARTUP mode
Step 2	**[CP]** Wait EcuM to enter RUN	**[SWC]** Mode notification must indicate a change in RUN mode
Step 3	**[SWC]** query mode using EcuM_CurrentMode()	**[SWC]** Check that currentMode is RUN
Step 4	**[SWC]** executes EcuM_StateRequest operation RequestRUN() for User 1	-
Step 5	**[SWC]** query mode using EcuM_CurentMode()	**[SWC]** check that currentMode is RUN
Step 6	**[SWC]** executes EcuM_StateRequest operation RequestPOSTRUN() for User 1	-
Step 7	**[SWC]** query mode using EcuM_CurrentMode()	**[SWC]** Check that currentMode is RUN
Step 8	**[SWC]** executes EcuM_StateRequest operation ReleaseRUN() for User 1	**[SWC]** Mode notification must indicate a change in POSTRUN mode
Step 9	**[SWC]** query mode using EcuM_CurrentMode()	**[SWC]** Check that currentMode is

		POSTRUN
Step 10	**[SWC]** executes EcuM_StateRequest operation ReleasePOSTRUN() for User 1	-
Step 11	**[SWC]** executes ComM_UserRequest operation RequestComMode(NO_COMMUNICATION) to inactivete the ATF communication and allow ECU to go in Sleep mode (no other active user)	-
Step 12	**[SWC]** executes EcuM_StateRequest operation RequestPOST_RUN() for User 1	**[SWC]** Mode notification must indicate a change in SLEEP mode
Step 13	**[CP]** SUT is woken up by TTII	**[SWC]** Mode notification must indicate a change in WAKE_SLEEP mode
Step 14	**[CP]** wake SUT by wakeup source identified to enter RUN	**[SWC]** Mode Notification must indicate a change in RUN
Step 15	**[SWC]** executes EcuM_ShutdownTarget operation SelectShutdownTarget() to set shutdownTarget to ECUM_SHUTDOWN_TARGET_OFF	-
Step 16	**[SWC]** Release Ecu (by waiting exit from SelfRun or request no communication according to ATF implementation)	-
Step 17	**[CP]** waits 5 seconds	**[CP]** ECU is shutdown
Post-conditions	None	

3.3.2 [ATS_ECUM_00244] Getting the current mode of EcuMFixed module without POSTRUN state

Test Objective	Getting the current mode of EcuMFixed module without POSTRUN state		
ID	ATS_ECUM_00244	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00037		
Trace to SWS Item	ECUStateManagerFixed: SWS_EcuM_00749 ECUStateManagerFixed: SWS_EcuM_00750 ECUStateManagerFixed: SWS_EcuM_00752 ECUStateManagerFixed: SWS_EcuMf_0031		
Requirements / Reference	Configuration use case : UC01.01		

<table>
<tr><td>to Test Environment</td><td></td></tr>
<tr><td>Configuration Parameters</td><td>1 SWC user connected to EcuM Service through the EcuM_StateRequest interface and EcuM_currentMode interface.

One way to wakeup uses TTII configuration. This mode should be configured to allow entering WAKE_SLEEP state.

Add a second wakeup source able to enter in RUN mode</td></tr>
<tr><td>Summary</td><td>The aim of this test is to test the mode switch notification and the availability of the EcuM state through the service EcuM_CurrentMode.

Here are the main steps of this test :

1. Start the SUT
○ Awaiting result : Mode notification must indicate a change in STARTUP mode
2. Request the RUN state
○ Awaiting result : Mode notification must indicate a change in RUN mode
3. Release the RUN state
○ Awaiting result : Mode notification must indicate a change in SLEEP mode
4. Wake up the SUT
○ Awaiting result : Mode notification must indicate a change in WAKE_SLEEP mode
5. Select the shutdown target OFF, and wait 5 seconds
○ Awaiting result : Mode notification must indicate a change in SHUTDOWN mode</td></tr>
<tr><td>Needed Adaptation to other Releases</td><td>Needed Adaptation for any Release earlier than [4.2.1]
<table><tr><td>Configuration: [low]

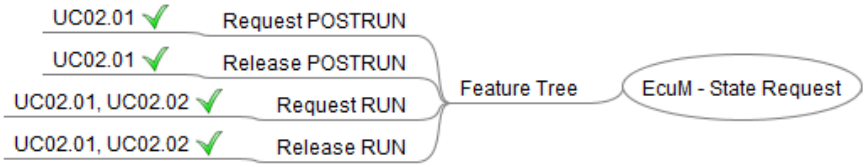
Test Steps: [low]</td><td>Names of shutdown targets differ in releases earlier than R4.2.1</td></tr></table></td></tr>
<tr><td>Pre-conditions</td><td>At Ecu Startup, the BswM activates the Com Channel used by ATF.</td></tr>
<tr><td colspan="3">Main Test Execution</td></tr>
<tr><td colspan="2">Test Steps</td><td>Pass Criteria</td></tr>
<tr><td>Step 1</td><td>[CP]
restart SUT</td><td>[SWC]
Mode notification must indicate a change in STARTUP mode</td></tr>
<tr><td>Step 2</td><td>[CP]
Wait EcuM to enter RUN</td><td>[SWC]
Mode notification must indicate a change in RUN mode</td></tr>
<tr><td>Step 3</td><td>[SWC]
query mode using EcuM_CurrentMode()</td><td>[SWC]
Check that currentMode is RUN</td></tr>
<tr><td>Step 4</td><td>[SWC]
executes ComM_UserRequest operation RequestComMode(NO_COMMUNICATION) to inactivate the ATF communication and allow ECU to go in Sleep mode (no other active user)</td><td>[SWC]
Mode notification must indicate a change in SLEEP mode</td></tr>
</table>

Step 5	**[SWC]** On Enter Sleep mode, query mode using EcuM_CurrentMode()	**[SWC]** Check that currentMode is SLEEP
Step 6	**[CP]** SUT is woken up by TTII	**[SWC]** Mode notification must indicate a change in WAKE_SLEEP mode
Step 7	**[CP]** wake SUT by wakeup source identified to enter RUN	**[SWC]** Mode Notification must indicate a change in RUN
Step 8	**[SWC]** executes EcuM_ShutdownTarget operation SelectShutdownTarget() to set shutdownTarget to ECUM_SHUTDOWN_TARGET_OFF	-
Step 9	**[SWC]** Release Ecu (by waiting exit from SelfRun or request no communication according to ATF implementation)	-
Step 10	**[CP]** waits 5 seconds	**[SWC]** Mode notification must indicate a change in SHUTDOWN mode
Post-conditions	None	

4 RS_BRF_01488 – EcuM State Request

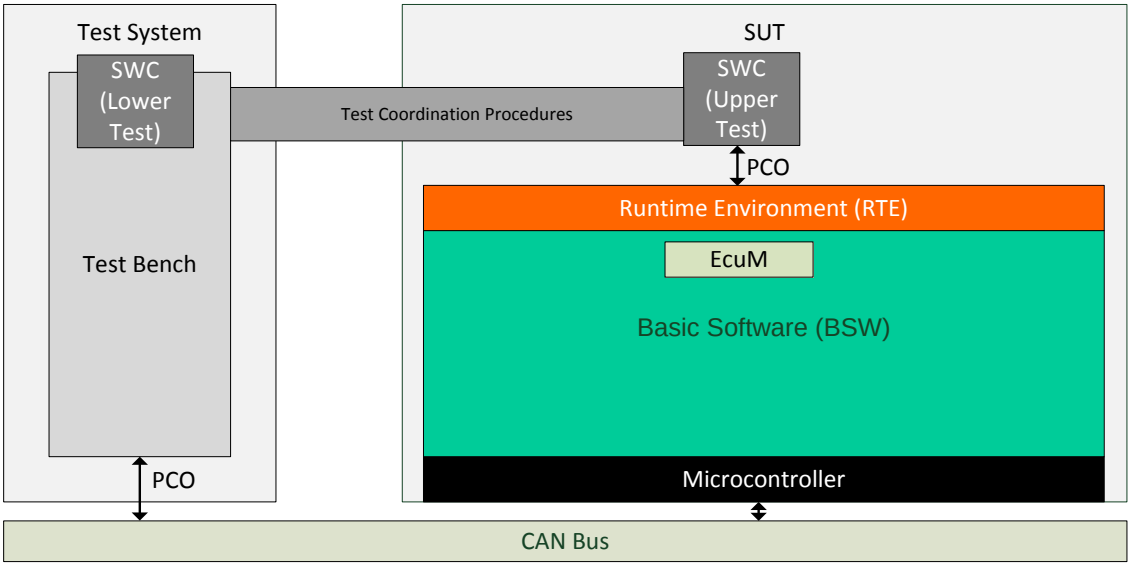
4.1 General Test Objective and Approach

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



4.1.1 Test System

4.1.1.1 Overview on Architecture



The test system architecture consists of SWC Upper Tester (3 SWCs) on the SUT.
Internal communication and mode switchs are handled on SUT side.
The Wait steps are handled on Test Bench side.

4.1.1.2 Specific Requirements

None.

4.1.1.3 Test Coordination Requirements

None.

4.1.2 Test Configuration

This section describes sets of requirements on configuration.

These sets are later referenced by test cases.

No configuration files are provided, they need to be developed when the test suites is implemented.

4.1.2.1 Required ECU Extract of System Description Files

For the EcuM tests cases on Current Mode feature, three users are needed.

4.1.2.2 Required ECU Configuration Description Files

The section describes the common EcuC parameters between test cases that are required by the implementer of the test cases.

Use Case UC02.01:

- ➔ EcuMFixed Bsw component
- ➔ EcuMRunMinimumDuration = 5 seconds
- ➔ Only one user configured
- ➔ TTII is deactivated

Use Case UC02.02:

- ➔ EcuMFixed Bsw component
- ➔ EcuMRunMinimumDuration = 5 seconds
- ➔ 3 users configured

4.1.2.3 Required Software Component Description Files

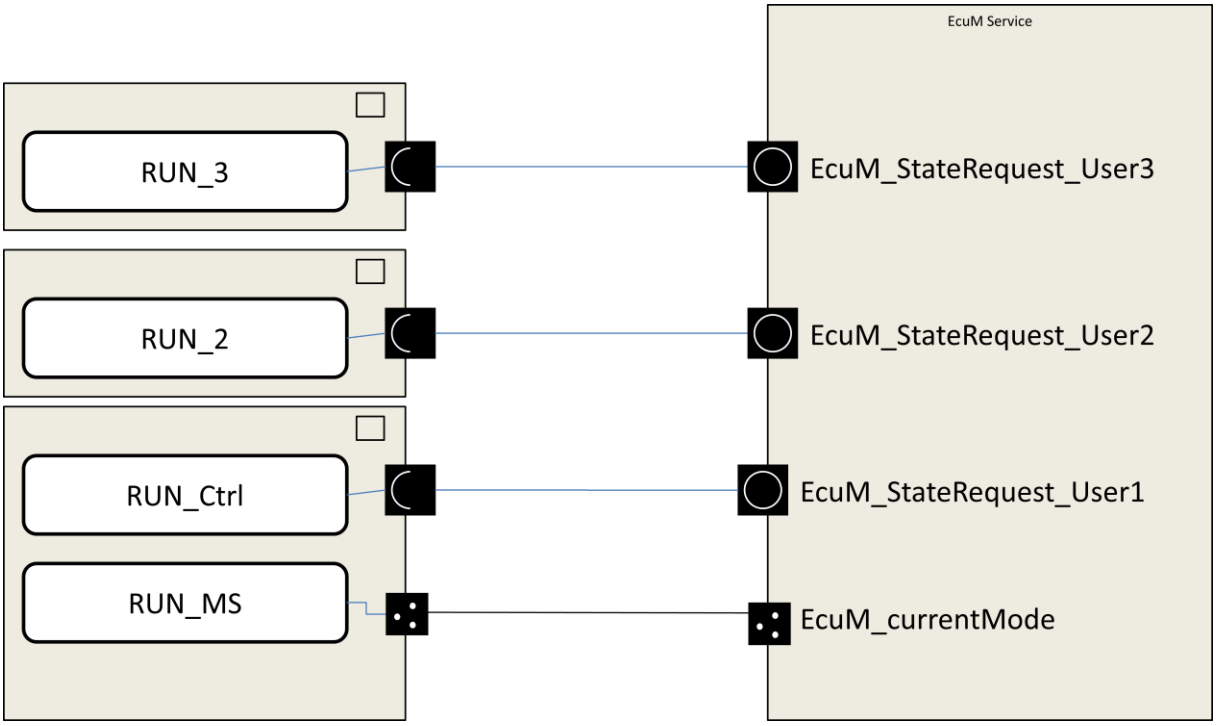
The section describes the SWC-D that are required by the implementer of the test cases.

For the EcuM tests cases on State Request, the SWC description required is the following:

UC02.02:

For this use case, 3 different users are needed to request RUN, POSTRUN and ReleaseRUN.

The connection to the EcuM Service is described below:



UC02.01:

As this configuration could reuse the previous configuration, only one SWC description is required to perform these tests.

EcuMRunMinimumDuration = 5 seconds

4.1.2.4 Mandatory vs. Customizable Parts

Mandatory parameters are listed in Tests Cases (see chapter 4.3 Test Cases).

Customizable parameters are (these values are test case independent):

- Dem configuration
- Initialization list of the BSW
- The different sleep modes
- The wakeup sources

4.1.3 Test Case Design

Not Applicable

4.2 Re-usable Test Steps

Not Applicable

4.3 Test Cases

4.3.1 [ATS_ECUM_00111] Requesting and releasing the RUN state on EcuMFixed

Test Objective	Requesting and releasing the RUN state on EcuMFixed		
ID	ATS_ECUM_00111	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00037		
Trace to SWS Item	ECUStateManagerFixed: SWS_EcuM_00814 ECUStateManagerFixed: SWS_EcuM_00815 ECUStateManagerFixed: SWS_EcuMf_0030		
Requirements / Reference to Test Environment	Configuration use case : UC02.01		
Configuration Parameters	EcuMRunMinimumDuration = 5 seconds 1 SWC EcuM user connected to SWC EcuM Service (EcuMFixed) through EcuM_StateRequest Client-Server Interface ECU can be woken up by CAN incoming frame (sent by TestBench). TTII is switched off Configure a way for the LT to make sure that the ECU went to shutdown (e.g. Nm messages, periodic messages from COM ... etc).		

Summary	The aim of this test is to verify the correct behavior of the following services :
	
	• RequestRUN
	• ReleaseRUN
	
	Here are the main steps of this test :
	
	1. Wake up the SUT
	2. Call the RequestRUN service
	3. Wait for 10 seconds
	  ○ Awaiting result : The SUT must NOT shutdown
	4. Make sure that no messages are sent on the bus including Network Management.
	5. Wait for (CanNmTimeoutTime + CanNmWaitBusSleepTime) seconds.
	6. Call ComM_GetCurrentComMode
	  ○ Awaiting result : The Current Com Mode should be COMM_NO_COMMUNICATION
	7. Call the ReleaseRUN service
	8. Wake up the SUT
	9. Wait for 4 seconds
	  ○ Awaiting result : The SUT must NOT shutdown
	10. Wait for 1 seconds
	  ○ Awaiting result : The SUT must shutdown		

Needed	None		

Adaptation to other Releases		
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** starts SWC	-
Step 2	**[SWC]** executes EcuM_StateRequest operation RequestRUN()	**[SWC]** EcuM_RequestRUN() should return E_OK
Step 3	**[CP]** wait 10 seconds	**[CP]** SUT should not shutdown
Step 4	**[LT]** Stop sending messages on the bus including "Network Management" message.	-
Step 5	**[CP]** Wait for (CanNmTimeoutTime + CanNmWaitBusSleepTime) Seconds *Note, Add a jitter to the configured time*	-
Step 6	**[SWC]** executes ComM_UserRequest operation GetCurrentComMode()	**[SWC]** The service should return E_OK ComMode should be COMM_NO_COMMUNICATION
Step 7	**[SWC]** executes EcuM_StateRequest operation ReleaseRUN()	**[SWC]** EcuM_ReleaseRUN() should return E_OK
Step 8	**[CP]** wait until SUT is shutdown	-
Step 9	**[CP]** wakes up SUT	-
Step 10	**[CP]** waits 4 seconds	**[CP]** SUT should NOT shutdown
Step 11	**[CP]** waits 1 seconds	**[CP]** SUT should shutdown
Post-conditions	None	

4.3.2 [ATS_ECUM_00112] Requesting and releasing the POSTRUN state on EcuMFixed

Test Objective	Requesting and releasing the POSTRUN state on EcuMFixed		
ID	ATS_ECUM_00112	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	EcuM, DET	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00037		
Trace to SWS Item	ECUStateManagerFixed: SWS_EcuM_00819 ECUStateManagerFixed: SWS_EcuM_00820		

	ECUStateManagerFixed: SWS_EcuMf_0030	
Requirements / Reference to Test Environment	Configuration use case : UC02.01	
Configuration Parameters	EcuMRunMinimumDuration = 5 seconds 1 SWC EcuM user connected to SWC EcuM Service (EcuMFixed) through EcuM_StateRequest Client-Server Interface ECU can be woken up by CAN incoming frame (sent by TestBench). Configure a way for the LT to make sure that the ECU went to shutdown (e.g. Nm messages, periodic messages from COM ... etc).	
Summary	The aim of this test is to verify the correct behavior of the following services : • RequestPOSTRUN • ReleasePOSTRUN	
Needed Adaptation to other Releases	None	
Pre-conditions	None	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** starts SWC	-
Step 2	**[SWC]** executes EcuM_StateRequest operation RequestRUN	**[SWC]** RequestRUN should return E_OK
Step 3	**[SWC]** executes EcuM_StateRequest operation RequestPOSTRUN	**[SWC]** RequestPOSTRUN should return E_OK
Step 4	**[LT]** Stop sending messages on the bus including "Network Management" message.	-
Step 5	**[CP]** Wait for (CanNmTimeoutTime + CanNmWaitBusSleepTime) Seconds *Note: Add a jitter to the configured time*	-
Step 6	**[SWC]** call ComM Service ComM_UserRequest operation GetCurrentComMode()	**[SWC]** GetCurrentComMode should return E_OK ComMode should be COMM_NO_COMMUNICATION
Step 7	**[SWC]** executes EcuM_StateRequest operation ReleaseRUN()	**[SWC]** ReleaseRUN should return E_OK
Step 8	**[CP]** waits 10 seconds	**[CP]** SUT should not shutdown
Step 9	**[SWC]** executes EcuM_StateRequest operation ReleasePOSTRUN()	**[SWC]** ReleasePOSTRUN should return E_OK

Step 10	**[CP]** waits until SUT is shutdown	-
Step 11	**[CP]** start SUT	-
Step 12	**[CP]** start SWC	-
Step 13	**[SWC]** executes EcuM_StateRequest operation RequestPOSTRUN()	**[SWC]** RequestPOSTRUN should return E_OK
Step 14	**[LT]** Stop sending messages on the bus including "Network Management" message.	-
Step 15	**[CP]** Wait for (CanNmTimeoutTime + CanNmWaitBusSleepTime) Seconds *Note: Add a jitter to the configured time*	-
Step 16	**[SWC]** call ComM Service ComM_UserRequest operation GetCurrentComMode()	**[SWC]** GetCurrentComMode should return E_OK ComMode should be COMM_NO_COMMUNICATION
Step 17	**[SWC]** executes EcuM_StateRequest operation ReleasePOSTRUN()	**[SWC]** ReleasePOSTRUN should return E_OK
Step 18	**[CP]** waits 4 seconds	**[CP]** SUT should not shutdown
Step 19	**[CP]** waits 1 second	**[CP]** SUT should shutdown
Step 20	**[CP]** start SUT	-
Step 21	**[SWC]** execute EcuM_StateRequest operation RequestPOSTRUN()	**[SWC]** RequestPOSTRUN should return E_OK
Step 22	**[SWC]** execute EcuM_StateRequest operation RequestPOSTRUN()	**[SWC]** RequestPOSTRUN should return E_NOT_OK
Step 23	**[SWC]** executes EcuM_StateRequest operation ReleasePOSTRUN()	**[SWC]** ReleasePOSTRUN should return E_OK
Step 24	**[SWC]** executes EcuM_StateRequest operation ReleasePOSTRUN()	**[SWC]** ReleasePOSTRUN should return E_NOT_OK
Step 25	**[SWC]** execute EcuM_StateRequest operation RequestRUN()	**[SWC]** RequestRUN should return E_OK
Step 26	**[SWC]** execute EcuM_StateRequest operation RequestRUN()	**[SWC]** RequestRUN should return E_NOT_OK
Step 27	**[SWC]** execute EcuM_StateRequest operation ReleaseRUN()	**[SWC]** ReleaseRUN should return E_OK
Step 28	**[SWC]**	**[SWC]**

	execute EcuM_StateRequest operation ReleaseRUN()	ReleaseRUN should return E_NOT_OK
Step 29	**[CP]** terminate SWC	-
Post-conditions	None	

4.3.3 [ATS_ECUM_00243] Requesting and releasing the RUN state in POSTRUN state on EcuMFixed

Test Objective	Requesting and releasing the RUN state in POSTRUN state on EcuMFixed		
ID	ATS_ECUM_00243	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00025 ATR: ATR_ATR_00037		
Trace to SWS Item	ECUStateManagerFixed: SWS_EcuM_00749 ECUStateManagerFixed: SWS_EcuM_00750 ECUStateManagerFixed: SWS_EcuM_00762		
Requirements / Reference to Test Environment	Configuration use case : UC02.02		
Configuration Parameters	EcuMRunMinimumDuration = 5 seconds 3 SWC EcuM users connected to SWC EcuM Service (EcuMFixed) through EcuM_StateRequest Client-Server Interface ECU can be woken up by incoming frame on the bus (sent by TestBench). Configure a way for the LT to make sure that the ECU went to shutdown (e.g. Nm messages, periodic messages from COM ... etc).		
Summary	The aim of this test is to verify the correct behavior of the following services when EcuM is in PostRun state : • RequestRUN • ReleaseRUN This test is done with multiple users (3 users configured in the EcuM). The aim of the test is to ensure that ECU do not quit the RUN state if there is still an active application.		
Needed Adaptation to other Releases	None		
Pre-conditions	None		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	**[CP]** starts RUN_Ctrl, RUN_2, RUN_3		-
Step 2	**[RUN<RUN_Ctrl>]**		**[RUN<RUN_Ctrl>]**

	call EcuM_StateRequest operation RequestRUN()	RequestRUN should return RTE_E_OK EcuM ModeSwitch port shall have the value RUN
Step 3	**[RUN<RUN_2>]** call EcuM_StateRequest operation RequestRUN()	**[RUN<RUN_2>]** RequestRUN should return RTE_E_OK EcuM ModeSwitch port shall have the value RUN
Step 4	**[RUN<RUN_3>]** call EcuM_StateRequest operation RequestRUN()	**[RUN<RUN_3>]** RequestRUN should return RTE_E_OK EcuM Mode Swith port shall have the value RUN
Step 5	**[CP]** wait 10 seconds	**[CP]** SUT should not shutdown
Step 6	**[RUN<RUN_2>]** call EcuM_StateRequest operation RequestPOSTRUN()	**[RUN<RUN_2>]** RequestPOSTRUN should return RTE_E_OK EcuM Switch port shall keep the value RUN and no mode switch occurs
Step 7	**[CP]** wait 10s	**[SWC]** EcuM Switch Port shall keep the value RUN and no mode switch occurs
Step 8	**[RUN<RUN_1>]** call EcuM_StateRequest operation RequestPOSTRUN()	**[RUN<RUN_1>]** RequestPOSTRUN should return RTE_E_OK EcuM Switch port shall keep the value RUN and no mode switch occurs
Step 9	**[CP]** wait 10s	**[CP]** SUT should not shutdown
Step 10	**[RUN<RUN_3>]** call EcuM_StateRequest operation RequestPOSTRUN()	**[RUN<RUN_3>]** RequestPOSTRUN should return RTE_E_OK EcuM Switch port shall return the value RUN and no mode switch occurs
Step 11	**[LT]** Stop sending messages on the bus including "Network Management" message.	-
Step 12	**[CP]** Wait for (CanNmTimeoutTime + CanNmWaitBusSleepTime) Seconds *Note: Add a jitter to the configured time*	-
Step 13	**[SWC]** call ComM Service ComM_UserRequest operation GetCurrentComMode()	**[SWC]** GetCurrentComMode should return E_OK

		Current Com Mode should be COMM_NO_COMMUNICATION
Step 14	**[RUN<RUN_Ctrl>]** call EcuM_StateRequest operation ReleaseRUN()	-
Step 15	**[RUN<RUN_2>]** call EcuM_StateRequest operation ReleaseRUN()	-
Step 16	[RUN<RUN_3>] call EcuM_StateRequest operation ReleaseRUN()	-
Step 17	**[CP]** Wait 10s	**[CP]** SUT should not shutdown
Step 18	**[RUN<RUN_Ctrl>]** call EcuM_StateRequest operation ReleasePOSTRUN()	-
Step 19	**[RUN<RUN_2>]** call EcuM_StateRequest operation ReleasePOSTRUN()	-
Step 20	**[RUN<RUN_3>]** call EcuM_StateRequest operation ReleasePOSTRUN()	-
Step 21	**[CP]** wait 10s	**[CP]** SUT should shutdown
Step 22	**[LT]** Send any frame on the bus to wake-up the ECU/SUT	**[CP]** SUT should wake-up
Step 23	**[CP]** Restart RUN_Ctrl, RUN_2, RUN_3	-
Step 24	**[RUN<RUN_Ctrl>]** call EcuM_StateRequest operation RequestRUN()	**[RUN<RUN_Ctrl>]** EcuM Switch port should return the value RUN
Step 25	**[RUN<RUN_Ctrl>]** call EcuM_StateRequest operation RequestPOSTRUN()	**[RUN<RUN_Ctrl>]** EcuM Switch port shall return the value RUN and no mode switch occurs
Step 26	**[CP]** wait 2s	**[CP]** SUT should not shutdown
Step 27	**[RUN<RUN_2>]** call EcuM_StateRequest operation RequestRUN()	**[RUN<RUN_2>]** EcuM Switch port shall return the value RUN and no mode switch occurs
Step 28	**[RUN<RUN_Ctrl>]** call EcuM_StateRequest operation ReleasePOSTRUN()	**[RUN<RUN_Ctrl>]** EcuM Switch Port should return the value RUN
Step 29	**[LT]** Stop sending messages on the bus including "Network Management" messages.	-
Step 30	**[CP]** Wait for (CanNmTimeoutTime + CanNmWaitBusSLeepTime) seconds *Note: Add a jitter to the configured time*	-
Step 31	**[SWC]** call ComM_UserRequest operation	**[SWC]** GetCurrentComMode should return

	GetCurrentComMode()	E_OK ComMode should be COMM_NO_COMMUNICATION
Step 32	**[RUN<RUN_Ctrl>]** call EcuM_StateRequest operation ReleaseRUN()	-
Step 33	**[RUN<RUN_2>]** call EcuM_StateRequest operation ReleaseRUN()	-
Step 34	**[CP]** wait 10s	**[CP]** SUT should shutdown
Post-conditions	None	

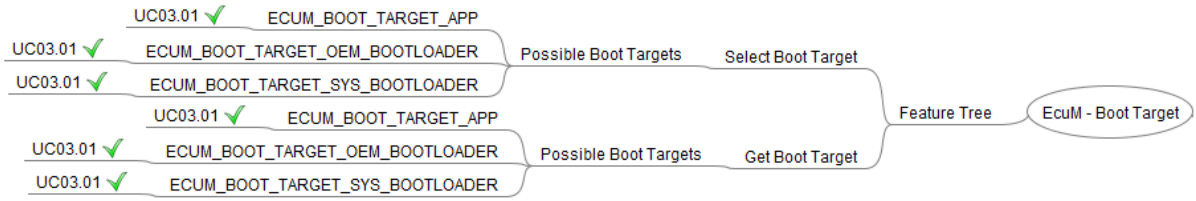
5 RS_BRF_02152 – EcuM Boot Target

5.1 General Test Objective and Approach

This Test Specification intends to cover the Current Mode feature of the EcuM as described in the AUTOSAR Feature [RS_BRF_02052].

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:

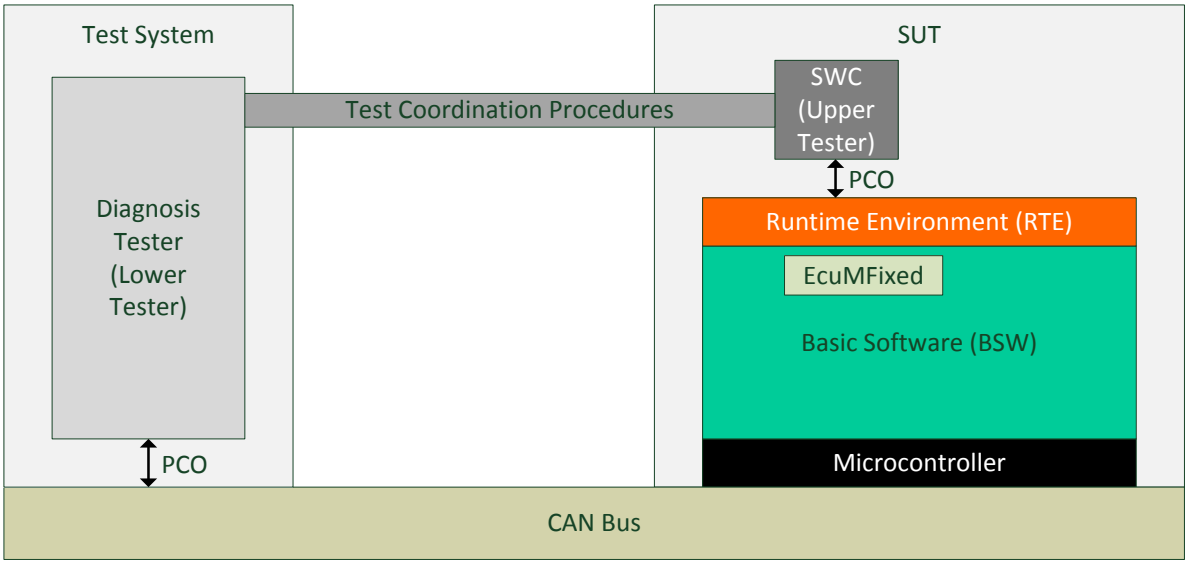


This specification gives the description of required tests environments (test bench, uses case, arxml files) and detailed tests cases for executing tests.

5.1.1 Test System

5.1.1.1 Overview on Architecture

The aim of this use case is to test the boot target feature of the EcuMFixed module.



The test system architecture consists of Test Bench that executes only test sequencer and gives actions request through Test coordination Procedures to embedded SWC.

5.1.1.2 Specific Requirements

Not Applicable.

5.1.1.3 Test Coordination Requirements

Not Applicable.

5.1.2 Test Configuration

This section describes sets of requirements on configuration.

These sets are later referenced by test cases.

No configuration files are provided, they need to be developed when the test suites is implemented.

5.1.2.1 Required ECU Extract of System Description Files

For the EcuM tests cases on Boot Target feature, only one user is needed.

5.1.2.2 Required ECU Configuration Description Files

The section describes the common EcuC parameters between test cases that are required by the implementer of the test cases.

Use Case UC03.01:

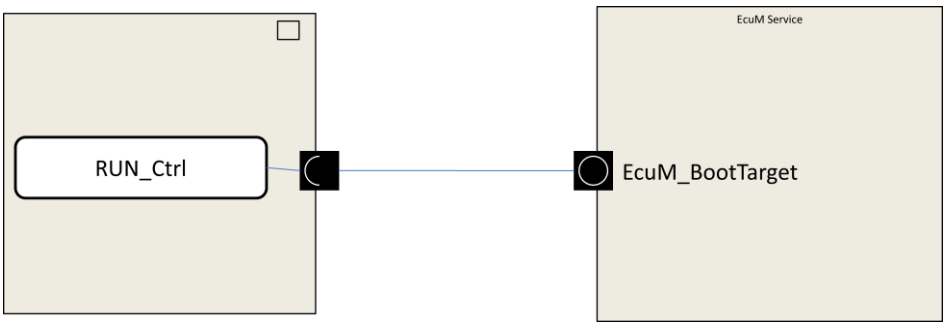
➔ EcuMFixed Bsw component

5.1.2.3 Required Software Component Description Files

The section describes the SWC-D that are required by the implementer of the test cases.

UC03.01

The SWC description is defined below:



5.1.2.4 Mandatory vs. Customizable Parts

Mandatory parameters are listed in Tests Cases (see chapter 5.3 Test Cases).

Customizable parameters are (these values are test case independent):

- Dem configuration
- Initialization list of the BSW
- The different sleep modes
- The wakeup sources

5.1.3 Test Case Design

Not Applicable

5.2 Re-usable Test Steps

Not Applicable

5.3 Test Cases

5.3.1 [ATS_ECUM_00114] Requesting and getting the Boot Target "Application" on EcuMFixed

Test Objective	Requesting and getting the Boot Target "Application" on EcuMFixed		
ID	ATS_ECUM_00114	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00036		
Trace to SWS Item	ECUStateManager: SWS_EcuM_02835 ECUStateManagerFixed: SWS_EcuM_02836 ECUStateManagerFixed: SWS_EcuMf_0033		
Requirements / Reference to Test Environment	Configuration use case : UC03.01		
Configuration Parameters	1 SWC EcuM user connected to SWC EcuM Service through EcuM_BootTarget Client-Server Interface Connection to Server ShutdownTarget Interface : - SelectBootTarget - GetBootTarget		

Summary	The aim of this test is to verify the behavior of the Boot Target feature.
	
	Here are the main steps of this test :
	
	1. Get the Boot Target
	  ○ Awaited result : ECUM_BOOT_TARGET_APP
	2. Set the boot target to ECUM_BOOT_TARGET_OEM_BOOTLOADER
	3. Get the Boot Target
	  ○ Awaited result : ECUM_BOOT_TARGET_OEM_BOOTLOADER
	4. Set the boot target to ECUM_BOOT_TARGET_APP
	5. Get the Boot Target
	  ○ Awaited result : ECUM_BOOT_TARGET_APP		

Needed Adaptation to other Releases	None		
Pre-conditions	SUT has been initialized with Boot Target : ECUM_BOOT_TARGET_APP		
Main Test Execution			
Test Steps		**Pass Criteria**	
Step 1	**[CP]** start SWC	-	
Step 2	**[SWC]**	**[SWC]**	

	executes EcuM_StateRequest operation GetBootTarget() to get boot target	GetBootTarget() should return E_OK Boot target should be ECUM_BOOT_TARGET_APP
Step 3	**[SWC]** executes EcuM_StateRequest operation SelectBootTarget() to set boot target to ECUM_BOOT_TARGET_OEM_BOOTLOADER	**[SWC]** SelectBootTarget() should return E_OK
Step 4	**[SWC]** executes EcuM_StateRequest operation GetBootTarget() to get boot target	**[SWC]** GetBootTarget() should return E_OK Boot target should be ECUM_BOOT_TARGET_OEM_BOOTLOADER
Step 5	**[SWC]** executes EcuM_StateRequest operation SelectBootTarget() to set boot target to ECUM_BOOT_TARGET_APP	**[SWC]** SelectBootTarget() should return E_OK
Step 6	**[SWC]** executes EcuM_StateRequest operation GetBootTarget() to get boot target	**[SWC]** GetBootTarget() should return E_OK Boot target should be ECUM_BOOT_TARGET_APP
Step 7	**[CP]** terminates SWC	-
Post-conditions	None	

5.3.2 [ATS_ECUM_00115] Requesting and getting the Boot Target "System Bootloader" on EcuMFixed

Test Objective	Requesting and getting the Boot Target "System Bootloader" on EcuMFixed		
ID	ATS_ECUM_00115	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00036		
Trace to SWS Item	ECUStateManager: SWS_EcuM_02835 ECUStateManagerFixed: SWS_EcuM_02836 ECUStateManagerFixed: SWS_EcuMf_0033		
Requirements / Reference to Test Environment	Configuration use case : UC03.01		
Configuration	1 SWC EcuM user connected to SWC EcuM Service through EcuM_BootTarget Client-Server Interface		

Parameters	Connection to Server ShutdownTarget Interface : - SelectBootTarget - GetBootTarget	
Summary	The aim of this test is to verify the behavior of the Boot Target feature. Here are the main steps of this test : 1. Set the boot target to ECUM_BOOT_TARGET_SYS_BOOTLOADER 2. Get the Boot Target o Awaited result : ECUM_BOOT_TARGET_SYS_BOOTLOADER 3. Set the boot target to ECUM_BOOT_TARGET_OEM_BOOTLOADER 4. Get the Boot Target o Awaited result : ECUM_BOOT_TARGET_OEM_BOOTLOADER	
Needed Adaptation to other Releases	None	
Pre-conditions	SUT is started	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** starts SWC	-
Step 2	**[SWC]** executes EcuM_StateRequest operation SelectBootTarget() to set boot target to ECUM_BOOT_TARGET_SYS_BOOTLOADER	**[SWC]** SelectBootTarget() should return E_OK
Step 3	**[SWC]** executes EcuM_StateRequest operation GetBootTarget() to get boot target	**[SWC]** GetBootTarget() should return E_OK Boot target should be ECUM_BOOT_TARGET_SYS_BOOTLOADER
Step 4	**[SWC]** executes EcuM_StateRequest operation SelectBootTarget() to set boot target to ECUM_BOOT_TARGET_OEM_BOOTLOADER	**[SWC]** SelectBootTarget() should return E_OK
Step 5	**[SWC]** executes EcuM_StateRequest operation GetBootTarget() to get boot target	**[SWC]** GetBootTarget() should return E_OK Boot target should be ECUM_BOOT_TARGET_OEM_BOOTLOADER
Step 6	**[CP]** terminates SWC	
Post-conditions	None	

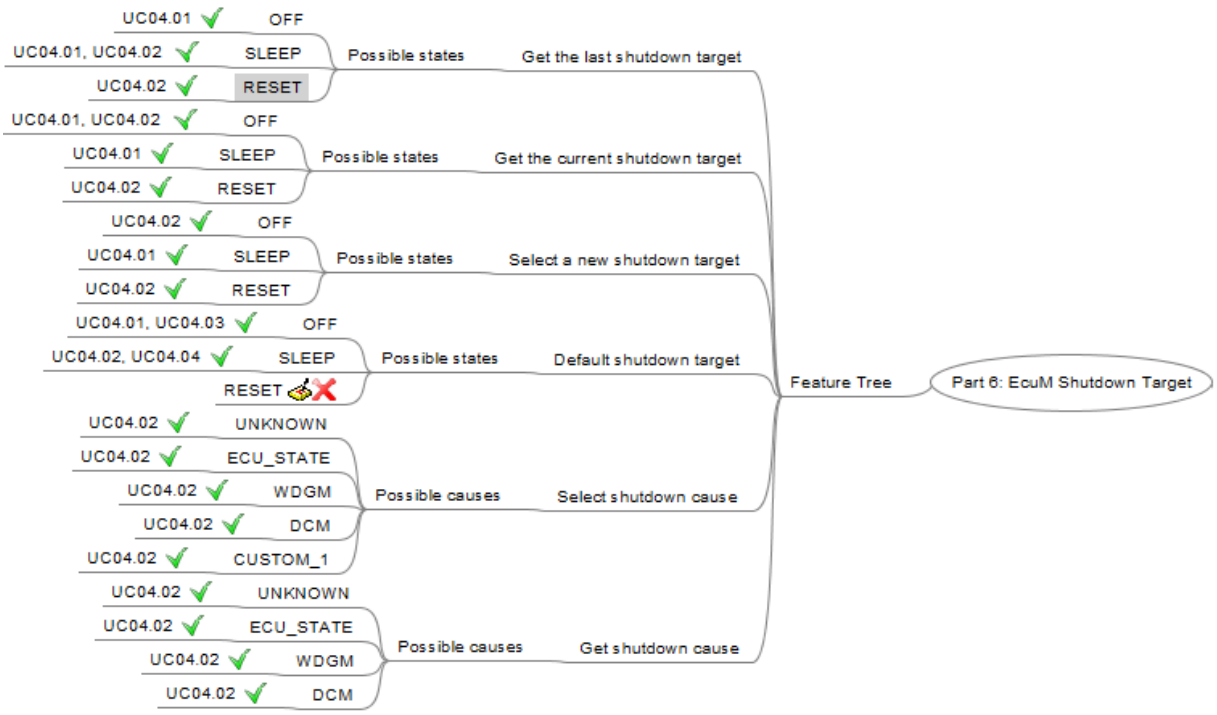
6 RS_BRF_02152 – EcuM Shutdown Target

6.1 General Test Objective and Approach

This Test Specification intends to cover the Shutdown Target feature of the EcuM as described in the AUTOSAR Feature [RS_BRF_02152].

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:

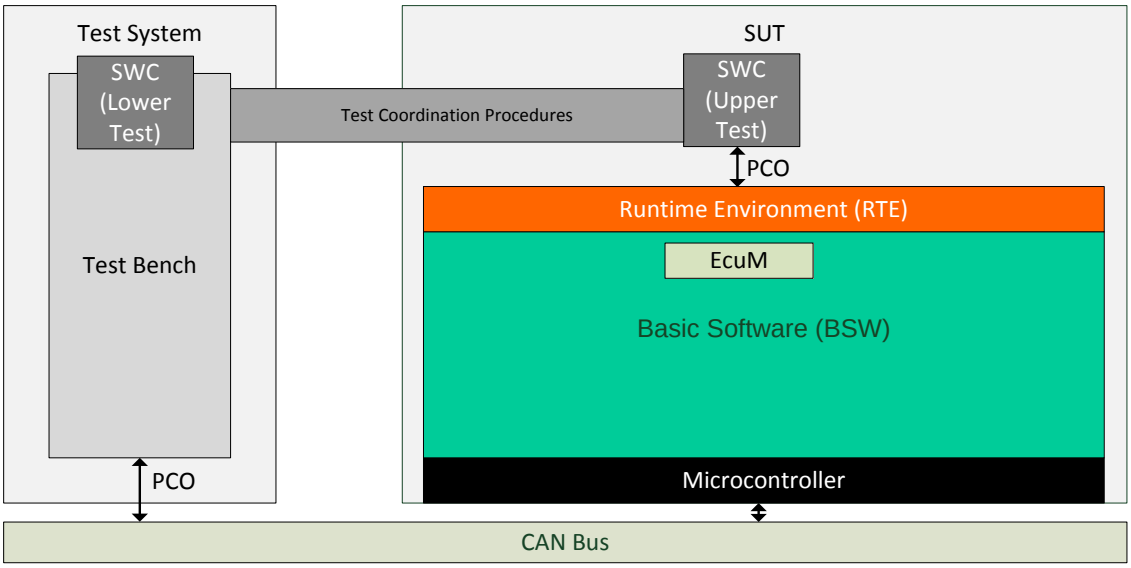


This specification gives the description of required tests environments (test bench, uses case, arxml files) and detailed tests cases for executing tests.

6.1.1 Test System

6.1.1.1 Overview on Architecture

The aim of this use case is to test the Shutdown Target feature of the EcuMFixed/EcuMFlex module.



The test system architecture consists of Test Bench that executes only test sequencer and gives actions request through Test coordination Procedures to embedded SWC.

6.1.1.2 Specific Requirements

Not Applicable.

6.1.1.3 Test Coordination Requirements

Not Applicable.

6.1.2 Test Configuration

This section describes sets of requirements on configuration.

These sets are later referenced by test cases.

No configuration files are provided, they need to be developed when the test suites is implemented.

6.1.2.1 Required ECU Extract of System Description Files

For the EcuM tests cases on Shutdown Target feature, only one user is needed.

6.1.2.2 Required ECU Configuration Description Files

The section describes the common EcuC parameters between test cases that are required by the implementer of the test cases.

Use Case UC04.01:

- ➔ EcuM Fixed module is used
- ➔ EcuMDefaultState = EcuMStateOff

Use Case UC04.02:

- ➔ EcuM Flexible module is used
- ➔ EcuMDefaultState = EcuMStateSleep

Use Case UC04.03:

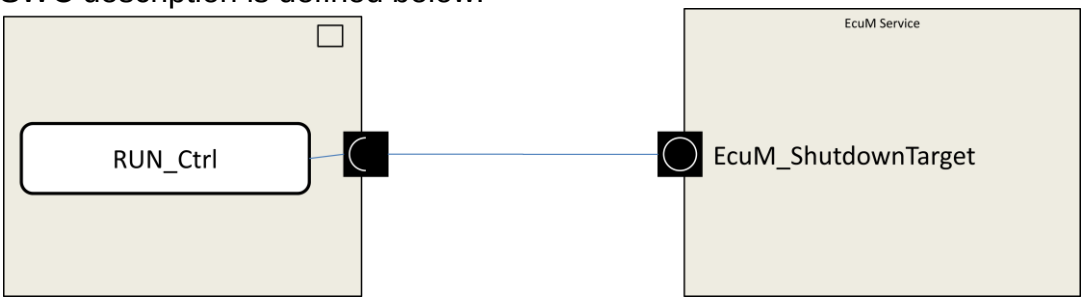
- ➔ EcuM Flexible module is used
- ➔ EcuMDefaultState = EcuMStateOff

Use Case UC04.04:

- ➔ EcuM Fixed module is used
- ➔ EcuMDefaultState = EcuMStateSleep

6.1.2.3 Required Software Component Description Files

The SWC description is defined below:



6.1.2.4 Mandatory vs. Customizable Parts

Mandatory parameters are listed in Tests Cases (see chapter 6.3 Test Cases).

Customizable parameters are (these values are test case independent):

- Dem configuration
- Initialization list of the BSW
- The different sleep modes
- The wakeup sources

6.1.3 Test Case Design

Not Applicable

6.2 Re-usable Test Steps

Not Applicable

6.3 Test Cases

6.3.1 [ATS_ECUM_00108] Selecting shutdown targets, and getting the current and the last shutdown target (Default Off)

Test Objective	Selecting shutdown targets, and getting the current and the last shutdown target (Default Off)		
ID	ATS_ECUM_00108	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00036		
Trace to SWS Item	ECUStateManager: SWS_EcuM_02822 ECUStateManager: SWS_EcuM_02824 ECUStateManager: SWS_EcuM_02825 ECUStateManagerFixed: SWS_EcuMf_0032		
Requirements / Reference to Test Environment	Configuration use case : UC04.01, UC04.03		
Configuration Parameters	1 SWC EcuM user connected to SWC EcuM Service through EcuM_ShutdownTarget Client-Server Interface Connection to Server ShutdownTarget Interface : - GetShutdownTarget - SelectShutdownTarget - GetLastShutdownTarget		
Summary	The goal of this test consists in testing the interface EcuM_ShutdownTarget for the EcuMFixed/EcuMFlex versions of the EcuM module. Here are the main steps of this test case : 1. Get the current shutdown target o Awaiting result : Shutdown target = OFF 2. Switch off the SUT, then switch on the SUT 3. Get the last shutdown target o Awaiting result : Shutdown target = OFF 4. Select the shutdown target SLEEP 5. Get the current shutdown target o Awaiting result : Shutdown target = SLEEP 6. Get the last shutdown target o Awaiting result : Shutdown target = OFF 7. Switch off the SUT, then switch on the SUT 8. Get the last shutdown target o Awaiting result : Shutdown target = SLEEP		
Needed Adaptation to other Releases	**Needed Adaptation for Release [3.2.2]** Configuration: [low] Test Steps: [low] `EcuM Flex do not exist in R3.2.2.`		

		Use UC04.01 only and exclude running this test case on UC04.03
	Needed Adaptation for any Release earlier than [4.2.1]	
	Configuration: [low] Test Steps: [low]	`Names of shutdown targets differ in releases earlier than R4.2.1`
Pre-conditions	The SUT is started.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[SWC]** executes EcuM_ShutdownTarget operation GetShutdownTarget() to get current shutdown target	**[SWC]** EcuM_ShutdownTarget operation GetShutdownTarget() should return E_OK Current shutdown target (parameter target) should be ECUM_SHUTDOWN_TARGET_OFF
Step 2	**[CP]** restarts SUT	-
Step 3	**[SWC]** executes EcuM_ShutdownTarget operation GetLastShutdownTarget() to get last shutdown target	**[SWC]** GetLastShutdownTarget() should return E_OK Last shutdown target (parameter target) should be ECUM_SHUTDOWN_TARGET_OFF
Step 4	**[SWC]** executes EcuM_ShutdownTarget operation SelectShutdownTarget() with shutdown target ECUM_SHUTDOWN_TARGET_SLEEP	**[SWC]** SelectShutdownTarget() should return E_OK
Step 5	**[SWC]**: executes EcuM_ShutdownTarget operation GetShutdownTarget() to get current shutdown target	**[SWC]**: GetShutdownTarget() should return E_OK Current shutdown target (parameter target) should be ECUM_SHUTDOWN_TARGET_SLEEP
Step 6	**[SWC]** executes EcuM_ShutdownTarget operation GetLastShutdownTarget() to get last shutdown target	**[SWC]** GetShutdownTarget() should return E_OK Last shutdown target (parameter target) should be ECUM_SHUTDOWN_TARGET_OFF
Step 7	**[CP]**	-

	restarts SUT	
Step 8	**[CP]** starts SWC	-
Step 9	**[SWC]** executes EcuM_ShutdownTarget operation GetLastShutdown Target() to get last shutdown target	**[SWC]** GetShutdownTarget() should return E_OK Last shutdown target (parameter target) should be ECUM_SHUTDOWN_TARGET_SLEEP
Post-conditions	None	

6.3.2 [ATS_ECUM_00109] Selecting shutdown targets, and getting the current and the last shutdown target (Default Sleep)

Test Objective	Selecting shutdown targets, and getting the current and the last shutdown target (Default Sleep)		
ID	ATS_ECUM_00109	**AUTOSAR Releases**	4.2.1|4.2.2
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00036		
Trace to SWS Item	ECUStateManager: SWS_EcuM_02822 ECUStateManager: SWS_EcuM_02824 ECUStateManager: SWS_EcuM_02825 ECUStateManager: SWS_EcuM_03011 ECUStateManager: SWS_EcuM_02979 ECUStateManagerFixed: SWS_EcuMf_0032		
Requirements / Reference to Test Environment	Configuration use case : UC04.02, UC04.04		
Configuration Parameters	1 SWC EcuM user connected to SWC EcuM Service through EcuM_ShutdownTarget Client-Server Interface Connection to Server ShutdownTarget Interface : - GetShutdownTarget - SelectShutdownTarget - GetLastShutdownTarget		
Summary	The goal of this test consists in testing the interface EcuM_ShutdownTarget for the EcuMFixed/EcuMFlex versions of the EcuM module. Here are the main steps of this test case : 1. Get the current shutdown target ○ Awaiting result : Shutdown target = SLEEP		

	2. Switch off the SUT, then switch on the SUT 3. Get the last shutdown target o Awaiting result : Shutdown target = SLEEP 4. Select the shutdown target OFF 5. Get the current shutdown target o Awaiting result : Shutdown target = OFF 6. Get the last shutdown target o Awaiting result : Shutdown target = SLEEP 7. Select the shutdown target RESET 8. Get the current shutdown target o Awaiting result : Shutdown target = RESET 9. Get the last shutdown target o Awaiting result : Shutdown target = SLEEP 10. Switch off the SUT, then switch on the SUT 11. Get the last shutdown target o Awaiting result : Shutdown target = RESET
Needed Adaptation to other Releases	**Needed Adaptation for Release [3.2.2]** Configuration: [low] — `EcuM Flex do not exist in R3.2.2.` Test Steps: [low] — This test case shall be removed **Needed Adaptation for any Release earlier than [4.2.1]** Configuration: [low] — `Names of shutdown targets differ in releases earlier than R4.2.1` Test Steps: [low]
Pre-conditions	The SUT is started.

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** starts SWC	-
Step 2	**[SWC]** executes EcuM_ShutdownTarget operation GetShutdownTarget() to get current shutdown target	**[SWC]** GetShutdownTarget() should return E_OK Current shutdown target (parameter target) should be ECUM_SHUTDOWN_TARGET_SLEEP
Step 3	**[CP]** restart SUT	-
Step 4	**[CP]** starts SWC	-
Step 5	**[SWC]** executes EcuM_ShutdownTarget operation GetLastShutdownTarget() to get last shutdown target	**[SWC]** GetLastShutdownTarget() should return E_OK Last shutdown target (parameter target) should be ECUM_SHUTDOWN_TARGET_SLEEP

Step 6	**[SWC]** executes EcuM_ShutdownTarget operation SelectShutdownTarget() to set shutdown target to ECUM_SHUTDOWN_TARGET_OFF	**[SWC]** SelectShutdownTarget() should return E_OK
Step 7	**[SWC]** executes EcuM_ShutdownTarget operation GetShutdownTarget() to get current shutdown target	**[SWC]** GetShutdownTarget() should return E_OK Current shutdown target (parameter target) should be ECUM_SHUTDOWN_TARGET_OFF
Step 8	**[SWC]** executes EcuM_ShutdownTarget operation GetLastShutdownTarget() to get last shutdown target	**[SWC]** GetLastShutdownTarget() should return E_OK Last shutdown target (parameter target) should be ECUM_SHUTDOWN_TARGET_SLEEP
Step 9	**[SWC]** executes EcuM_ShutdownTarget operation SelectShutdownTarget() to set shutdown target to ECUM_SHUTDOWN_TARGET_RESET	**[SWC]** SelectShutdownTarget() should return E_OK
Step 10	**[SWC]** executes EcuM_ShutdownTarget operation GetShutdownTarget() to get current shutdown target	**[SWC]** GetShutdownTarget() should return E_OK Current shutdown target (parameter target) should be ECUM_SHUTDOWN_TARGET_RESET
Step 11	**[SWC]** executes EcuM_ShutdownTarget operation GetLastShutdownTarget() to get last shutdown target	**[SWC]** GetLastShutdownTarget() should return E_OK Last shutdown target (parameter target) should be ECUM_SHUTDOWN_TARGET_SLEEP
Step 12	**[CP]** restarts SUT	-
Step 13	**[CP]** starts SWC	-
Step 14	**[SWC]** executes EcuM_ShutdownTarget operation GetLastShutdownTarget() to get last shutdown target	**[SWC]** GetLastShutdownTarget() should return E_OK Last shutdown target (parameter target) should be ECUM_SHUTDOWN_TARGET_RESET
Step 15	**[CP]** terminates SWC	-
Post-conditions	None	

6.3.3 [ATS_ECUM_00110] Selecting shutdown causes and getting shutdown causes on EcuMFlex

Test Objective	Selecting shutdown causes and getting shutdown causes on EcuMFlex		
ID	ATS_ECUM_00110	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1|4.2.1|4.2.2
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00036		
Trace to SWS Item	ECUStateManager: SWS_EcuM_04050 ECUStateManager: SWS_EcuM_04051 ECUStateManager: SWS_EcuM_03011 ECUStateManager: SWS_EcuM_02979		
Requirements / Reference to Test Environment	Configuration use case : UC04.02		
Configuration Parameters	1 SWC EcuM user connected to SWC EcuM Service (EcuMFlex) through EcuM_ShutdownTarget Client-Server Interface Connection to Server ShutdownTarget Interface : - SelectShutdownCause - GetShutdownCause EcuMShutdownCause(no upstream template parameter): - ECUM_CAUSE_ECU_STATE - ECUM_CAUSE_WDGM - ECUM_CAUSE_DCM - ECUM_CAUSE_CUSTOM_1		
Summary	The goal of this test consists in testing the interface EcuM_ShutdownTarget for the EcuMFlex version of the EcuM module. Here are the main steps of this test case : 1. Select the shutdown cause ECU_STATE 2. Get the shutdown cause o Expected result : ECU_STATE 3. Select the shutdown cause WDGM 4. Get the shutdown cause o Expected result : WDGM 5. Select the shutdown cause DCM 6. Get the shutdown cause o Expected result : DCM 7. Select the shutdown cause UNKNOWN 8. Get the shutdown cause o Expected result : UNKNOWN 9. Select the shutdown cause CUSTOM_1 10. Get the shutdown cause o Expected result : CUSTOM_1		
Needed Adaptation to other Releases	**Needed Adaptation for Release [3.2.2]** Configuration: [low]		

	Test Steps: [low]	`EcuM Flex do not exist in R3.2.2.` This test case shall be removed
Pre-conditions	The SUT is started.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	**[CP]** starts SWC	-
Step 2	**[SWC]** executes EcuM_ShutdownTarget operation SelectShutdownCause() to select shutdown cause ECU_STATE	**[SWC]** SelectShutdownCause() should return E_OK
Step 3	**[SWC]** executes EcuM_ShutdownTarget operation GetShutdownCause() to get shutdown cause	**[SWC]** GetShutdownCause() should return E_OK Shutdown cause should be ECU_STATE
Step 4	**[SWC]** executes EcuM_ShutdownTarget operation SelectShutdownCause() to select shutdown cause WDGM	**[SWC]** SelectShutdownCause() should return E_OK
Step 5	**[SWC]** executes EcuM_ShutdownTarget operation GetShutdownCause() to get shutdown cause	**[SWC]** GetShutdownCause() should return E_OK Shutdown cause should be WDGM
Step 6	**[SWC]** executes EcuM_ShutdownTarget operation SelectShutdownCause() to select shutdown cause DCM	**[SWC]** SelectShutdownCause() should return E_OK
Step 7	**[SWC]** executes EcuM_ShutdownTarget operation GetShutdownCause() to get shutdown cause	**[SWC]** GetShutdownCause() should return E_OK Shutdown cause should be DCM
Step 8	**[SWC]** executes EcuM_ShutdownTarget operation SelectShutdownCause() to select shutdown cause UNKNOWN	**[SWC]** SelectShutdownCause() should return E_OK
Step 9	**[SWC]** executes EcuM_ShutdownTarget operation GetShutdownCause() to get shutdown cause	**[SWC]** GetShutdownCause() should return E_OK Shutdown cause should be UNKNOWN
Step 10	**[SWC]** executes EcuM_ShutdownTarget operation SelectShutdownCause() to select shutdown cause CUSTOM_1	**[SWC]** SelectShutdownCause() should return E_OK
Step 11	**[SWC]** executes EcuM_ShutdownTarget operation GetShutdownCause() to get shutdown cause	**[SWC]** GetShutdownCause() should return E_OK Shutdown cause should be CUSTOM_1
Step 12	**[CP]** terminates SWC	-

Post-conditions	None

7 EcuM Additional Test Cases

7.1 General Test Objective and Approach

This Test Specification intends to cover the following EcuM features:

- EcuM_EnableWakeupSources And EcuM_DisableWakeupSources callouts
- RAM Integrity Check
- Wakeup Event Validation
- Shutdown Initiation
- EcuM_AlarmClock Port Interfaces

The tests use a test bench environment and Embedded Software Components that use these features.

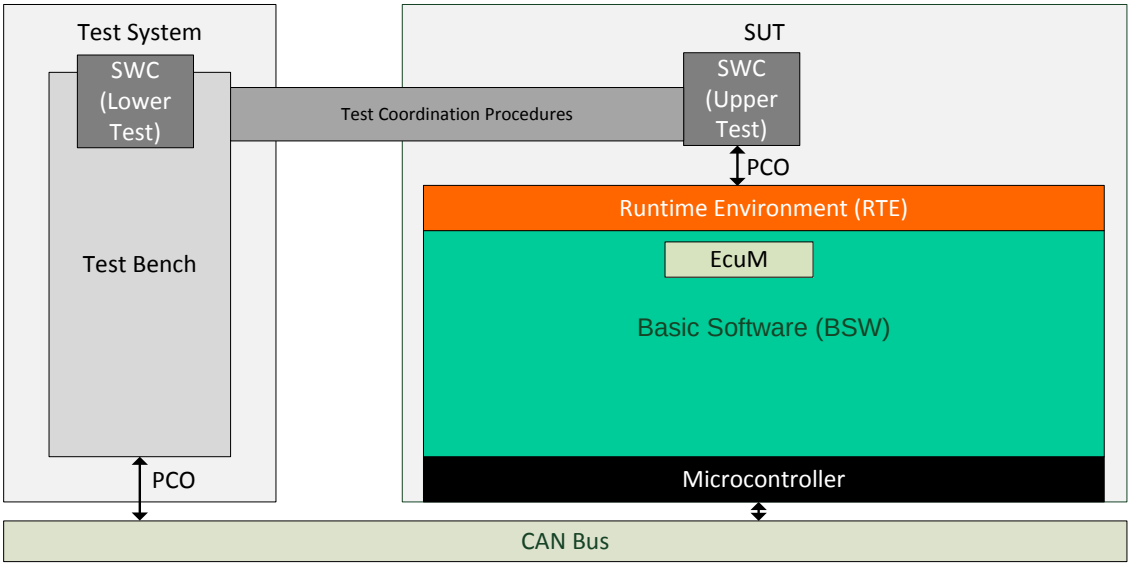
This test case document has been established to cover the mentioned features.

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

7.1.1 Test System

7.1.1.1 Overview on Architecture

The aim of this use case is to test the mentioned EcuM features of the EcuMFlex module.



The test system architecture consists of Test Bench that executes only test sequencer and gives actions request through Test coordination Procedures to embedded SWC.

7.1.1.2 Specific Requirements

Not Applicable.

7.1.1.3 Test Coordination Requirements

Not Applicable.

7.1.2 Test Configuration

This section describes sets of requirements on configuration.

No configuration files are provided, they need to be developed when the test suites is implemented.

7.1.2.1 Required ECU Extract of System Description Files

Minimum of one user is needed.

7.1.2.2 Required ECU Configuration Description Files

The section describes the common EcuC parameters between test cases that are required by the implementer of the test cases.

In BswM, these 2 modes shall be configured for all tests cases:

- Active Type Mode: Its actions objective should be to keep DUT active.
- Inactive Type Mode: Its actions objective should be to initiate the shutdown sequence.

7.1.2.3 Required Software Component Description Files

The SWC description and ports used should be fulfilling the interfaces and connections mentioned inside the test cases.

7.1.2.4 Mandatory vs. Customizable Parts

Mandatory parameters are listed in Tests Cases (see chapter 7.3 Test Cases).

Customizable parameters are (these values are test case independent):

- Dem configuration
- Initialization list of the BSW
- The different sleep modes
- The wakeup sources

7.1.3 Test Case Design

Not Applicable

7.2 Re-usable Test Steps

Not Applicable

7.3 Test Cases

7.3.1 [ATS_ECUM_01036] EcuM functionality for EcuM_EnableWakeupSources And EcuM_DisableWakeupSources callouts

Test Objective	EcuM functionality for EcuM_EnableWakeupSources And EcuM_DisableWakeupSources callouts		
ID	ATS_ECUM_01036	**AUTOSAR Releases**	4.0.3|4.2.2
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	ECUStateManager: SWS_EcuM_02918 ECUStateManager: SWS_EcuM_02546 ECUStateManager: SWS_EcuM_02922 ECUStateManager: SWS_EcuM_04084		
Requirements / Reference to Test Environment			
Configuration Parameters	EcuMwakeupSource = ECUM_WKSOURCE_POWER (source 0), ECUM_WKSOURCE_CAN (source 1), ECUM_WKSOURCE_LIN (source 2) EcuMDefaultState {ECUM_DEFAULT_SHUTDOWN_TARGET} = ECUM_STATE_SLEEP EcuMDefaultSleepModeRef = Reference to EcuMSleepMode which is configured for 'Halt' sequence. For EcuMSleepMode, 'EcuMWakeupSourceMask' should be configured to refer 'CAN' and 'LIN' wakeup sources. Callouts: EcuM_EnableWakeupSources = LinChannelEcuMWakeupSource, CanWakeupSourceRef (both CAN and LIN wake up sources shall be configured under this callout) EcuM_DisableWakeupSources = LinChannelEcuMWakeupSource, CanWakeupSourceRef (both CAN and LIN wake up sources shall be configured under this callout)		
Summary	To test the EcuM for executing the EcuM_EnableWakeupSources and EcuM_DisableWakeupSources callouts. To set the wakeup sources up for the next sleep mode, the ECU Manager module shall execute the user configured EcuM_EnableWakeupSources callout for the target sleep mode. The ECU Manager calls user configured EcuM_DisableWakeupSources callout to set the wakeup source(s) defined in the wakeupSource bitfield so that they are not be able to wake the ECU up.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM shall be in full communication state		
Main Test Execution			
Test Steps			**Pass Criteria**

Step 1	[SWC] Sends Active type requested mode through BswM_RequestMode to keep DUT active.	[SWC] BswM_RequestMode shall return with E_OK.
Step 2	[CP] Wait for 100 msec in DUT.	-
Step 4	[SWC] Sends Inactive type requested mode through BswM_RequestMode to initiate shutdown.	[SWC] EcuM_StateType shall enter into ECUM_STATE_PREP_SHUTDOWN state. Tester shall observe no frames. EcuM_EnableWakeupSources callout shall be invoked.
Step 5	[LT] Send a valid wakeup frame.	[SWC] EcuM_DisableWakeupSources callout shall be invoked.
Post-conditions	None	

7.3.2 [ATS_ECUM_01037] Provision For Ram Integrity Check

Test Objective	Provision For Ram Integrity Check		
ID	ATS_ECUM_01037	**AUTOSAR Releases**	4.0.3|4.2.2
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	ECUStateManager: SWS_EcuM_02863 ECUStateManager: SWS_EcuM_02961		
Requirements / Reference to Test Environment			
Configuration Parameters	EcuMwakeupSource = ECUM_WKSOURCE_POWER (source 0), ECUM_WKSOURCE_CAN (source 1), ECUM_WKSOURCE_LIN (source 2) EcuMDefaultState {ECUM_DEFAULT_SHUTDOWN_TARGET} = ECUM_STATE_SLEEP EcuMDefaultSleepModeRef = Reference to EcuMSleepMode which is configured for 'Halt' sequence. For EcuMSleepMode, 'EcuMWakeupSourceMask' should be configured to refer 'CAN' and 'LIN' wakeup sources. Callouts: EcuM_GenerateRamHash EcuM_CheckRamHash EcuM_ErrorHook		

Summary	To test the EcuM functionality for invoking the EcuM_GenerateRamHash callout where the system designer can place a RAM integrity check. The ECU Manager module shall invoke the user configured EcuM_GenerateRamHash callout before halting the microcontroller, and user configured EcuM_CheckRamHash callout after the processor returns from halt. User configured EcuM_ErrorHook callout is checked.	
Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM shall be in full communication state	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Sends Active type requested mode through BswM_RequestMode to keep DUT active.	[SWC] BswM_RequestMode shall return with E_OK.
Step 2	[CP] Wait for 100 msec in DUT.	-
Step 3	[SWC] Sends Inactive type requested mode through BswM_RequestMode to initiate shutdown.	[SWC] EcuM_StateType shall enter into ECUM_STATE_PREP_SHUTDOWN state. Tester shall observe no frames. EcuM_GenerateRamHash callout shall be invoked.
Step 4	[LT] Send a valid wakeup frame.	[SWC] EcuM_CheckRamHash callout shall be invoked. EcuM_ErrorHook callout shall not be invoked, indicating that RAM integrity test is successful.
Post-conditions	None	

7.3.3 [ATS_ECUM_01038] EcuM Functionality For Invoking Validation Protocol

Test Objective	EcuM Functionality For Invoking Validation Protocol		
ID	ATS_ECUM_01038	**AUTOSAR Releases**	4.0.3|4.2.2
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance			

Test Document		
Trace to SWS Item	ECUStateManager: SWS_EcuM_02975	
Requirements / Reference to Test Environment		
Configuration Parameters	EcuMwakeupSource = ECUM_WKSOURCE_POWER (source 0), ECUM_WKSOURCE_CAN (source 1) EcuMDefaultState {ECUM_DEFAULT_SHUTDOWN_TARGET} = ECUM_STATE_SLEEP EcuMDefaultSleepModeRef = Reference to EcuMSleepMode which is configured for 'Halt' sequence. For EcuMSleepMode, 'EcuMWakeupSourceMask' should be configured to refer CAN wakeup source. ECUM_WKSOURCE_CAN should be configured with EcuMValidationTimeout as 150 (msec). Callouts: EcuM_StartWakeupSources EcuM_CheckValidation	
Summary	To test EcuM Functionality for invoking Validation Protocol. The ECU Manager shall invoke wakeup validation only if required by configuration. If the validation protocol is not configured, then set wakeup event and send the request to validate wake up event EcuM_ValidateWakeupEvent. The ECU Manager shall execute the Wakeup Validation Protocol upon the Setting Wake up event function call Interaction of Wakeup Sources and the ECU Manager.	
Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state ComM shall be in full communication state	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Sends Active type requested mode through BswM_RequestMode to keep DUT active.	[SWC] BswM_RequestMode shall return with E_OK.
Step 2	[CP] Wait for 100 msec in DUT.	-
Step 3	[SWC] Sends Inactive type requested mode through BswM_RequestMode to initiate shutdown.	[SWC] EcuM_StateType shall enter into ECUM_STATE_PREP_SHUTDOWN state.
Step 4	-	[LT] Tester shall observe no CAN frames.
Step 5	[LT] Tester shall send valid wakeup frame.	[SWC] EcuM_StartWakeupSources callout shall be invoked for wakeupSource of

		CAN. EcuM_CheckValidation callout shall be invoked for wakeupSource of CAN.
Post-conditions	None	

7.3.4 [ATS_ECUM_01039] Shutdown Initiation When All User Requests Are Released

Test Objective	Shutdown Initiation When All User Requests Are Released		
ID	ATS_ECUM_01039	**AUTOSAR Releases**	4.0.3|4.2.2
Affected Modules	EcuM, BswM	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	BSWModeManager: SWS_BswM_00009 BSWModeManager: SWS_BswM_00035 BSWModeManager: SWS_BswM_00010 BSWModeManager: SWS_BswM_00012 BSWModeManager: SWS_BswM_00061 BSWModeManager: SWS_BswM_00013 BSWModeManager: SWS_BswM_00059 BSWModeManager: SWS_BswM_00014 BSWModeManager: SWS_BswM_00016 BSWModeManager: SWS_BswM_00015 ECUStateManager: SWS_EcuM_02181		
Requirements / Reference to Test Environment			
Configuration Parameters	BswMModeRequestPort = 3 User defined Configuration Reference: RunnableEntity_1(to request INITIATESHUTDOWN mode from BswM)		
Summary	To check the DUT for shutdown when all user requests are released. When all the user requests are released, the DUT shall proceed with shutdown.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized. EcuM shall be in POST RUN state.		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	[SWC] Invokes Rte_Write inside RunnableEntity_1 in order to request ACTIVE mode from BswM		[SWC] RTE_E_OK shall be returned

Step 2	[SWC] Invokes Rte_Mode in order to check the current state of EcuM	[SWC] Rte_Mode shall returns EcuM_StateType as ECUM_STATE_PREP_SHUTDOWN
Post-conditions	None	

7.3.5 [ATS_ECUM_01040] Services Of The Port Interface EcuM_AlarmClock

Test Objective	Services Of The Port Interface EcuM_AlarmClock		
ID	ATS_ECUM_01040	**AUTOSAR Releases**	4.0.3|4.2.2
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	ECUStateManager: SWS_EcuM_03013		
Requirements / Reference to Test Environment			
Configuration Parameters	time = Ex: 120 (EcuM_SetClock) = Ex: 10 (EcuM_SetRelWakeupAlarm) = Ex: 600 (EcuM_SetAbsWakeupAlarm) Add an Action list in BswM for wakeup notification to the SWC		
Summary	To check the services of the port interface EcuM_AlarmClock. The ECU State Manager module provides a client-server interface EcuM_AlarmClock which allows a SW-C to select its alarm relative to the current time (EcuM_SetRelWakeupAlarm), select its alarm to an absolute point in time (EcuM_SetAbsWakeupAlarm), cancel its alarm (EcuM_AbortWakeupAlarm), get the current time (EcuM_GetCurrentTime), get the absolute time in seconds of the next wakeup(EcuM_GetWakeupTime) and set EcuM Clock (EcuM_SetClock). These different services of the port interface EcuM_AlarmClock are tested in this test case.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized EcuM shall be in RUN state		
Main Test Execution			
Test Steps			**Pass Criteria**
Step 1	[SWC] Invokes Rte_Call in order to set EcuM Clock (Rte_Xxx_ SetClock)		[SWC] RTE_E_OK shall be returned
Step 2	[SWC] Invokes Rte_Call to select its alarm relative to the current time (Rte_		[SWC] RTE_E_OK shall be returned

	Xxx_SelectRelWakeupAlarm)	
Step 3	[SWC] Invokes Rte_Call to select its alarm to an absolute point in time (Rte_Xxx_ SelectAbsWakeupAlarm)	[SWC] RTE_E_OK shall be returned
Step 4	[SWC] Invokes Rte_Call to get the current time (Rte_Xxx_ GetCurrentTime)	[SWC] RTE_E_OK shall be returned The parameter “time” shall be updated with the current value of EcuM clock (time since battery connect).
Step 5	[SWC] Invokes Rte_Call to get the absolute time in seconds of the next wakeup (Rte_Xxx_ GetWakeupTime).	[SWC] RTE_E_OK shall be returned The parameter “time” shall be updated with the current value of the master alarm clock.
Step 6	[CP] Wait till the wakeup occurs	[SWC] Notification for the wakeup shall be called indicating a valid wakeup occured
Step 7	[SWC] Invokes Rte_Call to cancel its alarm (Rte_Xxx_ AbortWakeupAlarm)	[SWC] RTE_E_OK shall be returned
Post-conditions	None	

8 BswM Additional Test Cases

8.1 General Test Objective and Approach

This Test Specification intends to cover the following BswM features:

- BswM Mode Request in case of Immediate Processing
- BswM Mode Request in case of Deferred Processing
- Termination of Action List Execution
- Action Lists referring to other Action Lists
- Triggered and Conditional Execution (True and False Action Lists)

The tests use a test bench environment and Embedded Software Components that use these features.

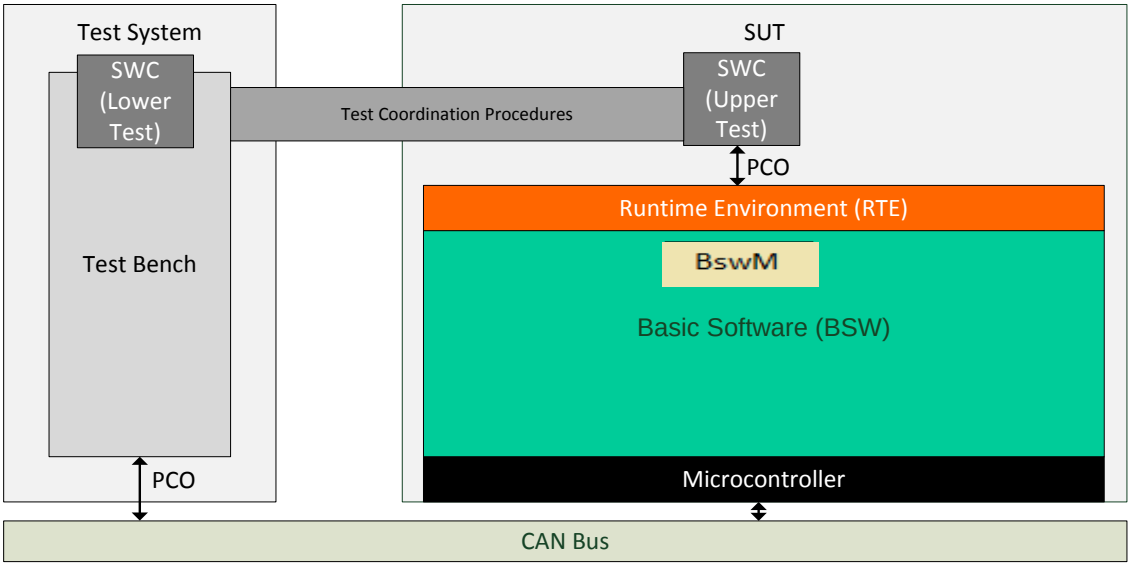
This test case document has been established to cover the mentioned features.

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

8.1.1 Test System

8.1.1.1 Overview on Architecture

The aim of this use case is to test the mentioned BswM features of the BswM module.



The test system architecture consists of Test Bench that executes only test sequencer and gives actions request through Test coordination Procedures to embedded SWC.

8.1.1.2 Specific Requirements

Not Applicable.

8.1.1.3 Test Coordination Requirements

Not Applicable.

8.1.2 Test Configuration

This section describes sets of requirements on configuration.

No configuration files are provided, they need to be developed when the test suites is implemented.

8.1.2.1 Required ECU Extract of System Description Files

Minimum of one user is needed.

8.1.2.2 Required ECU Configuration Description Files

The section describes the common EcuC parameters between test cases that are required by the implementer of the test cases.

No specific configurations other than those mentioned in every test case.

8.1.2.3 Required Software Component Description Files

The SWC description and ports used should be fulfilling the interfaces and connections mentioned inside the test cases.

8.1.2.4 Mandatory vs. Customizable Parts

Mandatory parameters are listed in Tests Cases (see chapter 8.3 Test Cases).

Customizable parameters are (these values are test case independent):

- Dem configuration
- Initialization list of the BSW
- The different sleep modes
- The wakeup sources

8.1.3 Test Case Design

Not Applicable

8.2 Re-usable Test Steps

Not Applicable

8.3 Test Cases

8.3.1 [ATS_ECUM_01041] Bswm Functionality To Call User Defined Functions When BswMModeRequestPort Is Configured For Immediate Processing

Test Objective	Bswm Functionality To Call User Defined Functions When BswMModeRequestPort Is Configured For Immediate Processing		
ID	ATS_ECUM_01041	**AUTOSAR Releases**	4.0.3|4.2.2
Affected Modules	BswM	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	BSWModeManager: SWS_BswM_00013 BSWModeManager: SWS_BswM_00040 BSWModeManager: SWS_BswM_00047		
Requirements / Reference to Test Environment			
Configuration Parameters	Config for BswM_RequestMode: BswMModeRequestPort = BswMGenericRequest BswMRequestProcessing = BSWM_IMMEDIATE BswMConditionType = BSWM_EQUALS BswMComMEnabled = TRUE BswMRuleInitState = BSWM_TRUE BswMActionListExecution = BSWM_TRIGGER BswMComMModeSwitch = ComMUser Config for BswMComMIndication: BswMModeRequestPort = BswMComMIndication BswMRequestProcessing = BSWM_IMMEDIATE BswMConditionType = BSWM_EQUALS BswMComMEnabled = TRUE BswMRuleInitState = BSWM_TRUE BswMActionListExecution = BSWM_TRIGGER BswMUserCallout = App_ComM_CurrentMode		
Summary	To test the BSWM functionality to call BswMUserCallout when BswMModeRequestPort is configured for immediate processing. Configure two Mode Request Ports, one as a Generic mode Request for requesting a mode change from Application to ComM and another as a Mode Request for BswMComMIndication. Configured user callout shall be invoked immediately when ComM shall notify BswM of its current mode. Within the user callout, the current ComM mode shall be read.		
Needed Adaptation to other Releases			
Pre-conditions	DUT shall be initialized		
Main Test Execution			
Test Steps		**Pass Criteria**	
Step 1	[SWC] Call BswM_RequestMode request with	[SWC] App_ComM_CurrentMode shall be	

	requested_mode as COMM_FULL_COMMUNICATION to keep ComM in Full communication.	invoked for ComM which indicates RequestedMode as COMM_FULL_COMMUNICATION.
Step 2	-	[LT] Message traffic starts in the bus
Step 3	[SWC] Call BswM_RequestMode request with requested_mode as COMM_NO_COMMUNICATION to keep ComM in No communication.	[SWC] App_ComM_CurrentMode shall be invoked for ComM which indicates RequestedMode as COMM_NO_COMMUNICATION.
Step 4	-	[LT] Message traffic stopped in the bus
Post-conditions	None	

8.3.2 [ATS_ECUM_01042] Current State Indication From CanSM For Deferred Processing

Test Objective	Current State Indication From CanSM For Deferred Processing		
ID	ATS_ECUM_01042	**AUTOSAR Releases**	4.0.3|4.2.2
Affected Modules	BswM	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	BSWModeManager: SWS_BswM_00014 BSWModeManager: SWS_BswM_00049		
Requirements / Reference to Test Environment			
Configuration Parameters	Config for BswM_RequestMode: BswMModeRequestPort = BswMGenericRequest BswMRequestProcessing = BSWM_IMMEDIATE BswMConditionType = BSWM_EQUALS BswMComMEnabled = TRUE BswMRuleInitState = BSWM_TRUE BswMActionListExecution = BSWM_TRIGGER BswMComMModeSwitch = ComMUser Config for BswMCanSMIndication: BswMModeRequestPort = BswMCanSMIndication BswMRequestProcessing = BSWM_ DEFERRED BswMConditionType = BSWM_EQUALS BswMCanSMEnabled = TRUE BswMRuleInitState = BSWM_TRUE BswMActionListExecution = BSWM_TRIGGER		

	BswMUserCallout (for state change) = App_CanSM_CurrentState
Summary	To test the Current State Indication from CanSM for Deferred processing. Configure two Mode Request Ports, one as a Generic mode Request for requesting a mode change from Application to ComM and another as a Mode Request for BswMCanSMIndication (ComMUser group should contain CAN channel which shall be used in here). Mode Switch Indications originating from the CanSM go through the BswM for further propagation to the SW-Cs.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be in full communication

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Send BswM_RequestMode request for changing requested_mode to COMM_NO_COMMUNICATION to keep Com in No communication.	[SWC] App_CanSM_CurrentState shall be invoked for ComM which indicates CurrentState as CANSM_BSWM_NO_COMMUNICATION
Step 2	[SWC] Send BswM_RequestMode request for changing requested_mode to COMM_FULL_COMMUNICATION to keep Com in Full communication.	[SWC] App_CanSM_CurrentState shall be invoked ComM which indicates RequestedMode as CANSM_BSWM_FULL_COMMUNICATION
Post-conditions	None	

8.3.3 [ATS_ECUM_01043] Termination Of Action List Execution Due To Error In One Action

Test Objective	Termination Of Action List Execution Due To Error In One Action		
ID	ATS_ECUM_01043	**AUTOSAR Releases**	4.0.3|4.2.2
Affected Modules	BswM	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	BSWModeManager: SWS_BswM_00055 BSWModeManager: SWS_BswM_00047		
Requirements / Reference to Test Environment			
Configuration	ComMUser = 1		

Parameters	BswMModeRequestPort = BswMComMIndication (port 0), BswMCanSMIndication (port 1), BswMLinSMIndication (port 2) BswMConditionType = BSWM_EQUALS BswMRuleInitState = BSWM_TRUE BswMActionListExecution = BSWM_TRIGGER BswMUserCallout = App_CanSM_CurrentState (for user 1), App_LinSM_CurrentState(for user 2), App_ComM_ComMode(for user 3) BswMAbortOnFail = TRUE
Summary	To test the termination of action list execution due to error in one action. Configure 3 actions for BswMComMIndication, BswMCanSMIndication and BswMLinSMIndication. Map these to action lists. BswMComMIndication is mapped to user 3. User 3 is configured in RTE not in ComM. Because of error, it is not going to execute the remaining action lists.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state

Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] BswM_RequestMode request with requested_mode as COMM_FULL_COMMUNICATION to keep Com in Full communication.	[SWC] Return with RTE_E_NOT_OK. No frames shall be observed on bus. Hint: BswMComMIndication is configured in RTE but not in ComM
Step 2	[SWC] ComM_GetCurrentComMode request to get current comMode.	[SWC] ComM_GetCurrentComMode shall return E_OK. App_ComM_ComMode shall not be invoked.
Post-conditions	None	

8.3.4 [ATS_ECUM_01044] Action List having reference to mode arbitration rule and other action list

Test Objective	Action List having reference to mode arbitration rule and other action list		
ID	ATS_ECUM_01044	**AUTOSAR Releases**	4.0.3|4.2.2
Affected Modules	BswM	**State**	reviewed
Trace to Requirement on Acceptance			

Test Document	
Trace to SWS Item	BSWModeManager: SWS_BswM_00018 BSWModeManager: SWS_BswM_00019 BSWModeManager: SWS_BswM_00067 BSWModeManager: SWS_BswM_00062
Requirements / Reference to Test Environment	
Configuration Parameters	Configuration for Base Action List: ModeRequestPort=GenericRequest_AL RequestProcessing=BSWM_IMMEDIATE ModeRequesterId=6 ModeCondition=ModeCondition_AL ConditionType=BSWM_EQUALS ConditionValue -> BswMode -> BswRequestedMode=BSWM_MODE_AL RuleTrueActionList=ActionList_AL_RuleTrueCond ActionListExecution=BSWM_TRIGGER ActionListItem -> UserCalloutFunction=UserCallout_AL_1 ActionListItem -> Rule=Rule_AL_NestedRule ActionListItem -> ActionList=ActionList_AL_NestedAL Configuration for Nested Rule within Base Action List: ModeRequestPort=GenericRequest_AL_NestedRule RequestProcessing=BSWM_IMMEDIATE ModeRequesterId=5 RuleInitState=BSWM_FALSE NestedExecutionOnly= BSWM_TRUE ModeCondition=ModeCondition_AL_NestedRule ConditionType=BSWM_NOT_EQUALS ConditionValue -> BswMode -> BswRequestedMode=BSWM_MODE_AL_NESTEDRULE RuleFalseActionList=ActionList_AL_RuleFalseCond ActionListExecution=BSWM_CONDITION ActionListItem -> UserCalloutFunction=UserCallout_AL_NestedRule_2
Summary	To test the functionality of BswM when an action list is referring to a mode arbitration rule and other action list. An action list may contain links to other action lists that BswM shall include in the execution. An action list may also include links to mode arbitration rules that BswM shall evaluate within the scope of the execution of the current action list. If a rule is included in an action list as specified above, any action list execution resulting from that evaluation shall be executed by BswM before it continues to execute the original action list. Action lists associated with rules evaluated in the context of the mode arbitration request shall be executed by BswM immediately when triggered by the mode arbitration, and not be deferred to the main function execution.
Needed Adaptation to other Releases	
Pre-conditions	DUT shall be initialized
Main Test Execution	
Test Steps	**Pass Criteria**

Step 1	[SWC] Send BswM_RequestMode request for changing requested_mode to BSWM_MODE_AL_NESTEDRULE to execute mode arbitration rule.	[SWC] Observe BswMUserCallout_AL_NestedRule_2 is NOT invoked as it is a Nested Rule for an Action List (AL) whose requested mode condition is not satisfied.
Step 2	[SWC] Send BswM_RequestMode request for changing requested_mode to BSWM_MODE_AL to execute mode arbitration rule and other action list.	[SWC] User callout BswMUserCallout_AL_1 shall be invoked. BswMUserCallout_AL_NestedRule_2 shall be invoked. BswMUserCallout_AL_NestedAL_3 shall be invoked.
Post-conditions	None	

8.3.5 [ATS_ECUM_01045] True and False action list configured for Triggered execution

Test Objective	True and False action list configured for Triggered execution		
ID	ATS_ECUM_01045	**AUTOSAR Releases**	4.0.3|4.2.2
Affected Modules	BswM	**State**	reviewed
Trace to Requirement on Acceptance Test Document			
Trace to SWS Item	BSWModeManager: SWS_BswM_00011 BSWModeManager: SWS_BswM_00023		
Requirements / Reference to Test Environment			
Configuration Parameters	BswMModeRequestPort = BswMGenericRequest_AL_Eval, BswMRequestProcessing = BSWM_IMMEDIATE, BswMModeRequesterId = 6. BswMModeCondition = BswMModeCondition_AL_Trigger, BswMConditionType = BSWM_EQUALS, BswMConditionValue -> BswMBswMode -> BswMBswRequestedMode = BSWM_MODE_TRIGGER. BswMRule = BswMRule_AL_Trigger, BswMRuleInitState = BSWM_FALSE. Action List for TRUE condition for RULE: BswMRuleTrueActionList = BswMActionList_AL_Trigger_True, BswMActionListExecution = BSWM_TRIGGER, BswMActionListItem -> BswMUserCalloutFunction = BswMUserCallout_AL_Condition_True. Action List for FALSE condition for RULE:		

	BswMRuleFalseActionList = BswMActionList_AL_Condition_False, BswMRuleFalseActionList = BswMActionList_AL_Trigger_False, BswMActionListExecution = BSWM_TRIGGER, BswMActionListItem -> BswMUserCalloutFunction = BswMUserCallout_AL_Condition_False, BswMRequestedMode = BSWM_MODE_INVALID (mode 1), BSWM_MODE_CONDITION (mode 2), BSWM_MODE_TRIGGER (mode 3)	
Summary	To test the functionality of BswM for True and False action list configured for Triggered execution. If a True action list is configured for triggered execution the BswM shall only execute it when the evaluation of the corresponding rule changes from False to True. If a False action list is configured for triggered execution the BswM shall only execute it when the evaluation of the corresponding rule changes from True to False.	
Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Send BswM_RequestMode request for changing requested_mode to BSWM_MODE_INVALID indicating Invalid mode.	[SWC] BswMUserCallout_AL_Condition_True shall not be invoked. BswMUserCallout_AL_Condition_False shall not be invoked.
Step 2	[SWC] Send BswM_RequestMode request for changing requested_mode to BSWM_MODE_TRIGGER indicating Trigger mode.	[SWC] BswMUserCallout_AL_Condition_True shall be invoked. BswMUserCallout_AL_Condition_False shall not be invoked.
Step 3	[SWC] Send BswM_RequestMode request for changing requested_mode to BSWM_MODE_CONDITION indicating Condition mode.	[SWC] BswMUserCallout_AL_Condition_True shall not be invoked. BswMUserCallout_AL_Condition_False shall be invoked.
Post-conditions	None	

8.3.6 [ATS_ECUM_01046] True and False action list configured for Conditional execution

Test Objective	True and False action list configured for Conditional execution		
ID	ATS_ECUM_01046	**AUTOSAR Releases**	4.0.3|4.2.2
Affected Modules	BswM	**State**	reviewed
Trace to Requirement			

on Acceptance Test Document		
Trace to SWS Item	BSWModeManager: SWS_BswM_00115 BSWModeManager: SWS_BswM_00116	
Requirements / Reference to Test Environment		
Configuration Parameters	BswMModeRequestPort = BswMGenericRequest_AL_Eval, BswMRequestProcessing = BSWM_IMMEDIATE, BswMModeRequesterId = 6. BswMModeCondition = BswMModeCondition_AL_Condition, BswMConditionType = BSWM_EQUALS, BswMConditionValue -> BswMBswMode -> BswMBswRequestedMode = BSWM_MODE_CONDITION. BswMRule = BswMRule_AL_Condition, BswMRuleInitState = BSWM_FALSE. Action List for TRUE condition for RULE: BswMRuleTrueActionList = BswMActionList_AL_Condition_True, BswMActionListExecution = BSWM_CONDITION, BswMActionListItem -> BswMUserCalloutFunction = BswMUserCallout_AL_Condition_True. BwMRuleTrueActionList = BswMActionList_AL_Condition_True Action List for FALSE condition for RULE: BswMRuleFalseActionList = BswMActionList_AL_Condition_False, BswMActionListExecution = BSWM_CONDITION, BswMActionListItem -> BswMUserCalloutFunction = BswMUserCallout_AL_Condition_False, BswMRequestedMode1 = BSWM_MODE_INVALID (mode 1), BSWM_MODE_CONDITION (mode 2), BSWM_MODE_TRIGGER (mode 3).	
Summary	To test the functionality of BswM for True and False action list configured for Conditional execution. If a True action list is configured for conditional execution the BswM shall execute it every time the corresponding rule is evaluated to True. If a False action list is configured for conditional execution the BswM shall execute it every time the corresponding rule is evaluated to False.	
Needed Adaptation to other Releases		
Pre-conditions	DUT shall be initialized EcuM module shall be in RUN state	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	[SWC] Call BswM_RequestMode request for changing requested_mode to BSWM_MODE_INVALID indicating Invalid mode.	[SWC] BswMUserCallout_AL_Condition_True shall not be invoked. BswMUserCallout_AL_Condition_False shall be invoked.
Step 2	[SWC] Call BswM_RequestMode request for changing requested_mode to	[SWC] BswMUserCallout_AL_Condition_True

	BSWM_MODE_CONDITION indicating Condition mode.	shall be invoked. BswMUserCallout_AL_Condition_False shall not be invoked.
Step 3	[SWC] Call BswM_RequestMode request for changing requested_mode to BSWM_MODE_TRIGGER indicating Trigger mode.	[SWC] BswMUserCallout_AL_Condition_True shall not be invoked. BswMUserCallout_AL_Condition_False shall be invoked.
Post-conditions	None	