Document Title	Acceptance Test Specification of Ecu Mode Management
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	665
Document Classification	Standard
Document Status	Final
Part of AUTOSAR Release	1.0.0

Document Change History		
Release	**Changed by**	**Change Description**
1.0.0	AUTOSAR Release Management	Initial release, including test suites on • RS_BRF_01488 – EcuM Current Mode • RS_BRF_01488 – EcuM State Request • RS_BRF_02152 – EcuM Boot Target • RS_BRF_02152 – EcuM Shutdown Target

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
NM	Network Management
PCO	Point of Control and Observation
PDU	Protocol Data Unit
RfC	Request for Change
Rx	Reception
SUT	System 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.

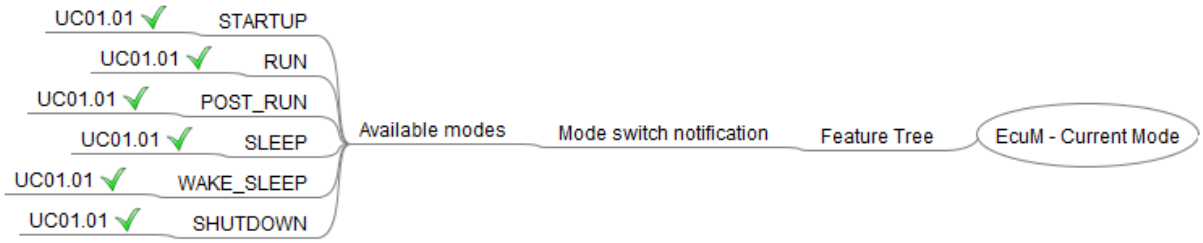
3 RS_BRF_01488 – EcuM Current Mode

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:

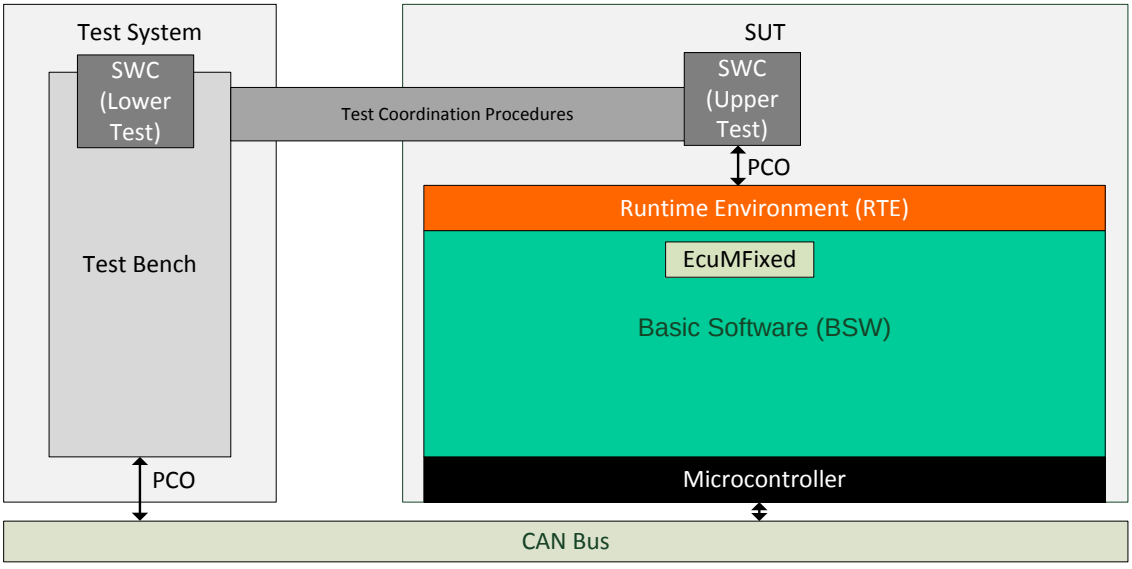


This specification gives the description of required tests environments (test bench, uses case, arxml files) and detailed tests cases for executing tests.

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.

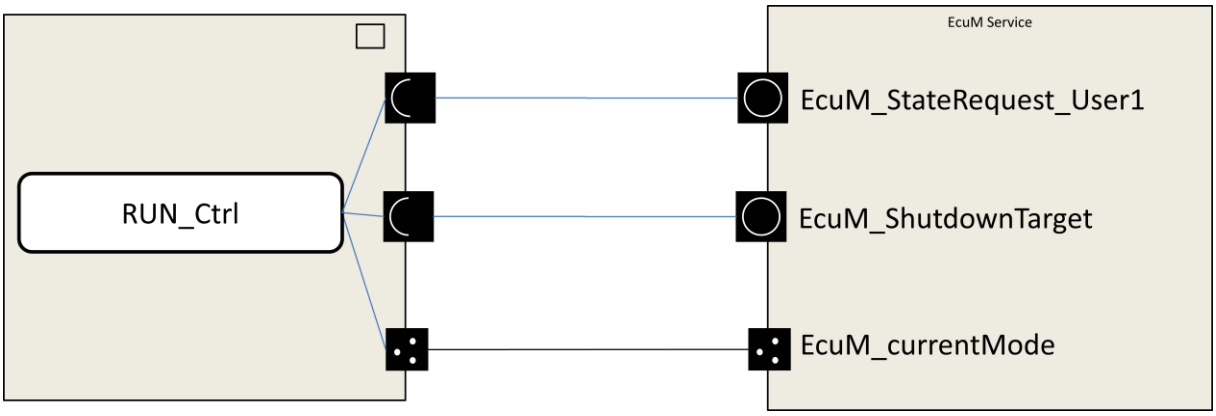
Use Case UC01.01:

- 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**	3.2.1|3.2.2|4.0.3|4.1.1
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00037		
Trace to R4.1.1 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		

	○ 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	None
Pre-conditions	At Ecu Startup, the BswM activates the Com Channel used by ATF.
Main Test Execution	
Test Steps	**Pass Criteria**
Step 1	TCP: restart SUT Mode notification must indicate a change in STARTUP mode
Step 2	TCP: Wait EcuM to enter RUN Mode notification must indicate a change in RUN mode
Step 3	RUN_Ctrl: query mode using EcuM_CurrentMode() Check that currentMode is RUN
Step 4	RUN_Ctrl: executes EcuM_StateRequest operation RequestRUN() for User 1
Step 5	RUN_Ctrl: query mode using EcuM_CurentMode() check that currentMode is RUN
Step 6	RUN_Ctrl: executes EcuM_StateRequest operation RequestPOSTRUN() for User 1
Step 7	RUN_Ctrl: query mode using EcuM_CurrentMode() Check that currentMode is RUN
Step 8	RUN_Ctrl: executes EcuM_StateRequest operation ReleaseRUN() for User 1 Mode notification must indicate a change in POSTRUN mode
Step 9	RUN_Ctrl: query mode using EcuM_CurrentMode() Check that currentMode is POSTRUN
Step 10	RUN_Ctrl: executes EcuM_StateRequest operation ReleasePOSTRUN() for User 1
Step 11	RUN_Ctrl: 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	RUN_Ctrl: executes EcuM_StateRequest operation RequestPOST_RUN() for User 1 Mode notification must indicate a change in SLEEP mode
Step 13	TCP: SUT is woken up by TTII Mode notification must indicate a

		change in WAKE_SLEEP mode
Step 14	TCP: wake SUT by wakeup source identified to enter RUN	Mode Notification must indicate a change in RUN
Step 15	RUN_Ctrl: executes EcuM_ShutdownTarget operation SelectShutdownTarget() to set shutdownTarget to ECUM_STATE_OFF	
Step 16	RUN_Ctrl: Release Ecu (by waiting exit from SelfRun or request no communication according to ATF implementation)	
Step 17	TCP: waits 5 seconds	ECU is shutdown
Post-conditions	None	

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

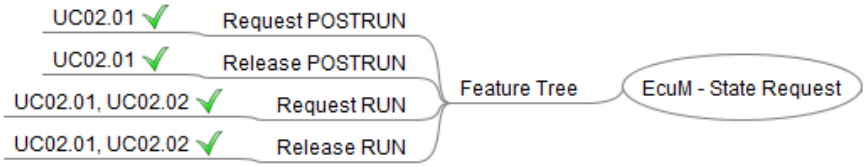
Test Objective	Getting the current mode of EcuMFixed module without POSTRUN state		
ID	ATS_ECUM_00244	**AUTOSAR Releases**	3.2.1|3.2.2|4.0.3|4.1.1
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00037		
Trace to R4.1.1 Item	ECUStateManagerFixed: 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 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		
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 ○ 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
Needed Adaptation to other Releases	None
Pre-conditions	At Ecu Startup, the BswM activates the Com Channel used by ATF.
Main Test Execution	
Test Steps	**Pass Criteria**
Step 1	TCP: restart SUT Mode notification must indicate a change in STARTUP mode
Step 2	TCP: Wait EcuM to enter RUN Mode notification must indicate a change in RUN mode
Step 3	RUN_Ctrl: query mode using EcuM_CurrentMode() Check that currentMode is RUN
Step 4	RUN_Ctrl: executes ComM_UserRequest operation RequestComMode(NO_COMMUNICATION) to inactivate the ATF communication and allow ECU to go in Sleep mode (no other active user) Mode notification must indicate a change in SLEEP mode
Step 5	RUN_Ctrl: On Enter Sleep mode, query mode using EcuM_CurrentMode() Check that currentMode is SLEEP
Step 6	TCP: SUT is woken up by TTII Mode notification must indicate a change in WAKE_SLEEP mode
Step 7	TCP: wake SUT by wakeup source identified to enter RUN Mode Notification must indicate a change in RUN
Step 8	RUN_Ctrl: executes EcuM_ShutdownTarget operation SelectShutdownTarget() to set shutdownTarget to ECUM_STATE_OFF
Step 9	RUN_Ctrl: Release Ecu (by waiting exit from SelfRun or request no communication according to ATF implementation)
Step 10	TCP: waits 5 seconds Mode notification must indicate a change in SHUTDOWN mode
Post-conditions	None

4 RS_BRF_01488 – EcuM State Request

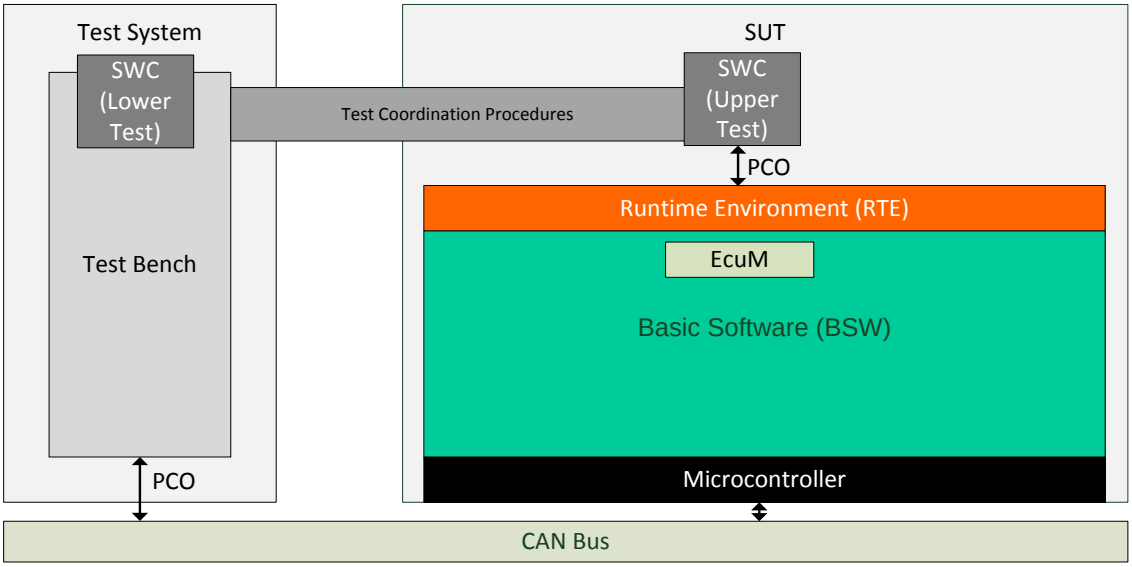
4.1 General Test Objective and Approach

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

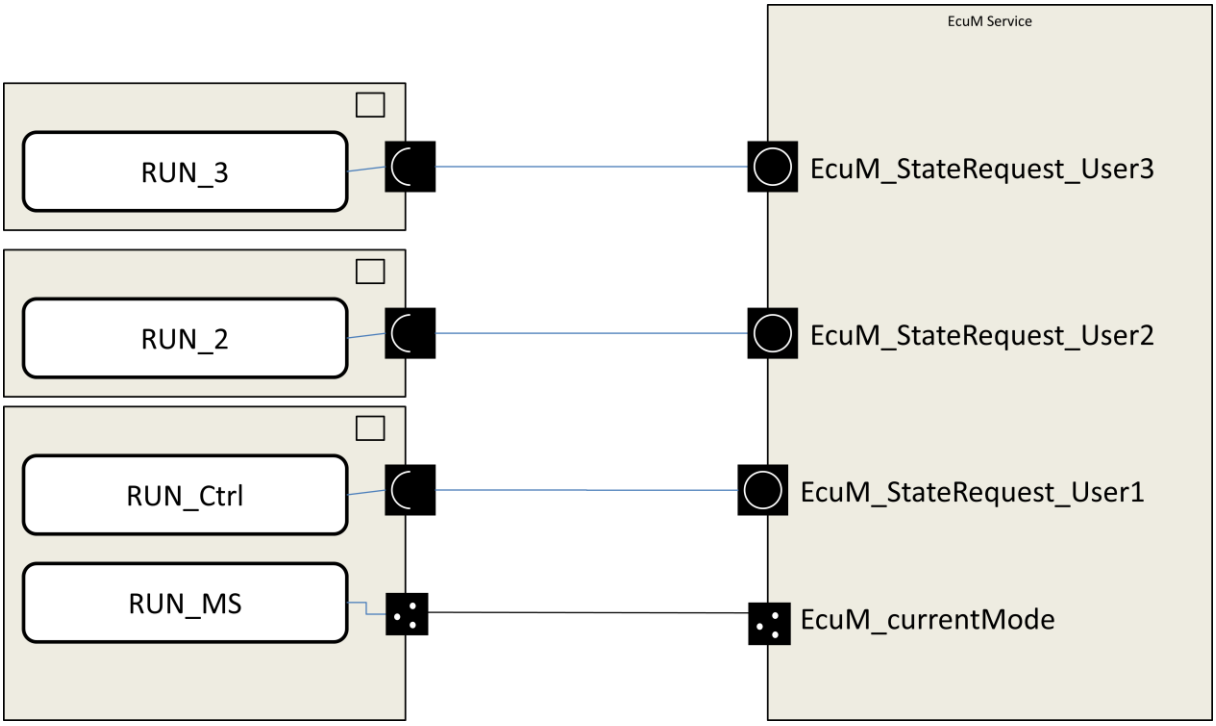
The section describes the SWC-D that are required by the implementer of the test cases.

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
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00037		
Trace to R4.1.1 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 Adaptation to other Releases	None
Pre-conditions	None
Main Test Execution	
Test Steps	**Pass Criteria**
Step 1	TCP: starts RUN_Ctrl
Step 2	RUN_Ctrl: executes EcuM_StateRequest operation RequestRUN() → EcuM_RequestRUN() should return E_OK
Step 3	TCP: wait 10 seconds → SUT should not shutdown
Step 4	TCP: Stop sending messages on the bus including "Network Management" message.
Step 5	TCP: Wait for (CanNmTimeoutTime + CanNmWaitBusSleepTime) Seconds *Note, Add a jitter to the configured time*
Step 6	TCP: executes ComM_UserRequest operation GetCurrentComMode() → The service should return E_OK ComMode should be COMM_NO_COMMUNICATION
Step 7	RUN_Ctrl: executes EcuM_StateRequest operation ReleaseRUN() → EcuM_ReleaseRUN() should return E_OK
Step 8	TCP: wait until SUT is shutdown
Step 9	TCP: wakes up SUT
Step 10	TCP: waits 4 seconds → SUT should NOT shutdown
Step 11	TCP: waits 1 seconds → 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
Affected Modules	EcuM, DET	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00037		
Trace to R4.1.1 Item	ECUStateManagerFixed: SWS_EcuM_00819 ECUStateManagerFixed: SWS_EcuM_00820 ECUStateManagerFixed: SWS_EcuMf_0030		
Requirements	Configuration use case : UC02.01		

/ Reference to Test Environment		
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	TCP: starts RUN_Ctrl	
Step 2	RUN_Ctrl: executes EcuM_StateRequest operation RequestRUN	RequestRUN should return E_OK
Step 3	RUN_Ctrl: executes EcuM_StateRequest operation RequestPOSTRUN	RequestPOSTRUN should return E_OK
Step 4	TCP: Stop sending messages on the bus including "Network Management" message.	
Step 5	TCP: Wait for (CanNmTimeoutTime + CanNmWaitBusSleepTime) Seconds *Note: Add a jitter to the configured time*	
Step 6	TCP: call ComM Service ComM_UserRequest operation GetCurrentComMode()	GetCurrentComMode should return E_OK ComMode should be COMM_NO_COMMUNICATION
Step 7	RUN_Ctrl: executes EcuM_StateRequest operation ReleaseRUN()	ReleaseRUN should return E_OK
Step 8	TCP: waits 10 seconds	SUT should not shutdown
Step 9	RUN_Ctrl: executes EcuM_StateRequest operation ReleasePOSTRUN()	ReleasePOSTRUN should return E_OK
Step 10	TCP: waits until SUT is shutdown	
Step 11	TCP: start SUT	
Step 12	TCP: start RUN_Ctrl	
Step 13	RUN_Ctrl: executes EcuM_StateRequest operation RequestPOSTRUN()	RequestPOSTRUN should return E_OK
Step 14	TCP: Stop sending messages on the bus including "Network Management" message.	

Step 15	TCP: Wait for (CanNmTimeoutTime + CanNmWaitBusSleepTime) Seconds *Note: Add a jitter to the configured time*	
Step 16	TCP: call ComM Service ComM_UserRequest operation GetCurrentComMode()	GetCurrentComMode should return E_OK ComMode should be COMM_NO_COMMUNICATION
Step 17	RUN_Ctrl: executes EcuM_StateRequest operation ReleasePOSTRUN()	ReleasePOSTRUN should return E_OK
Step 18	TCP: waits 4 seconds	SUT should not shutdown
Step 19	TCP: waits 1 second	SUT should shutdown
Step 20	TCP: start SUT	
Step 21	RUN_Ctrl: execute EcuM_StateRequest operation RequestPOSTRUN()	RequestPOSTRUN should return E_OK
Step 22	RUN_Ctrl: execute EcuM_StateRequest operation RequestPOSTRUN()	RequestPOSTRUN should return E_NOT_OK
Step 23	RUN_Ctrl: executes EcuM_StateRequest operation ReleasePOSTRUN()	ReleasePOSTRUN should return E_OK
Step 24	RUN_Ctrl: executes EcuM_StateRequest operation ReleasePOSTRUN()	ReleasePOSTRUN should return E_NOT_OK
Step 25	RUN_Ctrl: execute EcuM_StateRequest operation RequestRUN()	RequestRUN should return E_OK
Step 26	RUN_Ctrl: execute EcuM_StateRequest operation RequestRUN()	RequestRUN should return E_NOT_OK
Step 27	RUN_Ctrl: execute EcuM_StateRequest operation ReleaseRUN()	ReleaseRUN should return E_OK
Step 28	RUN_Ctrl: execute EcuM_StateRequest operation ReleaseRUN()	ReleaseRUN should return E_NOT_OK
Step 29	TCP: terminate RUN_Ctrl	
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
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00025 ATR: ATR_ATR_00037		
Trace to R4.1.1	ECUStateManagerFixed: SWS_EcuM_00749		

Item	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	TCP: starts RUN_Ctrl, RUN_2, RUN_3	
Step 2	RUN_Ctrl: call EcuM_StateRequest operation RequestRUN()	RequestRUN should return RTE_E_OK EcuM ModeSwitch port shall have the value RUN
Step 3	RUN_2: call EcuM_StateRequest operation RequestRUN()	RequestRUN should return RTE_E_OK EcuM ModeSwitch port shall have the value RUN
Step 4	RUN_3: call EcuM_StateRequest operation RequestRUN()	RequestRUN should return RTE_E_OK EcuM Mode Swith port shall have the value RUN
Step 5	TCP: wait 10 seconds	SUT should not shutdown
Step 6	RUN_2: call EcuM_StateRequest operation RequestPOSTRUN()	RequestPOSTRUN should return RTE_E_OK

		EcuM Switch port shall keep the value RUN and no mode switch occurs
Step 7	TCP: wait 10s	EcuM Switch Port shall keep the value RUN and no mode switch occurs
Step 8	RUN_1: call EcuM_StateRequest operation RequestPOSTRUN()	RequestPOSTRUN should return RTE_E_OK EcuM Switch port shall keep the value RUN and no mode switch occurs
Step 9	TCP: wait 10s	SUT should not shutdown
Step 10	RUN_3: call EcuM_StateRequest operation RequestPOSTRUN()	RequestPOSTRUN should return RTE_E_OK EcuM Switch port shall return the value RUN and no mode switch occurs
Step 11	TCP: Stop sending messages on the bus including "Network Management" message.	
Step 12	TCP: Wait for (CanNmTimeoutTime + CanNmWaitBusSleepTime) Seconds *Note: Add a jitter to the configured time*	
Step 13	TCP: call ComM Service ComM_UserRequest operation GetCurrentComMode()	GetCurrentComMode should return E_OK Current Com Mode should be COMM_NO_COMMUNICATION
Step 14	RUN_Ctrl: call EcuM_StateRequest operation ReleaseRUN() RUN_2: call EcuM_StateRequest operation ReleaseRUN() RUN_3: call EcuM_StateRequest operation ReleaseRUN()	
Step 15	TCP: Wait 10s	SUT should not shutdown
Step 16	RUN_Ctrl: call EcuM_StateRequest operation ReleasePOSTRUN() RUN_2: call EcuM_StateRequest operation ReleasePOSTRUN() RUN_3: call EcuM_StateRequest operation ReleasePOSTRUN()	
Step 17	TCP: wait 10s	SUT should shutdown
Step 18	LT: Send any frame on the bus to wake-up the ECU/SUT	SUT should wake-up
Step 19	TCP: Restart RUN_Ctrl, RUN_2, RUN_3	

Step 20	RUN_Ctrl: call EcuM_StateRequest operation RequestRUN()	EcuM Switch port should return the value RUN
Step 21	RUN_Ctrl: call EcuM_StateRequest operation RequestPOSTRUN()	EcuM Switch port shall return the value RUN and no mode switch occurs
Step 22	TCP: wait 2s	SUT should not shutdown
Step 23	RUN_2: call EcuM_StateRequest operation RequestRUN()	EcuM Switch port shall return the value RUN and no mode switch occurs
Step 24	RUN_Ctrl : call EcuM_StateRequest operation ReleasePOSTRUN()	EcuM Switch Port should return the value RUN
Step 25	TCP: Stop sending messages on the bus including "Network Management" message.	
Step 26	TCP: Wait for (CanNmTimeoutTime + CanNmWaitBusSleepTime) Seconds *Note: Add a jitter to the configured time*	
Step 27	TCP: call ComM_UserRequest operation GetCurrentComMode()	GetCurrentComMode should return E_OK ComMode should be COMM_NO_COMMUNICATION
Step 28	RUN_Ctrl: call EcuM_StateRequest operation ReleaseRUN() RUN_2: call EcuM_StateRequest operation ReleaseRUN()	
Step 29	TCP: wait 10s	SUT should shutdown
Post-conditions	None	

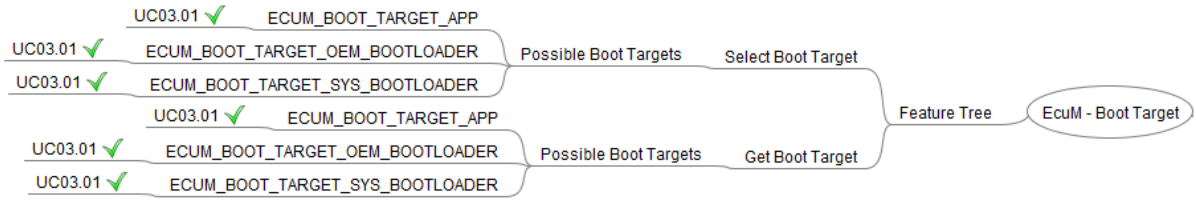
5 RS_BRF_02152 – EcuM Boot Target

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:

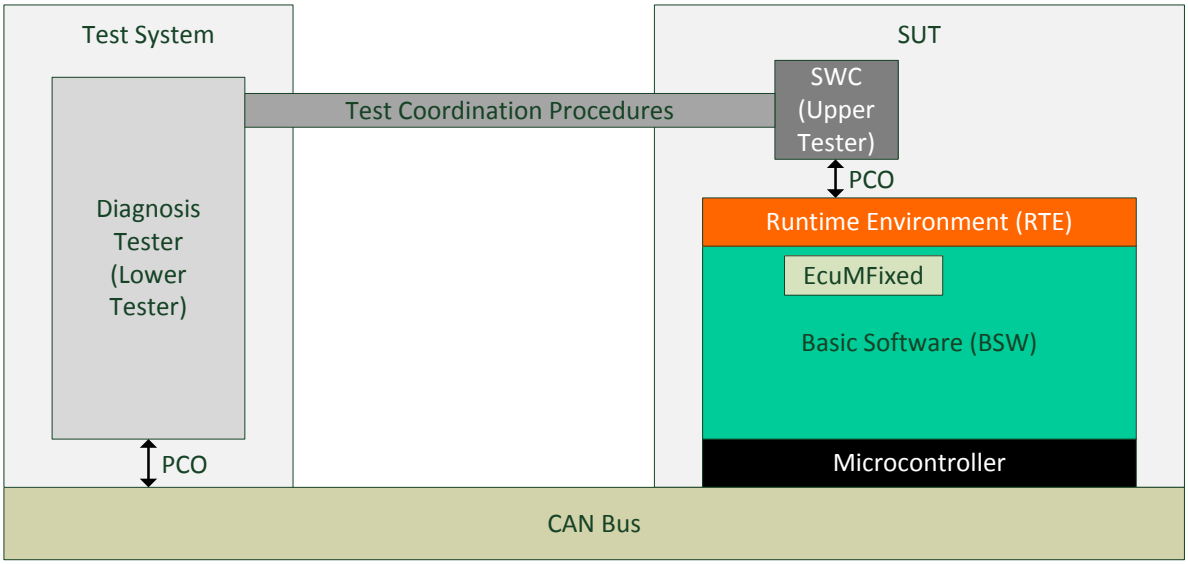


This specification gives the description of required tests environments (test bench, uses case, arxml files) and detailed tests cases for executing tests.

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:

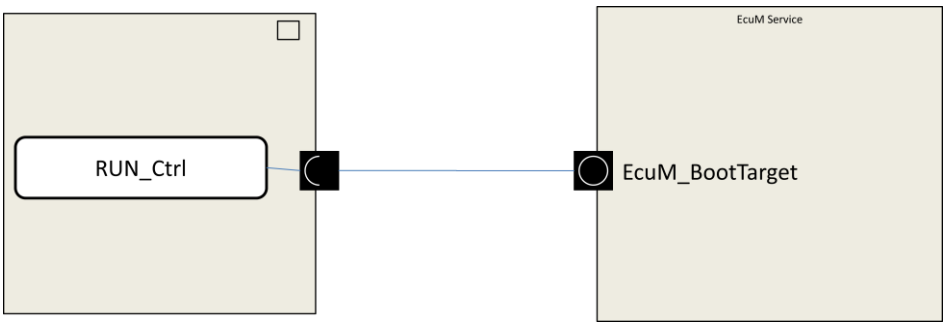
➔ EcuMFixed Bsw component

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
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00036		
Trace to R4.1.1 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	TCP: start RUN_Ctrl		
Step 2	RUN_Ctrl: executes EcuM_StateRequest operation GetBootTarget() to get boot target		GetBootTarget() should return E_OK Boot target should be ECUM_BOOT_TARGET_APP
Step 3	RUN_Ctrl: executes EcuM_StateRequest		SelectBootTarget() should return

	operation SelectBootTarget() to set boot target to ECUM_BOOT_TARGET_OEM_BOOTLOADER	E_OK
Step 4	RUN_Ctrl: executes EcuM_StateRequest operation GetBootTarget() to get boot target	GetBootTarget() should return E_OK Boot target should be ECUM_BOOT_TARGET_OEM_BOOTLOADER
Step 5	RUN_Ctrl: executes EcuM_StateRequest operation SelectBootTarget() to set boot target to ECUM_BOOT_TARGET_APP	SelectBootTarget() should return E_OK
Step 6	RUN_Ctrl: executes EcuM_StateRequest operation GetBootTarget() to get boot target	GetBootTarget() should return E_OK Boot target should be ECUM_BOOT_TARGET_APP
Step 7	TCP: terminates RUN_Ctrl	
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
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00036		
Trace to R4.1.1 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. Set the boot target to ECUM_BOOT_TARGET_SYS_BOOTLOADER 2. Get the Boot Target		

	○ Awaited result : ECUM_BOOT_TARGET_SYS_BOOTLOADER 3. Set the boot target to ECUM_BOOT_TARGET_OEM_BOOTLOADER 4. Get the Boot Target ○ 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	TCP: starts RUN_Ctrl	
Step 2	RUN_Ctrl: executes EcuM_StateRequest operation SelectBootTarget() to set boot target to ECUM_BOOT_TARGET_SYS_BOOTLOADER	SelectBootTarget() should return E_OK
Step 3	RUN_Ctrl: executes EcuM_StateRequest operation GetBootTarget() to get boot target	GetBootTarget() should return E_OK Boot target should be ECUM_BOOT_TARGET_SYS_BOOTLOADER
Step 4	RUN_Ctrl: executes EcuM_StateRequest operation SelectBootTarget() to set boot target to ECUM_BOOT_TARGET_OEM_BOOTLOADER	SelectBootTarget() should return E_OK
Step 5	RUN_Ctrl: executes EcuM_StateRequest operation GetBootTarget() to get boot target	GetBootTarget() should return E_OK Boot target should be ECUM_BOOT_TARGET_OEM_BOOTLOADER
Step 6	TCP: terminates RUN_Ctrl	
Post-conditions	None	

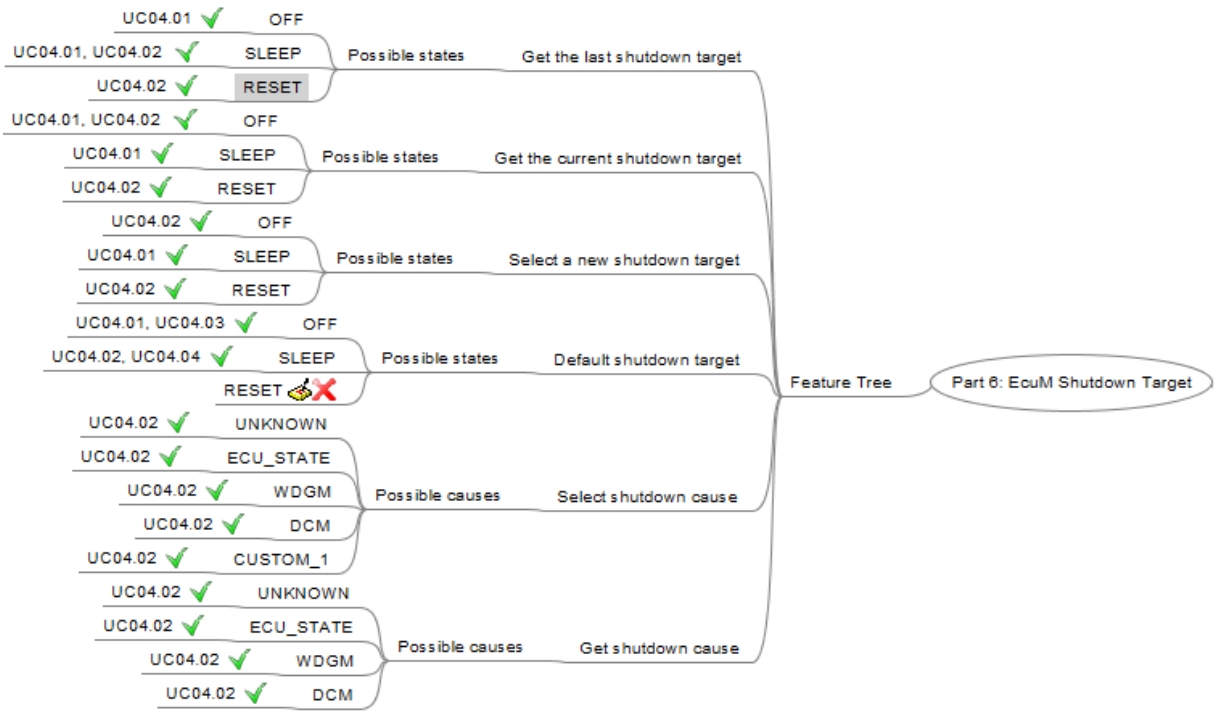
6 RS_BRF_02152 – EcuM Shutdown Target

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:

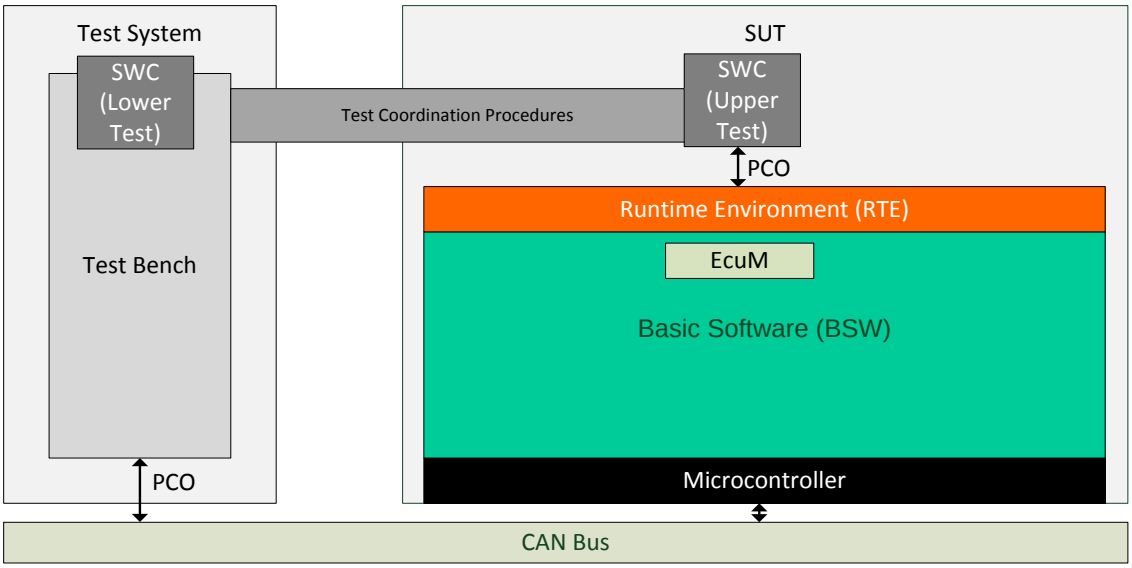


This specification gives the description of required tests environments (test bench, uses case, arxml files) and detailed tests cases for executing tests.

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:

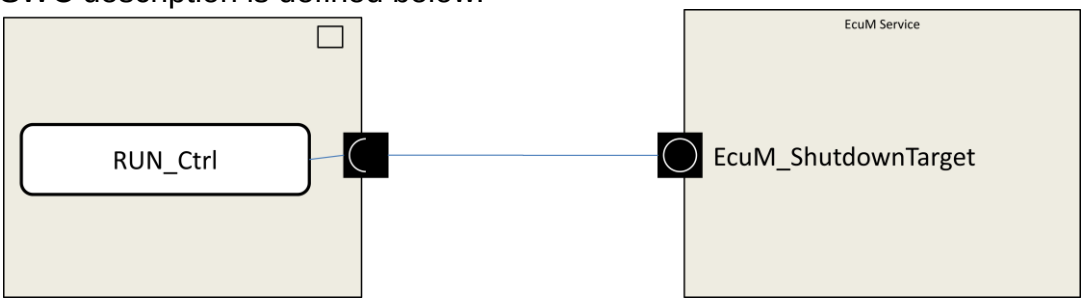
- ➔ EcuM Flexible module is used
- ➔ EcuMDefaultState = EcuMStateOff

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**	3.2.1|3.2.2|4.0.3|4.1.1
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00036		
Trace to R4.1.1 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
	  ○ Awaiting result : Shutdown target = OFF
	2. Switch off the SUT, then switch on the SUT
	3. Get the last shutdown target
	  ○ Awaiting result : Shutdown target = OFF
	4. Select the shutdown target SLEEP
	5. Get the current shutdown target
	  ○ Awaiting result : Shutdown target = SLEEP
	6. Get the last shutdown target
	  ○ Awaiting result : Shutdown target = OFF
	7. Switch off the SUT, then switch on the SUT
	8. Get the last shutdown target
	  ○ Awaiting result : Shutdown target = SLEEP		

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] Use UC04.01 only and exclude running this test case on		

	UC04.03	
Pre-conditions	The SUT is started.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetShutdownTarget() to get current shutdown target	EcuM_ShutdownTarget operation GetShutdownTarget() should return E_OK Current shutdown target (parameter target) should be ECUM_STATE_OFF
Step 2	TCP: restarts SUT	
Step 3	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetLastShutdownTarget() to get last shutdown target	GetLastShutdownTarget() should return E_OK Last shutdown target (parameter target) should be ECUM_STATE_OFF
Step 4	RUN_Ctrl: executes EcuM_ShutdownTarget operation SelectShutdownTarget() with shutdown target ECUM_STATE_SLEEP	SelectShutdownTarget() should return E_OK
Step 5	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetShutdownTarget() to get current shutdown target	GetShutdownTarget() should return E_OK Current shutdown target (parameter target) should be ECUM_STATE_SLEEP
Step 6	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetLastShutdownTarget() to get last shutdown target	GetShutdownTarget() should return E_OK Last shutdown target (parameter target) should be ECUM_STATE_OFF
Step 7	TCP: restarts SUT	
Step 8	TCP: starts RUN_Ctrl	
Step 9	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetLastShutdownTarget() to get last shutdown target	GetShutdownTarget() should return E_OK Last shutdown target (parameter target) should be ECUM_STATE_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**	3.2.1|3.2.2|4.0.3|4.1.1
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00036		
Trace to R4.1.1 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
	  ○ Awaiting result : Shutdown target = SLEEP
	4. Select the shutdown target OFF
	5. Get the current shutdown target
	  ○ Awaiting result : Shutdown target = OFF
	6. Get the last shutdown target
	  ○ Awaiting result : Shutdown target = SLEEP
	7. Select the shutdown target RESET
	8. Get the current shutdown target
	  ○ Awaiting result : Shutdown target = RESET
	9. Get the last shutdown target
	  ○ Awaiting result : Shutdown target = SLEEP
	10. Switch off the SUT, then switch on the SUT
	11. Get the last shutdown target
	  ○ 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	
Pre-conditions	The SUT is started.	
Main Test Execution		
Test Steps		**Pass Criteria**
Step 1	TCP: starts RUN_Ctrl	
Step 2	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetShutdownTarget() to get current shutdown target	GetShutdownTarget() should return E_OK Current shutdown target (parameter target) should be ECUM_STATE_SLEEP
Step 3	TCP: restart SUT	
Step 4	TCP: starts RUN_Ctrl	
Step 5	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetLastShutdownTarget() to get last shutdown target	GetLastShutdownTarget() should return E_OK Last shutdown target (parameter target) should be ECUM_STATE_SLEEP
Step 6	RUN_Ctrl: executes EcuM_ShutdownTarget operation SelectShutdownTarget() to set shutdown target to ECUM_STATE_OFF	SelectShutdownTarget() should return E_OK
Step 7	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetShutdownTarget() to get current shutdown target	GetShutdownTarget() should return E_OK Current shutdown target (parameter target) should be ECUM_STATE_OFF
Step 8	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetLastShutdownTarget() to get last shutdown target	GetLastShutdownTarget() should return E_OK Last shutdown target (parameter target) should be ECUM_STATE_SLEEP
Step 9	RUN_Ctrl: executes EcuM_ShutdownTarget operation SelectShutdownTarget() to set shutdown target to ECUM_STATE_RESET	SelectShutdownTarget() should return E_OK
Step 10	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetShutdownTarget() to get current shutdown target	GetShutdownTarget() should return E_OK Current shutdown target (parameter target) should be ECUM_STATE_RESET
Step 11	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetLastShutdownTarget() to get last shutdown target	GetLastShutdownTarget() should return E_OK Last shutdown target (parameter target) should be ECUM_STATE_SLEEP
Step 12	TCP: restarts SUT	

Step 13	TCP: starts RUN_Ctrl	
Step 14	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetLastShutdownTarget() to get last shutdown target	GetLastShutdownTarget() should return E_OK Last shutdown target (parameter target) should be ECUM_STATE_RESET
Step 15	TCP: terminates RUN_Ctrl	
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
Affected Modules	EcuM	**State**	reviewed
Trace to Requirement on Acceptance Test Document	ATR: ATR_ATR_00036		
Trace to R4.1.1 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
	  ○ Expected result : ECU_STATE
	3. Select the shutdown cause WDGM
	4. Get the shutdown cause
	  ○ Expected result : WDGM		

	5. Select the shutdown cause DCM 6. Get the shutdown cause ○ Expected result : DCM 7. Select the shutdown cause UNKNOWN 8. Get the shutdown cause ○ Expected result : UNKNOWN 9. Select the shutdown cause CUSTOM_1 10. Get the shutdown cause ○ Expected result : CUSTOM_1
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
Pre-conditions	The SUT is started.
Main Test Execution	

Test Steps		**Pass Criteria**
Step 1	TCP: starts RUN_Ctrl	
Step 2	RUN_Ctrl: executes EcuM_ShutdownTarget operation SelectShutdownCause() to select shutdown cause ECU_STATE	SelectShutdownCause() should return E_OK
Step 3	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetShutdownCause() to get shutdown cause	GetShutdownCause() should return E_OK Shutdown cause should be ECU_STATE
Step 4	RUN_Ctrl: executes EcuM_ShutdownTarget operation SelectShutdownCause() to select shutdown cause WDGM	SelectShutdownCause() should return E_OK
Step 5	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetShutdownCause() to get shutdown cause	GetShutdownCause() should return E_OK Shutdown cause should be WDGM
Step 6	RUN_Ctrl: executes EcuM_ShutdownTarget operation SelectShutdownCause() to select shutdown cause DCM	SelectShutdownCause() should return E_OK
Step 7	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetShutdownCause() to get shutdown cause	GetShutdownCause() should return E_OK Shutdown cause should be DCM
Step 8	RUN_Ctrl: executes EcuM_ShutdownTarget operation SelectShutdownCause() to select shutdown cause UNKNOWN	SelectShutdownCause() should return E_OK
Step 9	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetShutdownCause() to get shutdown cause	GetShutdownCause() should return E_OK Shutdown cause should be UNKNOWN
Step 10	RUN_Ctrl: executes	SelectShutdownCause() should

	EcuM_ShutdownTarget operation SelectShutdownCause() to select shutdown cause CUSTOM_1	return E_OK
Step 11	RUN_Ctrl: executes EcuM_ShutdownTarget operation GetShutdownCause() to get shutdown cause	GetShutdownCause() should return E_OK Shutdown cause should be CUSTOM_1
Step 12	TCP: terminates RUN_Ctrl	
Post-conditions	None	