

<b>Document Title</b>	Specification of Hardware Test Manager on start up and shutdown
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	703

<b>Document Status</b>	obsolete
<b>Part of AUTOSAR Standard</b>	Classic Platform
<b>Part of Standard Release</b>	R25-11

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Description</b>
2025-11-27	R25-11	AUTOSAR Release Management	<ul style="list-style-type: none"><li>Marked the document as obsolete</li></ul>
2024-11-27	R24-11	AUTOSAR Release Management	<ul style="list-style-type: none"><li>No content changes</li></ul>
2023-11-23	R23-11	AUTOSAR Release Management	<ul style="list-style-type: none"><li>No content changes</li></ul>
2022-11-24	R22-11	AUTOSAR Release Management	<ul style="list-style-type: none"><li>No content changes</li></ul>
2021-11-25	R21-11	AUTOSAR Release Management	<ul style="list-style-type: none"><li>No content changes</li></ul>
2020-11-30	R20-11	AUTOSAR Release Management	<ul style="list-style-type: none"><li>Chapter 7.3.1 generated from BSW UML model</li><li>Chapter 7.1 structure updated</li></ul>
2019-11-28	R19-11	AUTOSAR Release Management	<ul style="list-style-type: none"><li>Chapter 8 generated from BSW UML model</li><li>Changed Document Status from Final to published</li></ul>
2018-10-31	4.4.0	AUTOSAR Release Management	<ul style="list-style-type: none"><li>Headerfile cleanup</li></ul>



△

2016-11-30	4.3.0	AUTOSAR Release Management	• Initial Release
------------	-------	----------------------------------	-------------------

## **Disclaimer**

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## Table of Contents

1	Introduction and functional overview	6
2	Acronyms and Abbreviations	9
3	Related documentation	10
3.1	Input documents & related standards and norms	10
3.2	Related specification	10
4	Constraints and assumptions	11
4.1	Limitations	11
4.2	Applicability to car domains	11
5	Dependencies to other modules	12
5.1	EcuM	12
5.2	Application SWC	12
5.3	RTE	12
5.4	Dependencies with MSTP	12
5.5	MCU	13
5.6	Default Error Tracer (Det)	13
5.7	File structure	13
5.7.1	Code file structure	13
6	Requirements Tracing	14
7	Functional specification	16
7.1	General behavior	16
7.2	Hardware Test Management	16
7.2.1	Background & Rationale	16
7.2.2	Requirements	16
7.2.3	States of HTMSS module	17
7.3	Error Classification	18
7.3.1	Development Errors	18
7.3.2	Runtime Errors	18
7.3.3	Production Errors	18
7.3.4	Extended Production Errors	18
7.4	Security Events	18
8	API specification	19
8.1	Imported types	19
8.2	Type definitions	19
8.2.1	HTMSS_TestCfgType	19
8.2.2	HTMSS_TestStatusType	20
8.2.3	HTMSS_TestGroupType	20
8.2.4	HTMSS_TestResultType	21

8.3 Function definitions . . . . .	21
8.3.1 HTMSS_Init . . . . .	21
8.3.2 HTMSS_StartTest . . . . .	23
8.3.3 HTMSS_GetTestStatus . . . . .	24
8.3.4 HTMSS_GetVersionInfo . . . . .	26
8.4 Callback notifications . . . . .	26
8.5 Scheduled functions . . . . .	26
8.6 Expected interfaces . . . . .	26
8.6.1 Mandatory Interfaces . . . . .	27
8.6.2 Optional Interfaces . . . . .	27
8.6.3 Configurable interfaces . . . . .	27
8.7 Service Interfaces . . . . .	27
8.7.1 Client server interface - GetTestStatus . . . . .	27
8.8 Callout Definitions . . . . .	28
8.8.1 HTMSS_StartupTestErrorHandler . . . . .	29
8.8.2 HTMSS_ShutdownTestErrorHandler . . . . .	29
9 Sequence diagrams . . . . .	31
9.1 Sequence diagram example of HTMSS Initialization . . . . .	31
9.2 Sequence diagram example of startup test execution . . . . .	32
9.3 Sequence diagram example of shutdown test execution . . . . .	33
9.4 Sequence diagram example handling the last shutdown test results immediately after the ECU reset raised by MSTP module . . . . .	34
9.5 Sequence diagram example of collecting the shutdown test results . . . . .	35
9.6 Sequence diagram example of ECU shutdown when HTMSS is inte- grated in the system . . . . .	36
9.7 Sequence diagram example of application SWC collecting the test results	37
10 Configuration specification . . . . .	38
10.1 Containers and configuration parameters . . . . .	38
10.1.1 HTMSS . . . . .	38
10.1.2 HTMSSGeneral . . . . .	39
10.1.3 HTMSSConfigSet . . . . .	40
10.2 Published Information . . . . .	41
A Not applicable requirements . . . . .	42
B Change history of AUTOSAR traceable items . . . . .	43
B.1 Traceable item history of this document according to AUTOSAR Release R24-25 . . . . .	43
B.1.1 Added Specification Items in R25-11 . . . . .	43
B.1.2 Changed Specification Items in R25-11 . . . . .	43
B.1.3 Deleted Specification Items in R25-11 . . . . .	43

# 1 Introduction and functional overview

This specification describes the concept, interfaces and the configuration of the module Hardware Test Management start up and shutdown (HTMSS).

The module HTMSS [1] is a basic software module at the service layer of the standardized basic software architecture of AUTOSAR.

The HTMSS [1] module shall provide the test status/results for the application SWC usages.

The purpose of this module is to provide an infrastructure for integrating/transforming the microcontroller manufacturer specific start up and shutdown tests (e.g. BIST) test results/status within the AUTOSAR standard software platform.

The basic functionalities of this module includes collecting the test results/status from the MSTP, configure MSTP tests, start tests execution, provide the MSTP test status to EcuM module and application SWC to evaluate the test results for the system behavior.

The HTMSS [1] module integrates on the level of the AUTOSAR BSW service layer. Below figure shows the functional integration of the HTMSS [1] module in AUTOSAR software platform.

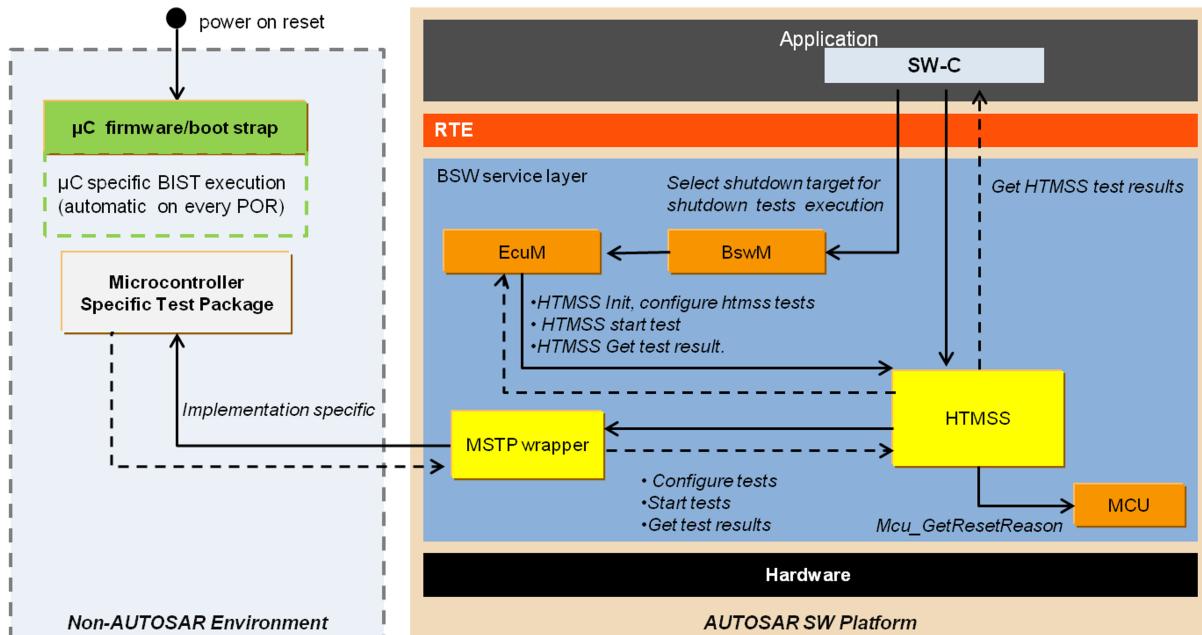


Figure 1.1: HTMSS interaction overview

**Note:** MSTP wrapper is an intermediate module for accessing the MSTP module from an AR standardized module HTMSS. The MSTP wrapper can be implemented manually or can be generated/configured using AUTOSAR methodology/process.

The HTMSS module pre-integration requirements are:

- It shall be possible to run Microcontroller Specific Test Package (MSTP) startup and shutdown tests on the device under development
- The test results/status are available to the HTMSS module access
- It shall be possible to configure the MSTP start up and shutdown tests via HTMSS module

The role of HTMSS module in different phases of the standard AUTOSAR software execution platform is depicted below.

**Note:** The HTMSS concept may be considered for integration in AUTOSAR architecture to achieve safety goals for a safety relevant ECU, but it is NOT mandatory always.

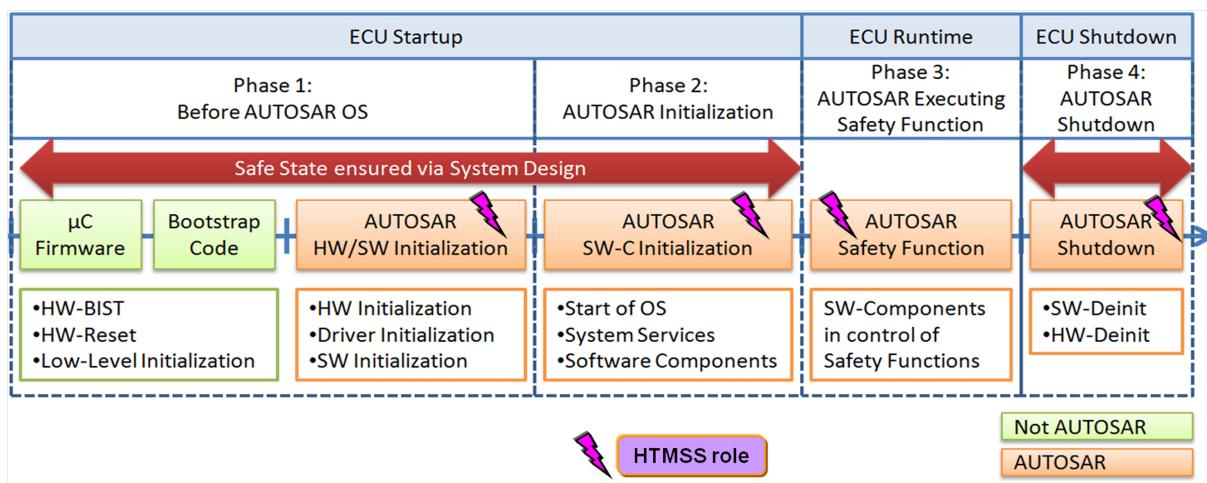


Figure 1.2: HTMSS phases overview

**Note:** The HTMSS phases described below are for explaining the functionalities of HTMSS module in a typical AUTOSAR ECU software execution environment. These phases shall NOT be referred to the phases defined in Ecum [2].

**HTMSS Phase 1 Before AUTOSAR OS [3]** - This phase is delimited by the MCU reset to the call to StartOS() function. During this state there a wide range of possibilities for execution of various types of tests. The MCU periphery and AUTOSAR is not initialized at first, this provides potential opportunity for executing destructive tests, MCU built-in tests, fault injection tests etc. That phase is also used to evaluate the results obtained by tests during shutdown phase, through the reset logic.

During AUTOSAR HW/SW Initialization by Ecum\_Init(), it is possible to execute further diagnostic tests within the AUTOSAR context. Rather non-destructive tests can be executed within Ecum [2].

The HTMSS will be fully available at the end of Phase 2, since it requires integral parts of AUTOSAR to be executed as a System Service.

**HTMSS Phase 2** AUTOSAR OS and SW-C initialization - the phase is delimited by the start of AUTOSAR OS [3] using the function call StartOS() until the complete AUTOSAR is initialized including application software components.

During this phase, the diagnostic test results can be provided by HTMSS and consumed by Safety SW-C for further decisions.

**HTMSS Phase 3** AUTOSAR executing safety function - During this phase, the system has started the intended functionality and safety function is part of it. The phase is suitable for monitoring mechanisms accommodation as well as some built-in diagnostic mechanisms, which could be single or latent fault contributors - ECC fault detection mechanisms, ADC operational capabilities etc. The HTMSS concept does not support Runtime Tests yet (which is a different set of tests), therefore HTMSS can only provide test results from the previously executed Startup and Shutdown tests during Phase 3.

**HTMSS Phase 4** AUTOSAR shutdown. This phase offers a possibility to execute tests, which are not preferable to be executed at any other phase (for example the execution time is too long) and which are able to communicate their results over an MCU reset. The results can be evaluated during a subsequent MCU startup.

## 2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the HWTestManager module that are not included in the [4, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
ADC	Analog to Digital converter
BIST	Built In Self Test
BSW	Basic Software
DET	Default error tracer
ECU	Electronic Control Unit
ECUM	Electronic Control Unit Manager
HTMSS	Hardware Test Management startup shutdown
MCU	Micro Controller Unit
MSTP	Microcontroller Specific Test Package
RTE	Run Time Environment

**Table 2.1: Acronyms and abbreviations used in the scope of this Document**

## 3 Related documentation

### 3.1 Input documents & related standards and norms

- [1] Specification of Hardware Test Manager on start up and shutdown  
AUTOSAR\_CP\_SWS\_HWTestManager
- [2] Specification of ECU State Manager  
AUTOSAR\_CP\_SWS\_ECUStateManager
- [3] Specification of Operating System  
AUTOSAR\_CP\_SWS\_OS
- [4] Glossary  
AUTOSAR\_FO\_TR\_Glossary
- [5] General Specification of Basic Software Modules  
AUTOSAR\_CP\_SWS\_BSWGeneral
- [6] Layered Software Architecture  
AUTOSAR\_CP\_EXP\_LayeredSoftwareArchitecture
- [7] Specification of RTE Software  
AUTOSAR\_CP\_SWS\_RTE

### 3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [5, SWS BSW General], which is also valid for HWTestManager.

Thus, the specification SWS BSW General shall be considered as additional and required specification for HWTestManager.

## 4 Constraints and assumptions

This document is applicable for AUTOSAR release 4.3.0

### 4.1 Limitations

The use of this module is optional and only in case where the provided functionality is required.

To integrate needed range of testing capabilities for specific solution, it has required that all affected modules need to implement interfaces with HTMSS.

Example: The MSTP (microcontroller specific test package) module from the semi manufacturer is a mandatory module for integrating the HTMSS module in AUTOSAR software platform

The start up/shutdown test configurations are up to the system integrator and based on the MSTP test configuration capabilities and features.

The module HTMSS shall interact with the assumed test module (MSTP) via a wrapper implementation (in this spec named as MSTP wrapper) using AR methodology/process.

The test results storage in NV memory and DEM error reporting requirements are out of scope from HTMSS module. Integrator shall manage these requirements at respective application SWC level, if needed.

### 4.2 Applicability to car domains

Each ECU is designed to provide predefined functionality in the context of given system architecture. Then it is of great importance that this ECU operates without failures, which in turn can be avoided or detected before they appear, by simple monitoring of expected faults. One strategy to check the operability of ECU is to execute destructive tests (during start up and shutdown of ECU) that check the given logic and conditions, and keep the results for further analysis. The HTMSS module depicts the need to address results from such tests on ECU, and to provide their status on request.

## 5 Dependencies to other modules

The HTMSS has interfaces to some BSW Modules [5] and application SWC in the AUTOSAR architecture [6]. Additionally HTMSS has interfaces with Microcontroller Specific Test Package (MSTP) outside the AUTOSAR architecture. However, the interactions with MSTP are implementation specific.

### 5.1 EcuM

The ECU State Manager [2] shall access the HTMSS services to start the tests, and collect test results/status from the device under test.

The ECUM STARTUP phase and SHUTDOWN phase incorporates the main functionalities of HTMSS module in AUTOSAR software platform.

### 5.2 Application SWC

The application software component shall collect the HTMSS test results (via RTE [7]) for evaluations and then to determine the software behavior. Additionally, if needed the test results shall be stored in the non-volatile memory for later use.

### 5.3 RTE

Through the RTE [7] data exchange the test result/status are shared between the HTMSS module and the application software layer.

### 5.4 Dependencies with MSTP

The HTMSS [1] may access the MSTP module (could be a non-AR software module, being synchronous and/or asynchronous) to manage the below functionalities/ features within the AUTOSAR software platform.

- To configure the start up and shutdown tests configured in the HTMSS module
- To trigger the MSTP tests during the ECUM start up and shutdown phases
- To collect the test results and provide to application software for its usage

**Note:** HTMSS module shall interact with MSTP module through wrapper module/-source code (named in this spec MSTP wrapper) configurated/generated using AUTOSAR methodology and process.

## 5.5 MCU

The HTMSS receives the reset reason from the MCU driver (e.g. reset caused by shutdown tests execution)

## 5.6 Default Error Tracer (Det)

If the DET is enabled, the HTMSS module informs the Default error tracer about the detected development errors.

## 5.7 File structure

### 5.7.1 Code file structure

#### [SWSHTMSS\_00001]

*Upstream requirements:* [SRS\\_BSW\\_00346](#)

[The code file structure shall contain one or more source files HTMSS\_<xxx>.c, which contains the entire parts of the HTMSS code.]

## 6 Requirements Tracing

The following tables reference the requirements specified in [5] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00159]	All modules of the AUTOSAR Basic Software shall support a tool based configuration	[SWS_HTMSS_00016]
[SRS_BSW_00301]	All AUTOSAR Basic Software Modules shall only import the necessary information	[SWS_HTMSS_00012]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_HTMSS_00038]
[SRS_BSW_00337]	Classification of development errors	[SWS_HTMSS_00011]
[SRS_BSW_00345]	BSW Modules shall support pre-compile configuration	[SWS_HTMSS_00006] [SWS_HTMSS_00016]
[SRS_BSW_00346]	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	[SWS_HTMSS_00001]
[SRS_BSW_00384]	The Basic Software Module specifications shall specify at least in the description which other modules they require	[SWS_HTMSS_00040]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_HTMSS_00029] [SWS_HTMSS_00037]
[SRS_BSW_00404]	BSW Modules shall support post-build configuration	[SWS_HTMSS_00015]
[SRS_BSW_00407]	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	[SWS_HTMSS_00039]
[SRS_BSW_00480]	Null pointer errors shall follow a naming rule	[SWS_HTMSS_00024]
[SRS_HTMSS_00001]	The HTMSS shall allow configuration of start up and shutdown tests	[SWS_HTMSS_00014] [SWS_HTMSS_00017] [SWS_HTMSS_00018] [SWS_HTMSS_00023] [SWS_HTMSS_91006] [SWS_HTMSS_91007] [SWS_HTMSS_91008] [SWS_HTMSS_91009]
[SRS_HTMSS_00002]	The HTMSS shall allow the configuration of tests at individual hardware resource level	[SWS_HTMSS_00008] [SWS_HTMSS_00009] [SWS_HTMSS_0014] [SWS_HTMSS_0017] [SWS_HTMSS_0018] [SWS_HTMSS_0022] [SWS_HTMSS_91006] [SWS_HTMSS_91007] [SWS_HTMSS_91008] [SWS_HTMSS_91009]
[SRS_HTMSS_00003]	The HTMSS shall provide a service to collect the MSTP tests results	[SWS_HTMSS_00005] [SWS_HTMSS_00010] [SWS_HTMSS_00032] [SWS_HTMSS_00033] [SWS_HTMSS_00034] [SWS_HTMSS_00035] [SWS_HTMSS_00036]
[SRS_HTMSS_00004]	The HTMSS shall provide a mechanism to share the test results with the application layer software	[SWS_HTMSS_00031] [SWS_HTMSS_00032] [SWS_HTMSS_00042]





Requirement	Description	Satisfied by
[SRS_HTMSS_00005]	The HTMSS shall provide a service to configure/Initialise the MSTP tests during ECUM start up phase	[SWS_HTMSS_00019] [SWS_HTMSS_00020] [SWS_HTMSS_00021] [SWS_HTMSS_00030]
[SRS_HTMSS_00006]	The HTMSS shall provide a service to trigger the tests execution	[SWS_HTMSS_00025] [SWS_HTMSS_00026] [SWS_HTMSS_00027] [SWS_HTMSS_00028] [SWS_HTMSS_00030]
[SRS_HTMSS_00007]	HTMSS shall provide callout options to handle the test failure conditions	[SWS_HTMSS_00043] [SWS_HTMSS_00044]

**Table 6.1: Requirements Tracing**

## 7 Functional specification

### 7.1 General behavior

The basic functionalities of HTMSS can be divided into the following main groups:

- Initialization of HTMSS module
- Configure the MSTP tests based on HTMSS configuration (start up/shutdown)
- Interface for starting the MSTP tests (startup and shutdown)
- Provide the MSTP tests status to the other autosar modules (incl. appln SWC)

**Note:** The HTMSS shall not add any test functionality corresponding to MSTP tests. The tests implementation and execution is out of HTMSS scope.

### 7.2 Hardware Test Management

#### 7.2.1 Background & Rationale

The overall objective is to provide a fault status of the microcontroller operation by means of hardware test execution in safety-related systems, built on standard AUTOSAR platform.

The concept shall provide a facility to execute and log a predefined set of tests. Additionally shall support HW monitoring activities in the context of microcontroller in use, obtain tests results and propagate them to stakeholder software components in AUTOSAR environment.

The goal of HTMSS module is to standardize the accessible interfaces that can perform the microcontroller specific start up/shutdown tests outside the AUTOSAR environment and then to collect & evaluate the test results within AUTOSAR software execution context.

#### 7.2.2 Requirements

##### [SWS-HTMSS\_00005]

*Upstream requirements:* [SRS-HTMSS\\_00003](#)

[The HTMSS shall be able to read the microcontroller specific start up and shutdown test results/status, on the requested hardware.]

**Note:** It is the responsibility of user/integrator to evaluate the HTMSS provided start up and shutdown test results (i.e. in case of failure) and then to define the software reactions. The error hooks (start up & shutdown) shall evaluate the test results.

**HINT:** Integrator may have prioritised the tests handling the test results based on the criticality/relevance in the system/safety goals etc. In case of a critical error integrator shall decide to go back to reset state or shutdown state.

### [SWS\_HTMSS\_00006]

*Upstream requirements:* [SRS\\_BSW\\_00345](#)

〔The pre-compile time configuration parameters shall be checked statically (at least during compile time) for correctness.〕

### [SWS\_HTMSS\_00008]

*Upstream requirements:* [SRS\\_HTMSS\\_00002](#)

〔In one configuration, there shall be more than one test per module for testing. These tests could be executed in parallel.〕

**Note:** The configuration parameter to handle above requirement shall be adapted in HTMSSConfigSet [Chapter 10.1.3](#)

### [SWS\_HTMSS\_00009]

*Upstream requirements:* [SRS\\_HTMSS\\_00002](#)

〔It shall be possible to test the individual hardware resources (e.g. selected via module / channel ID) on the given hardware.〕

**HINT:** A microcontroller may contain two hardware unit for the ADC peripheral. There shall be a support to test / obtain test results for each ADC unit individually.

**Note:** There can be no guarantee that errors in unused hardware resources do not propagate or have influence on the rest of the microcontroller. Therefore, a complete hardware tests may need to be executed and the test results shall be considered appropriately based on the safety requirements of the ECU.〕

Refer [Chapter 10.1.3](#) HTMSSConfigSet for configuration parameter implementations.

## 7.2.3 States of HTMSS module

### [SWS\_HTMSS\_00010]

*Upstream requirements:* [SRS\\_HTMSS\\_00003](#)

〔

State	Description
HTMSS_UINIT	HTMSS module uninitialized(default value before module initialization)
HTMSS_INIT	HTMSS module initialized
HTMSS_BUSY	The HTMSS requested tests are not completed/progressing/initiated
HTMSS_IDLE	HTMSS requested tests are completed/ NO pending tests are running

〕

## 7.3 Error Classification

Chapter [5, General Specification of Basic Software Modules] 7.2 “*Error Handling*” describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

### 7.3.1 Development Errors

#### [SWS\_HTMSS\_00011] Definition of development errors in module HTMSS

*Upstream requirements:* [SRS\\_BSW\\_00337](#)

]

Type of error	Related error code	Error value
A service was called prior to initialization	HTMSS_E_NOT_INIT	0x01
A null pointer was passed as an argument	HTMSS_E_NULL_POINTER	0x02
A parameter was invalid (unspecific)	HTMSS_E_PARAM_INVALID	0x03
Function called when test request is running	HTMSS_E_BUSY	0x04

]

### 7.3.2 Runtime Errors

There are no runtime errors.

### 7.3.3 Production Errors

There are no production errors.

### 7.3.4 Extended Production Errors

There are no extended production errors.

## 7.4 Security Events

The module does not report security events.

## 8 API specification

### 8.1 Imported types

In this chapter, all types included from the following files are listed:

The HTMSS shall use only the following imported types of other modules:

#### **[SWS-HTMSS\_00012] Definition of imported datatypes of module HTMSS**

*Upstream requirements:* [SRS\\_BSW\\_00301](#)

〔

<i>Module</i>	<i>Header File</i>	<i>Imported Type</i>
Mcu	Mcu.h	Mcu_ResetType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

〕

### 8.2 Type definitions

The following Data Types shall be used for the functions defined in this specification

#### 8.2.1 HTMSS\_TestCfgType

#### **[SWS-HTMSS\_91006] Definition of datatype HTMSS\_TestCfgType**

*Upstream requirements:* [SRS-HTMSS\\_00001](#), [SRS-HTMSS\\_00002](#)

〔

<b>Name</b>	HTMSS_TestCfgType	
<b>Kind</b>	Structure	
<b>Elements</b>	Implementation specific	
	<b>Type</b>	—
	<b>Comment</b>	The content of the configuration data structure is implementation specific.
<b>Description</b>	Configuration data structure of HTMSS module	
<b>Available via</b>	HTMSS.h	

〕

## 8.2.2 HTMSS\_TestStatusType

### [SWS-HTMSS-91009] Definition of datatype HTMSS\_TestStatusType

*Upstream requirements:* [SRS-HTMSS-00001](#), [SRS-HTMSS-00002](#)

[

<b>Name</b>	HTMSS_TestStatusType		
<b>Kind</b>	Enumeration		
<b>Range</b>	HTMSS_STATUS_OK	–	Test status PASS
	HTMSS_STATUS_NOK	–	Test status FAIL
	HTMSS_STATUS_INVALID	–	Test status is Invalid
	HTMSS_STATUS_UNINIT	–	Test status is not initialized
<b>Description</b>	HTMSS_TestStatusType describes status of test.		
<b>Available via</b>	HTMSS.h		

]

## 8.2.3 HTMSS\_TestGroupType

### [SWS-HTMSS-91007] Definition of ImplementationDataType HTMSS\_TestGroupType

*Upstream requirements:* [SRS-HTMSS-00001](#), [SRS-HTMSS-00002](#)

[

<b>Name</b>	HTMSS_TestGroupType		
<b>Kind</b>	Enumeration		
<b>Range</b>	HTMSS_STARTUP	–	Test to be executed at startup only
	HTMSS_SHUTDOWN	–	Test to be executed at shutdown only
	HTMSS_STARTUP_SHUTDOWN	–	Test to be executed at start up and shutdown
<b>Description</b>	HTMSS_TestGroupType describes the test group type		
<b>Variation</b>	–		
<b>Available via</b>	HTMSS.h		

]

## 8.2.4 HTMSS\_TestResultType

### [SWS-HTMSS-91008] Definition of ImplementationDataType HTMSS\_TestResultType

*Upstream requirements:* [SRS-HTMSS-00001](#), [SRS-HTMSS-00002](#)

]

<b>Name</b>	HTMSS_TestResultType									
<b>Kind</b>	Structure									
<b>Elements</b>	TestResult <table border="1"> <tr> <td><b>Type</b></td> <td>uint8</td> </tr> <tr> <td><b>Comment</b></td> <td>The test result (e.g. pass, fail, invalid)</td> </tr> </table> TestSignature <table border="1"> <tr> <td><b>Type</b></td> <td>uint8</td> </tr> <tr> <td><b>Comment</b></td> <td>The identifier of the tested resource</td> </tr> </table>		<b>Type</b>	uint8	<b>Comment</b>	The test result (e.g. pass, fail, invalid)	<b>Type</b>	uint8	<b>Comment</b>	The identifier of the tested resource
<b>Type</b>	uint8									
<b>Comment</b>	The test result (e.g. pass, fail, invalid)									
<b>Type</b>	uint8									
<b>Comment</b>	The identifier of the tested resource									
<b>Description</b>	It describes the current test result									
<b>Variation</b>	–									
<b>Available via</b>	HTMSS.h									

]

## 8.3 Function definitions

The following sections specify the provided API functions of the HTMSS module.

### 8.3.1 HTMSS\_Init

#### [SWS-HTMSS-00014] Definition of API function HTMSS\_Init

*Upstream requirements:* [SRS-HTMSS-00001](#), [SRS-HTMSS-00002](#)

]

<b>Service Name</b>	HTMSS_Init	
<b>Syntax</b>	<code>void HTMSS_Init (</code> <code>    const HTMSS_TestCfgType* ConfigPtr</code> <code>)</code>	
<b>Service ID [hex]</b>	0x01	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	ConfigPtr	Pointer to configuration set in Variant PB (Variant PC requires a NULL_PTR).
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	Initializes the HTMSS module	

▽



<b>Available via</b>	HTMSS.h
----------------------	---------

]

### **[SWSHTMSS\_00015]**

*Upstream requirements:* [SRS\\_BSW\\_00404](#)

〔In case of Variant PB: The function [HTMSS\\_Init](#) shall initialize the HTMSS module and according to the configuration set referenced by [ConfigPtr](#).〕

### **[SWSHTMSS\_00016]**

*Upstream requirements:* [SRS\\_BSW\\_00345](#), [SRS\\_BSW\\_00159](#)

〔In case of Variant PC: The function [HTMSS\\_Init](#) shall initialize the HTMSS according to the pre-compile configuration set.〕

### **[SWSHTMSS\_00017]**

*Upstream requirements:* [SRSHTMSS\\_00001](#), [SRSHTMSS\\_00002](#)

〔The service [HTMSS\\_Init](#) shall initialize the global variables and data structures of the HTMSS including flags and buffers.〕

### **[SWSHTMSS\_00018]**

*Upstream requirements:* [SRSHTMSS\\_00001](#), [SRSHTMSS\\_00002](#)

〔The function [HTMSS\\_Init](#) shall initialize MSTP module and configure the MSTP tests.〕

### **[SWSHTMSS\_00019]**

*Upstream requirements:* [SRSHTMSS\\_00005](#)

〔The HTMSS is not functional until this function has been called.〕

### **[SWSHTMSS\_00020]**

*Upstream requirements:* [SRSHTMSS\\_00005](#)

〔The function [HTMSS\\_Init](#) shall determine the last reset reason by calling the [Mcu\\_GetResetReason](#) of the MCU driver.〕

### **[SWSHTMSS\_00021]**

*Upstream requirements:* [SRSHTMSS\\_00005](#)

〔The function [HTMSS\\_Init](#) shall configure the MSTP tests (both start up and shutdown), if the last reset reason is not [MCU\\_HWTEST\\_RESET](#).〕

### **[SWSHTMSS\_00022]**

*Upstream requirements:* [SRSHTMSS\\_00002](#)

〔The function [HTMSS\\_Init](#) shall configure the start up and shutdown tests individually.〕

**Hint:** The interfaces and test configurations with MSTP module are implementation specific. There may be some cases to configure the same type of tests for execution

in both start up and shutdown slots. [HTMSS\\_TestGroupType](#) can be used in this context.]

**[SWS-HTMSS\_00023]**

*Upstream requirements:* [SRS-HTMSS\\_00001](#)

[The function [HTMSS\\_Init](#) shall set the HTMSS state to [HTMSS\\_UNINIT](#), if the configuration of MSTP tests fails for any reason.]

**[SWS-HTMSS\_00024]**

*Upstream requirements:* [SRS\\_BSW\\_00480](#)

[If DET for the HTMSS is enabled: the function [HTMSS\\_Init](#) shall check for valid pointer. In case of an error, [HTMSS\\_Init](#) shall raise the development error [HTMSS\\_E\\_NULL\\_POINTER](#).]

### 8.3.2 HTMSS\_StartTest

**[SWS-HTMSS\_00025] Definition of API function HTMSS\_StartTest**

*Upstream requirements:* [SRS-HTMSS\\_00006](#)

[

<b>Service Name</b>	HTMSS_StartTest	
<b>Syntax</b>	Std_ReturnType HTMSS_StartTest ( <a href="#">HTMSS_TestGroupType</a> GrpId )	
<b>Service ID [hex]</b>	0x03	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	GrpId	The test group type (e.g. start up or shut down)
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	Standard return from function execution
<b>Description</b>	Starts the MSTP configured tests	
<b>Available via</b>	HTMSS.h	

]

**[SWS-HTMSS\_00026]**

*Upstream requirements:* [SRS-HTMSS\\_00006](#)

[The function [HTMSS\\_StartTest](#) shall trigger the MSTP test operation on the requested hardware. If this is successful, [E\\_OK](#) shall be returned.]

**[SWS-HTMSS\_00027]**

*Upstream requirements:* [SRS-HTMSS\\_00006](#)

[The function [HTMSS\\_StartTest](#) shall set the HTMSS state to [HTMSS\\_BUSY](#), if the MSTP status confirm that, test trigger is successful.]

**[SWS\_HTMSS\_00028]**

*Upstream requirements:* [SRS\\_HTMSS\\_00006](#)

〔The function [HTMSS\\_StartTest](#) shall handle the start up and shutdown test requests for the device under test.〕

**Hint:** The interface between HTMSS and the MSTP module is implementation specific. Because the semi manufacturer may define the method of interacting with MSTP module for the device.〕

**[SWS\_HTMSS\_00029]**

*Upstream requirements:* [SRS\\_BSW\\_00386](#)

〔If DET for the HTMSS is enabled: the function [HTMSS\\_StartTest](#) shall check for valid initialization. In case of failure [HTMSS\\_StartTest](#) shall raise the development error [HTMSS\\_E\\_NOT\\_INIT](#) and return [E\\_NOT\\_OK](#).〕

**[SWS\_HTMSS\_00030]**

*Upstream requirements:* [SRS\\_HTMSS\\_00005](#), [SRS\\_HTMSS\\_00006](#)

〔If DET for the HTMSS is enabled: the function [HTMSS\\_StartTest](#) shall check for the valid input parameter. In case of an error, [HTMSS\\_StartTest](#) shall raise the development error [HTMSS\\_E\\_PARAM\\_INVALID](#) and return [E\\_NOT\\_OK](#).〕

**[SWS\_HTMSS\_00031]**

*Upstream requirements:* [SRS\\_HTMSS\\_00004](#)

〔If DET for the HTMSS is enabled: when called while a start request is already in place, is not in the state [HTMSS\\_IDLE](#), the function [HTMSS\\_StartTest](#) shall raise the development error [HTMSS\\_E\\_BUSY](#) and return [E\\_NOT\\_OK](#).〕

### 8.3.3 HTMSS\_GetTestStatus

**[SWS\_HTMSS\_00032] Definition of API function HTMSS\_GetTestStatus**

*Upstream requirements:* [SRS\\_HTMSS\\_00003](#), [SRS\\_HTMSS\\_00004](#)

〔

<b>Service Name</b>	HTMSS_GetTestStatus	
<b>Syntax</b>	<pre>HTMSS_TestStatusType HTMSS_GetTestStatus (     HTMSS_TestGroupType GrpId,     HTMSS_TestResultType* RequestTestResultPtr )</pre>	
<b>Service ID [hex]</b>	0x04	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	GrpId	The test group type (e.g. start up or shutdown)
<b>Parameters (inout)</b>	None	





<b>Parameters (out)</b>	RequestTestResultPtr	Pointer to store the request result
<b>Return value</b>	HTMSS_TestStatusType	–
<b>Description</b>	Returns current test status on requested test	
<b>Available via</b>	HTMSS.h	

]

**[SWSHTMSS\_00033]***Upstream requirements: SRSHTMSS\_00003*

〔The function `HTMSS_GetTestStatus` shall collect the test status from the MSTP and store the read data in the out parameter `RequestTestResultPtr`, if OUT parameter is not a `NULL_PTR` and return `HTMSS_TestStatusType` with test status result.〕

**[SWSHTMSS\_00034]***Upstream requirements: SRSHTMSS\_00003*

〔If the OUT parameter is `NULL_PTR`, the function `HTMSS_GetTestStatus` shall NOT update the OUT parameter but shall return `HTMSS_TestStatusType` with the test status result.〕

**[SWSHTMSS\_00035]***Upstream requirements: SRSHTMSS\_00003*

〔The function `HTMSS_GetTestStatus` shall set the HTMSS state to `HTMSS_IDLE`, if the MSTP provided status confirm the test completion.〕

**[SWSHTMSS\_00036]***Upstream requirements: SRSHTMSS\_00003*

〔The function `HTMSS_GetTestStatus` shall provide the start up and shutdown test status depend on the input parameter `GrpId`.

**Hint:** The integrator/user shall ensure that a valid test status is readily available before invoking this API function.

**Note:** The methods/mechanisms to read/collect the requested test status are implementation specific and depending on the MSTP interfaces and microcontroller manufacturer specific guidelines.〕

**[SWSHTMSS\_00037]***Upstream requirements: SRS\_BSW\_00386*

〔If DET for the HTMSS is enabled the function `HTMSS_GetTestStatus` shall check for valid initialization. In case of failure `HTMSS_GetTestStatus` shall raise the development error `HTMSS_E_NOT_INIT` and return `HTMSS_STATUS_UNINIT`.〕

**[SWS\_HTMSS\_00038]**

*Upstream requirements:* [SRS\\_BSW\\_00323](#)

〔If DET for the HTMSS is enabled: the function `HTMSS_GetTestStatus` shall check for the valid input parameter. In case of an error, `HTMSS_GetTestStatus` shall raise the development error `HTMSS_E_PARAM_INVALID` and return `HTMSS_STATUS_INVALID`.〕

### 8.3.4 HTMSS\_GetVersionInfo

**[SWS\_HTMSS\_00039] Definition of API function HTMSS\_GetVersionInfo**

*Upstream requirements:* [SRS\\_BSW\\_00407](#)

〔

<b>Service Name</b>	HTMSS_GetVersionInfo	
<b>Syntax</b>	<code>void HTMSS_GetVersionInfo (</code> <code>Std_VersionInfoType* versioninfo</code> <code>)</code>	
<b>Service ID [hex]</b>	0x06	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	None	
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	versioninfo	Pointer to where to store the version information of this module.
<b>Return value</b>	None	
<b>Description</b>	—	
<b>Available via</b>	HTMSS.h	

〕

## 8.4 Callback notifications

None

## 8.5 Scheduled functions

None

## 8.6 Expected interfaces

In this chapter, all interfaces required from other modules are listed.

### 8.6.1 Mandatory Interfaces

This chapter defines all interfaces, which are required to fulfill the core functionality of the module.

#### [SWS\_HTMSS\_00040] Definition of mandatory interfaces required by module HTMSS

*Upstream requirements:* [SRS\\_BSW\\_00384](#)

[

API Function	Header File	Description
Mcu_GetResetReason	Mcu.h	The service reads the reset type from the hardware, if supported.

]

### 8.6.2 Optional Interfaces

This chapter defines all interfaces, which are required to fulfill an optional functionality of the module.

#### [SWS\_HTMSS\_00041] Definition of optional interfaces requested by module HTMSS

API Function	Header File	Description
Det_ReportError	Det.h	Service to report development errors.

]

### 8.6.3 Configurable interfaces

There are no configurable interfaces.

## 8.7 Service Interfaces

This chapter formally specifies the corresponding AUTOSAR service in terms of the SWC template. The interface described here is used to generate the RTE between application software and the HTMSS module.

### 8.7.1 Client server interface - GetTestStatus

#### [SWS\_HTMSS\_00042] Definition of ClientServerInterface GetTestStatus

*Upstream requirements:* [SRS\\_HTMSS\\_00004](#)

[

<b>Name</b>	GetTestStatus		
<b>Comment</b>	–		
<b>IsService</b>	true		
<b>Variation</b>	–		
<b>Possible Errors</b>	0	HTMSS_STATUS_OK	Test status PASS
	0	HTMSS_STATUS_NOK	Test status FAIL
	0	HTMSS_STATUS_UNINIT	Test status is not initialized
	0	HTMSS_STATUS_INVALID	Test status is Invalid

<b>Operation</b>	GetTestStatus		
<b>Comment</b>	Returns current test status on requested test		
<b>Relates to</b>	<a href="#">HTMSS_GetTestStatus</a>		
<b>Variation</b>	–		
<b>Parameters</b>	GrpId		
	<b>Type</b>	<a href="#">HTMSS_TestGroupType</a>	
	<b>Direction</b>	IN	
	<b>Comment</b>	The test group type (e.g. start up or shutdown)	
	<b>Variation</b>	–	
	RequestTestResultPtr		
	<b>Type</b>	<a href="#">HTMSS_TestResultType</a>	
	<b>Direction</b>	OUT	
	<b>Comment</b>	Pointer to provide Test results along with Test	
	<b>Variation</b>	–	
<b>Possible Errors</b>	<a href="#">HTMSS_STATUS_OK</a> <a href="#">HTMSS_STATUS_NOK</a> <a href="#">HTMSS_STATUS_INVALID</a> <a href="#">HTMSS_STATUS_UNINIT</a>		

]

### [SWS-HTMSS\_91011] Definition of Port GetTestStatus required by module HTMSS

<b>Name</b>	GetTestStatus		
<b>Kind</b>	RequiredPort	<b>Interface</b>	<a href="#">GetTestStatus</a>
<b>Description</b>	–		
<b>Variation</b>	–		

]

## 8.8 Callout Definitions

Callouts are code fragments that must be added to the HTMSS module during ECU integration. The content of most callouts is hand-written code. The HTMSS module configuration tool generates a default implementation for some callouts which is edited manually by the integrator. Conceptually, these callouts belong to the ECU integration code.

**Note:** The error hook is an integration code to control the ECU processing in case of any tests failure. It may be a critical error for the system under development, so the integrator can react to the test failure (e.g. reset, halt, safe state)

### 8.8.1 HTMSS\_StartupTestErrorHandler

#### [SWS-HTMSS-00043] Definition of API function HTMSS\_StartupTestErrorHandler

*Upstream requirements:* [SRS-HTMSS-00007](#)

[

<b>Service Name</b>	HTMSS_StartupTestErrorHandler
<b>Syntax</b>	void HTMSS_StartupTestErrorHandler ( void )
<b>Service ID [hex]</b>	0x07
<b>Sync/Async</b>	Synchronous
<b>Reentrancy</b>	Non Reentrant
<b>Parameters (in)</b>	None
<b>Parameters (inout)</b>	None
<b>Parameters (out)</b>	None
<b>Return value</b>	None
<b>Description</b>	The ECU State Manager will call the error hook if the HTMSS provided startup test results have a failure. In this situation, the integrator has to control the CPU processing based on the system requirements, i.e. reset, safe state etc...
<b>Available via</b>	

]

### 8.8.2 HTMSS\_ShutdownTestErrorHandler

#### [SWS-HTMSS-00044] Definition of callout function HTMSS\_ShutdownTestErrorHandler

*Upstream requirements:* [SRS-HTMSS-00007](#)

[

<b>Service Name</b>	HTMSS_ShutdownTestErrorHandler
<b>Syntax</b>	void HTMSS_ShutdownTestErrorHandler ( void )
<b>Service ID [hex]</b>	0x08
<b>Sync/Async</b>	Synchronous
<b>Reentrancy</b>	Non Reentrant
<b>Parameters (in)</b>	None
<b>Parameters (inout)</b>	None
<b>Parameters (out)</b>	None

▽

△

<b>Return value</b>	None
<b>Description</b>	The ECU State Manager will call the error hook if the HTMSS provided shutdown test results have a failure. In this situation, the integrator has to control the CPU processing based on the system requirements, i.e. reset, safe state etc...
<b>Available via</b>	HTMSS.h

]

## 9 Sequence diagrams

### 9.1 Sequence diagram example of HTMSS Initialization

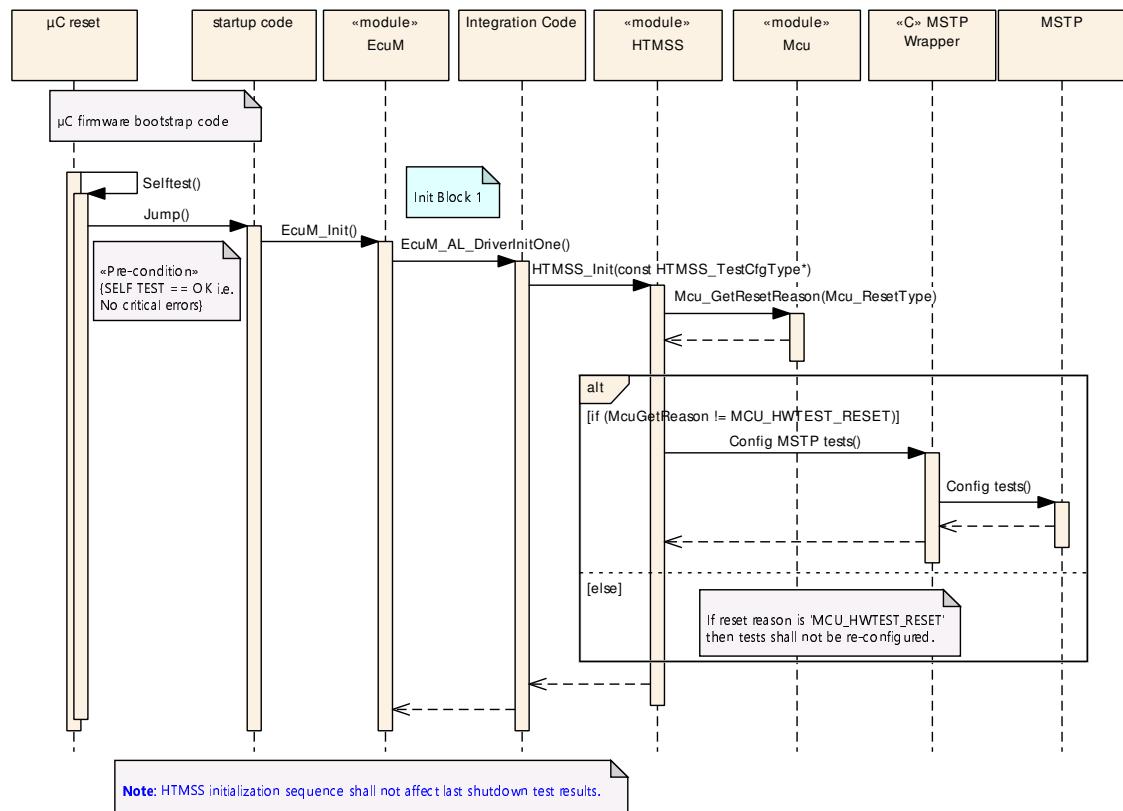


Figure 9.1: HTMSS Initialization

## 9.2 Sequence diagram example of startup test execution

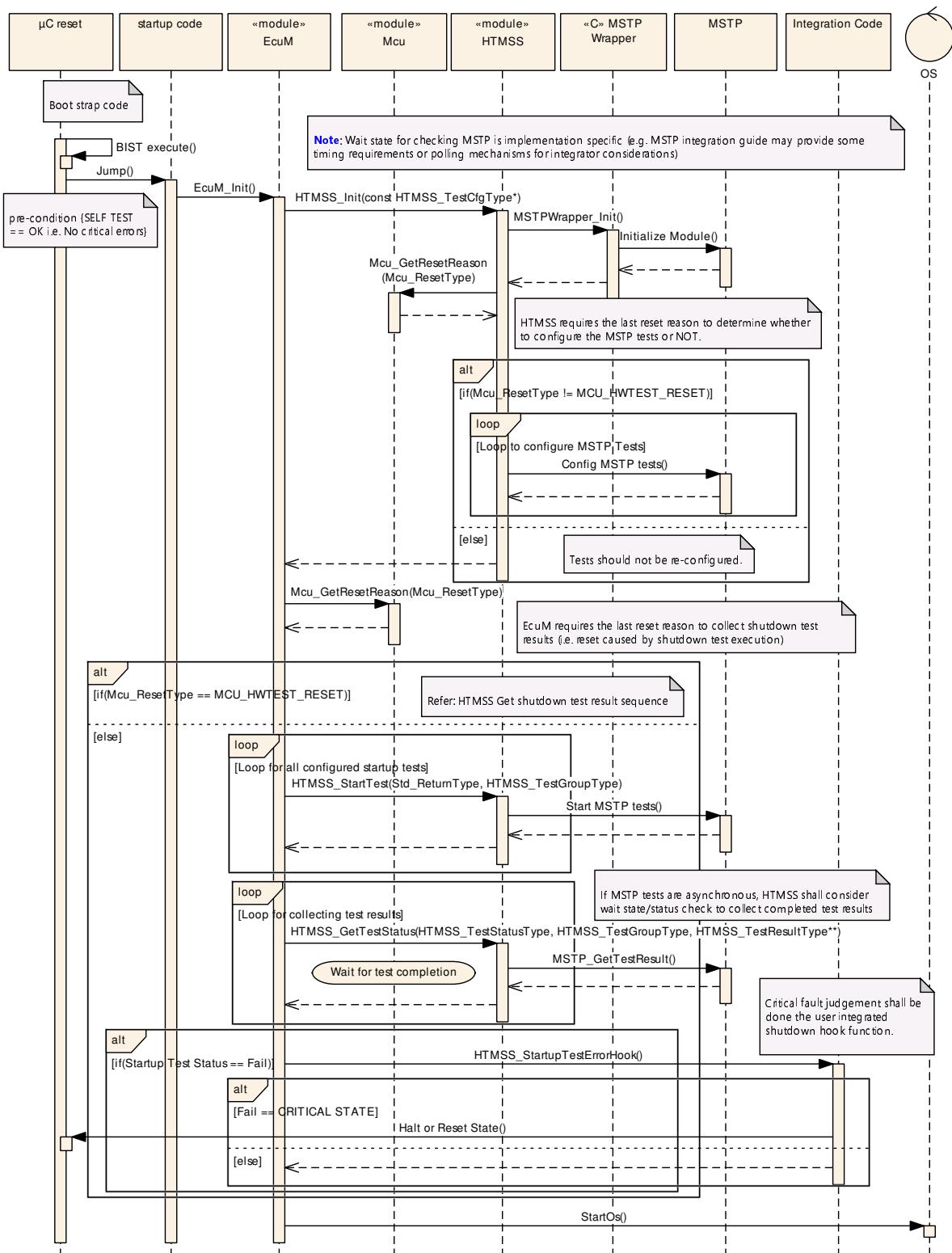


Figure 9.2: Startup test execution

### 9.3 Sequence diagram example of shutdown test execution

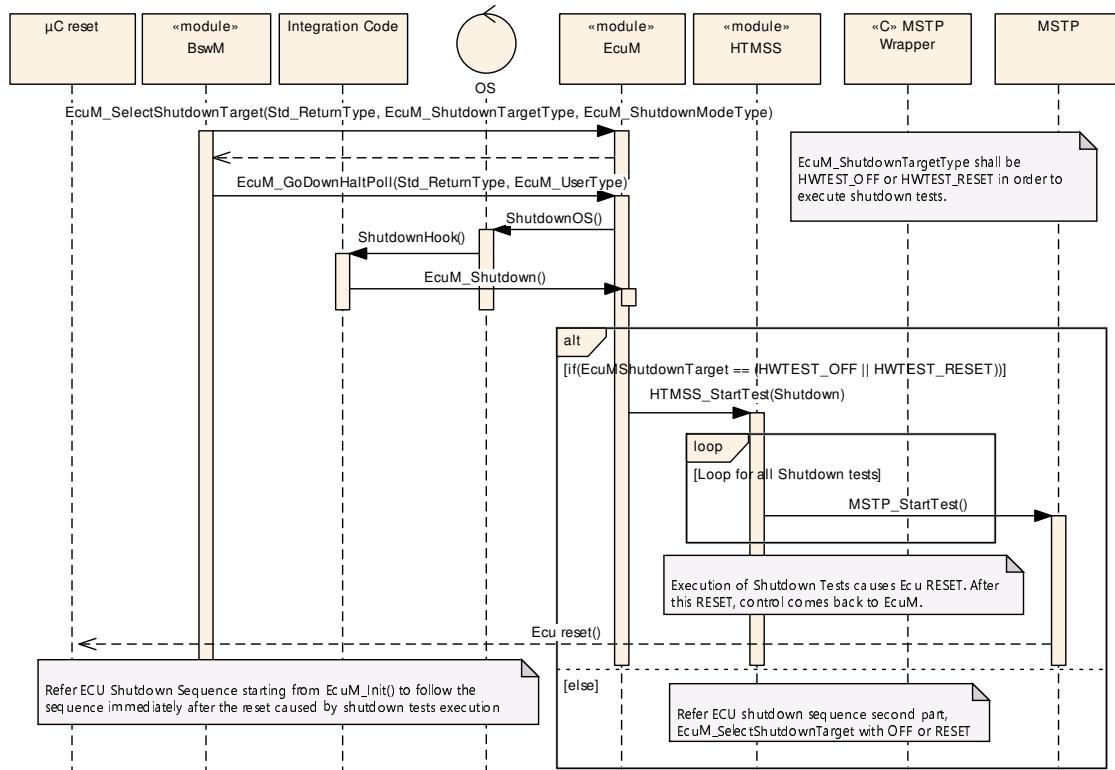


Figure 9.3: Shutdown test execution

## 9.4 Sequence diagram example handling the last shutdown test results immediately after the ECU reset raised by MSTP module

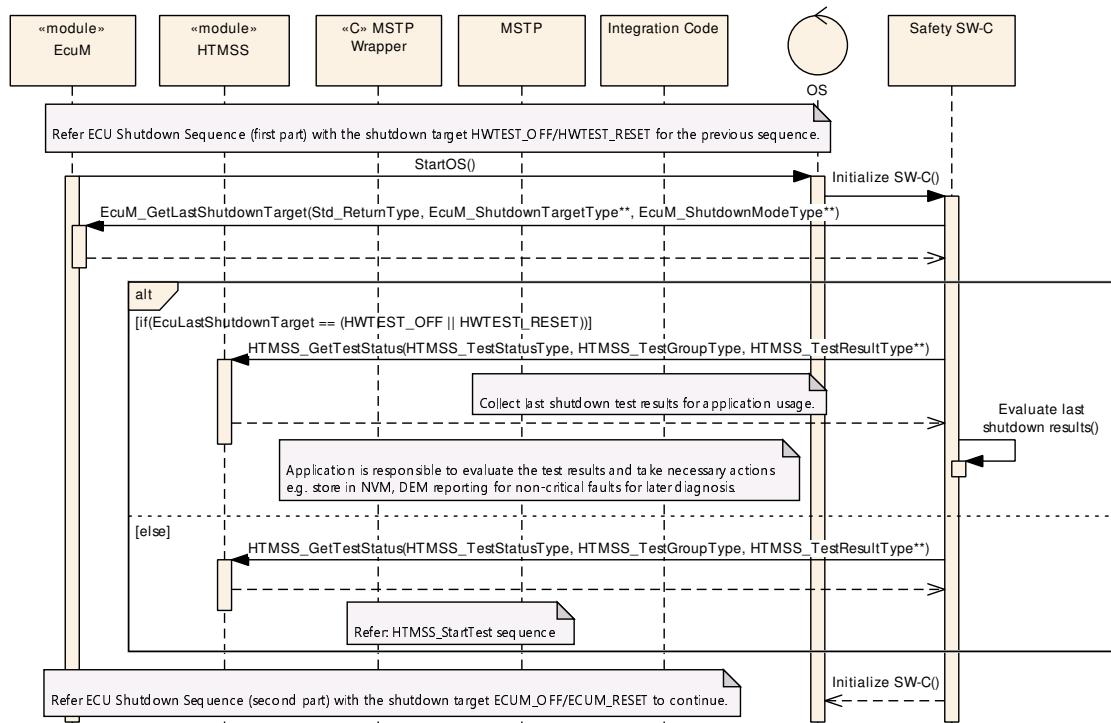
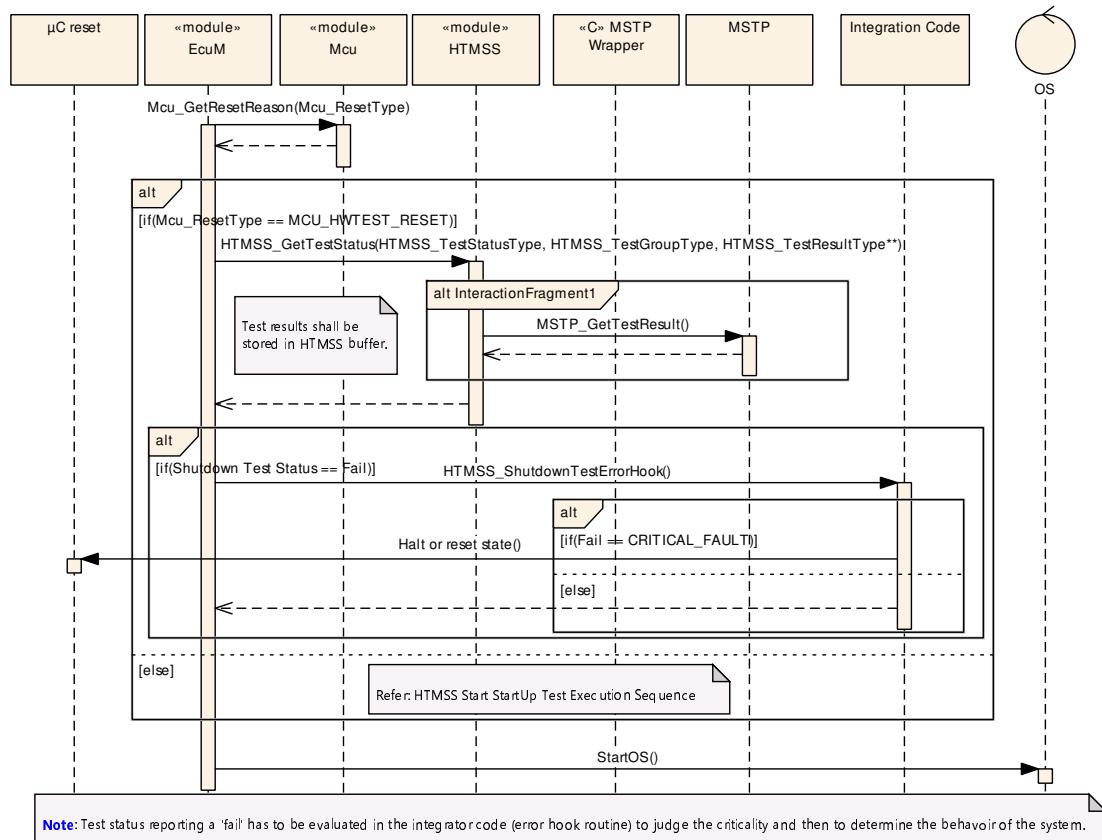


Figure 9.4: Handling the last shutdown test results

## 9.5 Sequence diagram example of collecting the shutdown test results



**Figure 9.5: Collecting the shutdown test results**

## 9.6 Sequence diagram example of ECU shutdown when HTMSS is integrated in the system

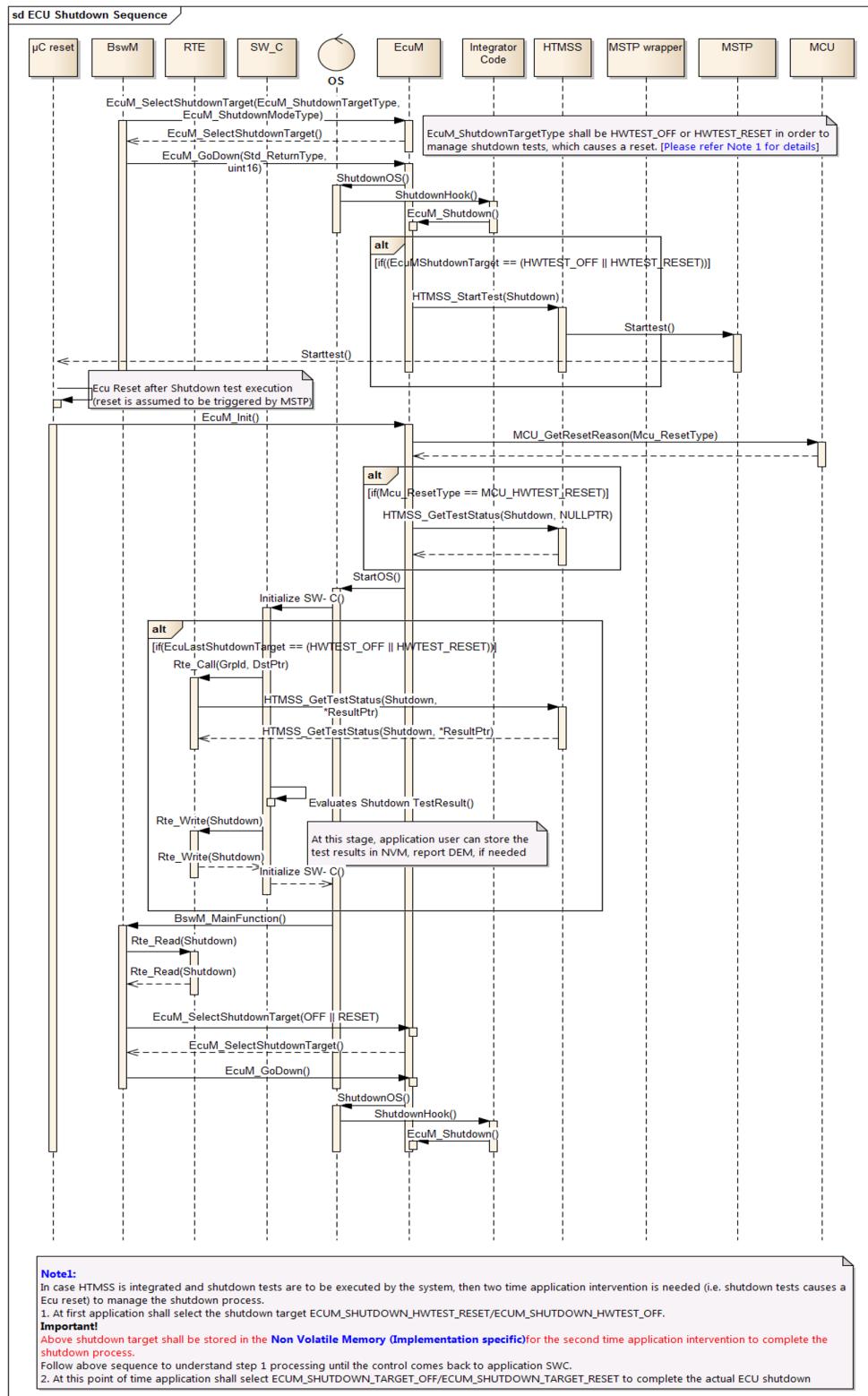
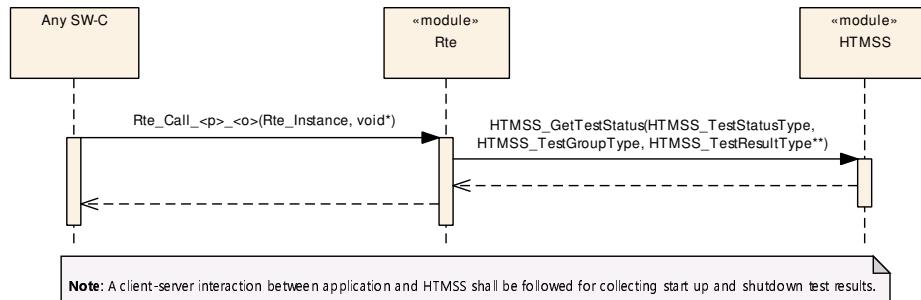


Figure 9.6: ECU shutdown with HTMSS integrated

## 9.7 Sequence diagram example of application SWC collecting the test results



**Figure 9.7: Application SWC collecting the test results**

## 10 Configuration specification

The following chapters summarize all configuration parameters. The detailed meanings of the parameters are described in the Chapters below.

### 10.1 Containers and configuration parameters

The following chapters summarize all configuration parameters. However, the content of this chapter is, intended to provide as an example for reference purposes.

The actual implementation is up to the specification user.

#### 10.1.1 HTMSS

##### [ECUC-HTMSS\_00001] Definition of EcucModuleDef HTMSS

*Status:* OBSOLETE

〔

<b>Module Name</b>	HTMSS
<b>Description</b>	Configuration of the Hardware Test Management start up and shutdown (HTMSS) module.
<b>Post-Build Variant Support</b>	false
<b>Supported Config Variants</b>	VARIANT-PRE-COMPIL

Included Containers		
Container Name	Multiplicity	Dependency
HTMSSConfigSet	1..*	This is the base container that contains the configuration parameters & sub containers of HTMSS module. <b>Tags:</b> atp.Status=obsolete
HTMSSGeneral	1	This container holds the general parameters of the HTMSS module. <b>Tags:</b> atp.Status=obsolete

〕

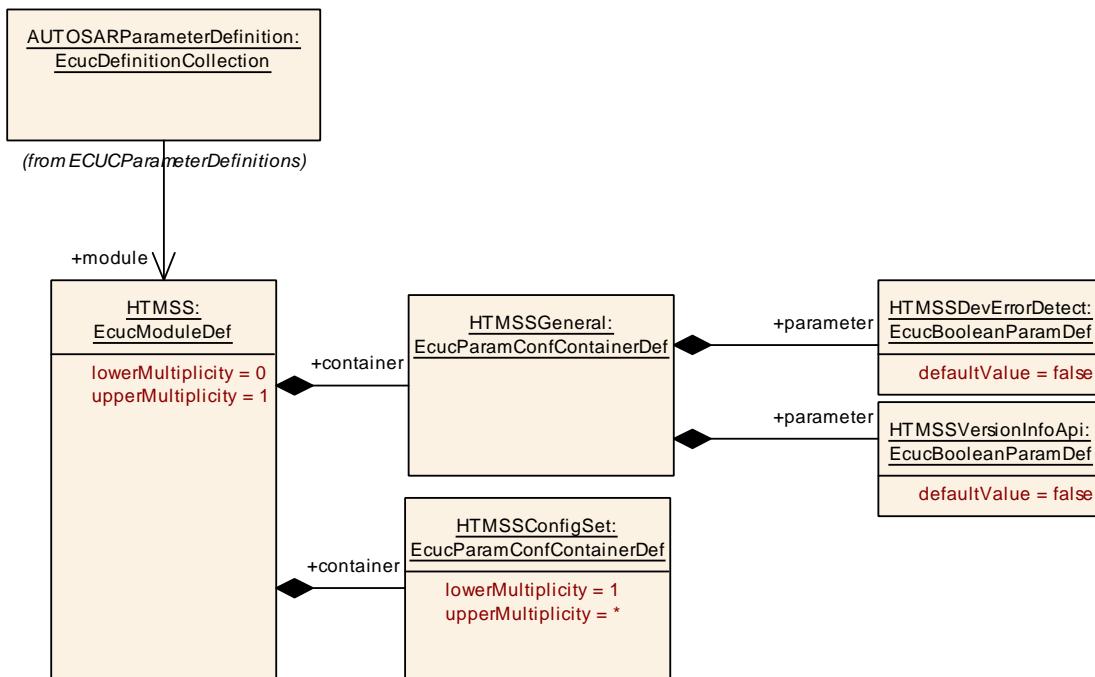


Figure 10.1: HTMSS ECUC Parameters

### 10.1.2 HTMSSGeneral

#### [ECUC\_HTMSS\_00619] Definition of EcucParamConfContainerDef HTMSSGeneral

*Status:* OBSOLETE

]

Container Name	HTMSSGeneral
Parent Container	HTMSS
Description	This container holds the general parameters of the HTMSS module. <b>Tags:</b> atp.Status=obsolete
Multiplicity	1
Configuration Parameters	

Included Parameters			
Parameter Name	Multiplicity	ECUC ID	
HTMSSDevErrorDetect	1	[ECUC_HTMSS_00002]	
HTMSSVersionInfoApi	1	[ECUC_HTMSS_00003]	

No Included Containers
------------------------

**[ECUC\_HTMSS\_00002] Definition of EcucBooleanParamDef HTMSSDevErrorDe-tect***Status:* OBSOLETE

[

Parameter Name	HTMSSDevErrorDetect		
Parent Container	<a href="#">HTMSSGeneral</a>		
Description	Switch for enabling the DET. <b>Tags:</b> atp.Status=obsolete		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	-	
	Post-build time	-	
Dependency			

]

**[ECUC\_HTMSS\_00003] Definition of EcucBooleanParamDef HTMSSVersionInfo Api***Status:* OBSOLETE

[

Parameter Name	HTMSSVersionInfoApi		
Parent Container	<a href="#">HTMSSGeneral</a>		
Description	Activate/Deactivate the version information API. <b>Tags:</b> atp.Status=obsolete		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	-	
	Post-build time	-	
Dependency			

]

**10.1.3 HTMSSConfigSet****[ECUC\_HTMSS\_00012] Definition of EcucParamConfContainerDef HTMSSConfig Set***Status:* OBSOLETE

[

Container Name	HTMSSConfigSet
Parent Container	HTMSS
Description	This is the base container that contains the configuration parameters & sub containers of HTMSS module. <b>Tags:</b> atp.Status=obsolete
Multiplicity	1..*
Configuration Parameters	

No Included Parameters

No Included Containers

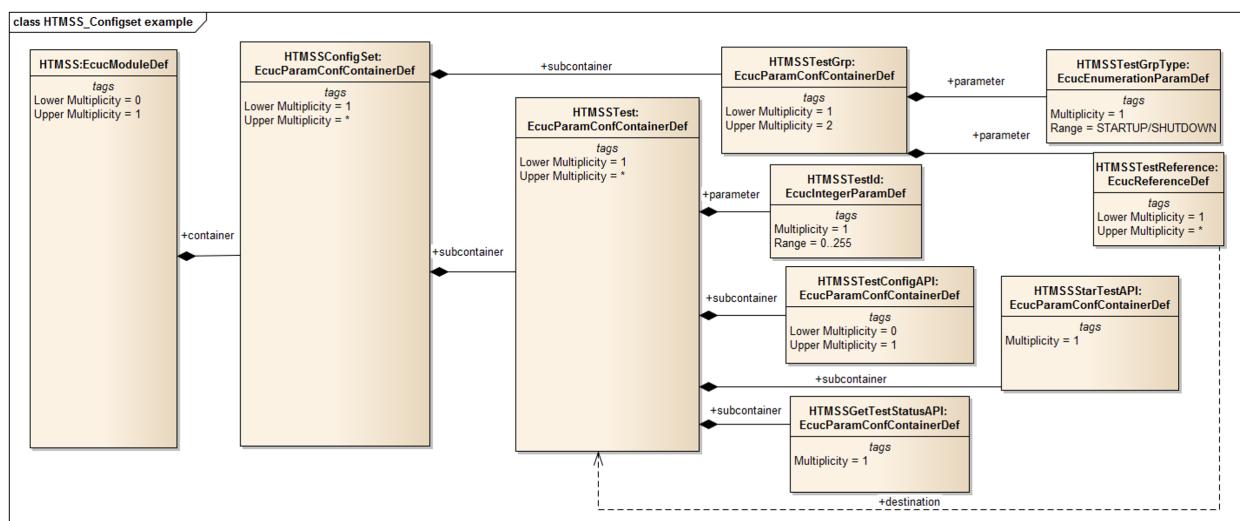


Figure 10.2: HTMSS Configuration

## 10.2 Published Information

For details refer to [5] Chapter 10.3 “Published Information”.

## A Not applicable requirements

### **[SWS\_HTMSS\_NA\_00999]**

[SRS\_BSW\_00406], [SRS\_BSW\_00416], [SRS\_BSW\_00417], [SRS\_BSW\_00419],  
[SRS\_BSW\_00422], [SRS\_BSW\_00425], [SRS\_BSW\_00432], [SRS\_BSW\_00452],  
[SRS\_BSW\_00403], [SRS\_BSW\_00461], [SRS\_BSW\_00466], [SRS\_BSW\_00469],  
[SRS\_BSW\_00470], [SRS\_BSW\_00471], [SRS\_BSW\_00472], [SRS\_BSW\_00478],  
[SRS\_BSW\_00490], [SRS\_BSW\_00492], [SRS\_BSW\_00399], [SRS\_BSW\_00396],  
[SRS\_BSW\_00395], [SRS\_BSW\_00393], [SRS\_BSW\_00392], [SRS\_BSW\_00390],  
[SRS\_BSW\_00389], [SRS\_BSW\_00388], [SRS\_BSW\_00385], [SRS\_BSW\_00383],  
[SRS\_BSW\_00375], [SRS\_BSW\_00369], [SRS\_BSW\_00336], [SRS\_BSW\_00170],  
[SRS\_BSW\_00168], [SRS\_BSW\_00458]

These requirements are not applicable to this specification.

## B Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyper-links in the document.

### B.1 Traceable item history of this document according to AUTOSAR Release R24-25

#### B.1.1 Added Specification Items in R25-11

none

#### B.1.2 Changed Specification Items in R25-11

[ECUC\_HTMSS\_00001] [ECUC\_HTMSS\_00002] [ECUC\_HTMSS\_00003] [ECUC\_HTMSS\_00012] [ECUC\_HTMSS\_00619] [SWS\_HTMSS\_00032] [SWS\_HTMSS\_00039] [SWS\_HTMSS\_00042] [SWS\_HTMSS\_00043] [SWS\_HTMSS\_00044] [SWS\_HTMSS\_91006] [SWS\_HTMSS\_91007] [SWS\_HTMSS\_91008] [SWS\_HTMSS\_91009] [SWS\_HTMSS\_91011]

#### B.1.3 Deleted Specification Items in R25-11

none