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

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.4.0

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2018-10-31	4.4.0	AUTOSAR Release Management	• Headerfile cleanup
2016-11-30	4.3.0	AUTOSAR Release Management	• Initial Release

Disclaimer

Table of Contents

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 is a basic software module at the service layer of the standardized basic software architecture of AUTOSAR.
The HTMSS 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 module integrates on the level of the AUTOSAR BSW service layer. Below figure shows the functional integration of the HTMSS module in AUTOSAR software platform.

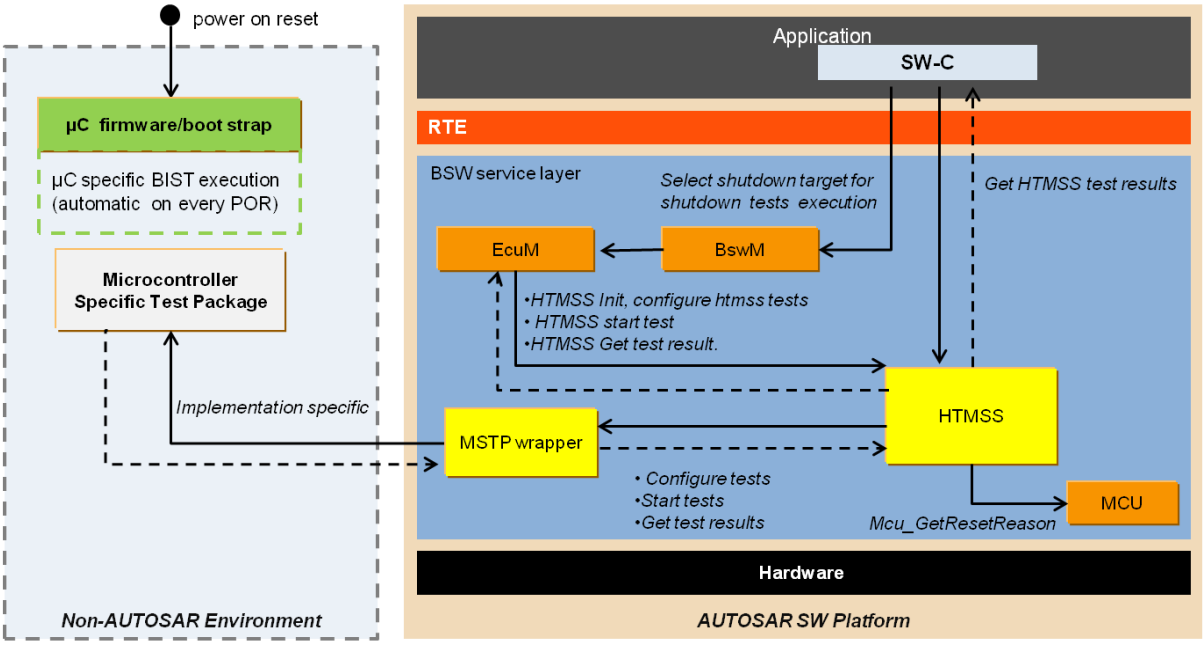


Figure 1 HTMSS interaction overview

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.

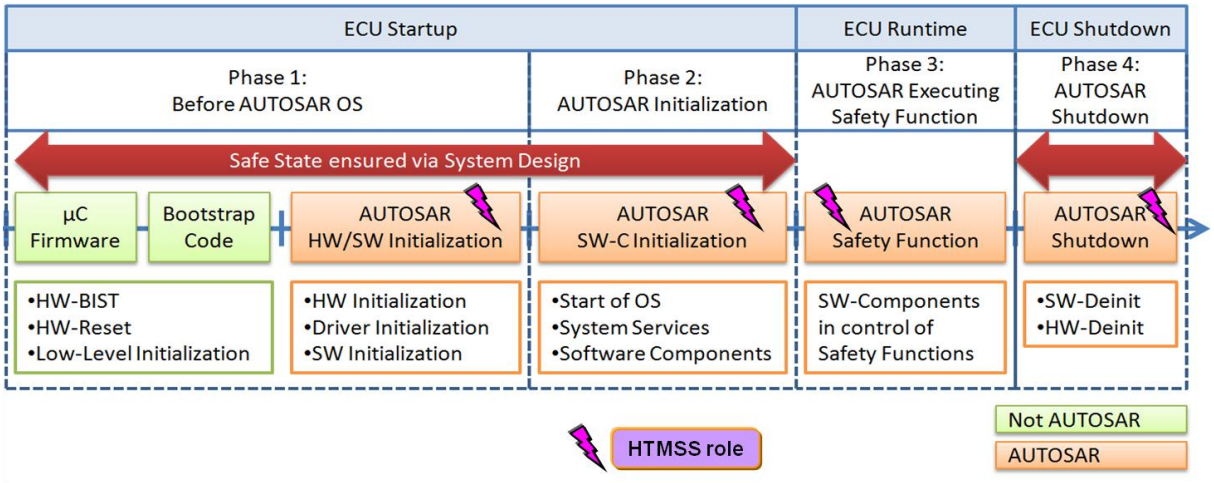


Figure 2 HTMSS phases overview

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.

- HTMSS Phase 1:

  Before AUTOSAR OS – 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.

  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 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

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

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_BasicSoftwareModules.pdf

[2] AUTOSAR Layered Software Architecture
AUTOSAR_LayeredSoftwareArchitecture.pdf

[3] AUTOSAR General Specification for Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

[4] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping.pdf

[5] Specification of RTE
AUTOSAR_SWS_RTE.pdf

[6] Specification of ECU state manager
AUTOSAR_SWS_ECUStateManager.pdf

[7] Requirements on HTMSS
AUTOSAR_SRS_HTMSS.pdf

[8] Technical Report on HTMSS
AUTOSAR_TR_HWTestManagementIntegrationGuide.pdf

3.2 Related standards and norms

[5] IEC 7498-1 The Basic Model, IEC Norm, 1994

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software (SWS BSW General) [3], which are also valid for HTMSS module
Thus, the specification SWS BSW General [3] shall be considered as additional and required specification for AUTOSAR HTMSS module.

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 and application SWC in the AUTOSAR architecture. 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 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) 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 data exchange the test result/status are shared between the HTMSS module and the application software layer.

5.4 Dependencies with MSTP

The HTMSS 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

[SWS_HTMSS_00001]⌈
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 traceability

The following table references relevant features specified in [3] and links to the fulfilments of these.

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_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_00384	The Basic Software Module specifications shall specify at least in the description which other modules they require	SWS_HTMSS_00040
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_HTMSS_00001	The HTMSS shall allow configuration of start up and shutdown tests	SWS_HTMSS_00014, SWS_HTMSS_00017, SWS_HTMSS_00023
SRS_HTMSS_00002	The HTMSS shall allow the configuration of tests at individual hardware resource level	SWS_HTMSS_00008, SWS_HTMSS_00009, SWS_HTMSS_00014, SWS_HTMSS_00017, SWS_HTMSS_00022
SRS_HTMSS_00003	The HTMSS shall provide a service to collect the MSTP tests results	noname, SWS_HTMSS_00005, SWS_HTMSS_00032, SWS_HTMSS_00033, SWS_HTMSS_00034, SWS_HTMSS_00036
SRS_HTMSS_00004	The HTMSS shall provide a mechanism to share the test results with the application layer software	SWS_HTMSS_00042
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
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
SRS_HTMSS_00007	HTMSS shall provide callout options to handle the test failure conditions	SWS_HTMSS_00043, SWS_HTMSS_00044

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]⌈
The HTMSS shall be able to read the microcontroller specific start up and shutdown test results/status, on the requested hardware.
⌋(SRS_HTMSS_00003)

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 sate.

[SWS_HTMSS_00006]⌈
The pre-compile time configuration parameters shall be checked statically (at least during compile time) for correctness. ⌋(SRS_BSW_00345)

[SWS_HTMSS_00008]⌈
In one configuration, there shall be more than one test per module for testing. These tests could be executed in parallel.
⌋(SRS_HTMSS_00002)

Note: The configuration parameter to handle above requirement shall be adapted in HTMSSConfigSet (Chapter 10.2.3)

[SWS_HTMSS_00009]⌈
It shall be possible to test the individual hardware resources (e.g. selected via module / channel ID) on the given hardware.
Refer Chapter 10.2.3 HTMSSConfigSet for configuration parameter implementations.

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.
⌋(SRS_HTMSS_00002)

7.2.3 States of HTMSS module

[SWS_HTMSS_00010]⌈

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

7.3.1 Development Errors

[SWS_HTMSS_00011]⌈

Type or error	*Relevance*	*Related error code*	*Value [hex]*
A service was called prior to initialization	Development	HTMSS_E_NOT_INIT	0x01
A null pointer was passed as an argument	Development	HTMSS_E_NULL_POINTER	0x02
A parameter was invalid (unspecific)	Development	HTMSS_E_PARAM_INVALID	0x03
Function called when test request is running	Development	HTMSS_E_BUSY	0x04

](SRS_BSW_00337)

7.3.2 Production Errors

None

8 API specification

8.1 Imported types

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

[SWS_HTMSS_00012][

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

Header file	***Imported Type***
Std_Types	Std_ReturnType
	Std_VersionInfoType

](SRS_BSW_00301)

8.2 Type definitions

[SWS_HTMSS_00013][
The following Data Types shall be used for the functions defined in this specification

8.2.1 HTMSS_TestCfgType

Name:	HTMSS_TestCfgType	
Type:	Structure	
Range:	Implementation specific	The content of the configuration data structure is implementation specific.
Description:	Configuration data structure of HTMSS module	
Available via:	HTMSS.h	

8.2.2 HTMSS_TestStatusType

Name:	HTMSS_TestStatusType	
Type:	enumeration	
Range:	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
Description:	HTMSS_TestStatusType describes status of test.	
Available via:	HTMSS.h	

8.2.3 HTMSS_TestGroupType

Name:	HTMSS_TestGroupType	
Type:	Enumeration	
	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
Description:	HTMSS_ TestGroupType describes the test group type	
Available via:	HTMSS.h	

8.2.4 HTMSS_TestResultType

Name:	HTMSS_TestResultType		
Type:	Struct		
	unit8	TestResult	The test result (e.g. pass, fail, invalid)
	uint8	TestSignature	The identifier of the tested resource
Description:	It describes the current test result		
Available via:	HTMSS.h		

](SRS_HTMSS_00001),(SRS_HTMSS_00002)

8.3 Function definitions

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

8.3.1 HTMSS_Init

[SWS_HTMSS_00014][

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

] (SRS_HTMSS_00001, SRS_HTMSS_00002)

[SWS_HTMSS_00015][
In case of Variant PB: The function HTMSS_Init shall initialize the HTMSS module and according to the configuration set referenced by ConfigPtr.] (SRS_BSW_00404)

[SWS_HTMSS_00016][
In case of Variant PC: The function HTMSS_Init shall initialize the HTMSS according to the pre-compile configuration set.] (SRS_BSW_00345, SRS_BSW_00159)

[SWS_HTMSS_00017][
The service HTMSS_Init() shall initialize the global variables and data structures of the HTMSS including flags and buffers.
] (SRS_HTMSS_00001, SRS_HTMSS_00002)

[SWS_HTMSS_00018][
The function HTMSS_Init() shall initialize MSTP module and configure the MSTP tests.
](SRS_HTMSS_00001, SRS_HTMSS_00002)

[SWS_HTMSS_00019][
The HTMSS is not functional until this function has been called.
] (SRS_HTMSS_00005)

[SWS_HTMSS_00020][
The function HTMSS_Init() shall determine the last reset reason by calling the Mcu_GetResetReason() of the MCU driver.
](SRS_HTMSS_00005)

[SWS_HTMSS_00021][
The function HTMSS_Init() shall configure the MSTP tests (both start up and shutdown), if the last reset reason is not MCU_HWTEST_RESET.
] (SRS_HTMSS_00005)

[SWS_HTMSS_00022][
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 (Refer section 8.2.3) can be used in this context.
] (SRS_HTMSS_00002)

[SWS_HTMSS_00023][
The function HTMSS_Init() shall set the HTMSS state to HTMSS_UNINIT, if the configuration of MSTP tests fails for any reason.
] (SRS_HTMSS_00001)

[SWS_HTMSS_00024][
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][

Service name:	HTMSS_StartTest
Syntax:	Std_ReturnType HTMSS_StartTest (HTMSS_TestGroupType GrpId)
Service ID [hex]:	0x03
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	GrpId The test group type (e.g. start up or shut down)

Parameters (inout):	
Parameters (out):	
Return value:	Std_ReturnType Standard return from function execution
Description:	Starts the MSTP configured tests
Available via:	HTMSS.h

] (SRS_HTMSS_00006)

[SWS_HTMSS_00026][
The function HTMSS_StartTest shall trigger the MSTP test operation on the requested hardware. If this is successful, E_OK shall be returned.
] (SRS_HTMSS_00006)

[SWS_HTMSS_00027][
The function HTMSS_StartTest shall set the HTMSS state to HTMSS_BUSY, if the MSTP status confirm that, test trigger is successful.] (SRS_HTMSS_00006)

[SWS_HTMSS_00028][
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.] (SRS_HTMSS_00006)

[SWS_HTMSS_00029][
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][
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][
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][

Service name:	HTMSS_GetTestStatus
Syntax:	HTMSS_TestStatusType HTMSS_GetTestStatus(HTMSS_TestGroupType GrpId, HTMSS_TestResultType * RequestTestResultPtr)
Service ID [hex]:	0x04
Sync/Async:	Synchronous

Reentrancy:	Non Reentrant	
Parameters (in):	GrpId	The test group type (e.g. start up or shutdown)
Parameters (inout):	None	
Parameters (out):	RequestTestResultPtr	Pointer to store the request result
Return value:	HTMSS_TestStatusType	
Description:	Returns current test status on requested test	
Available via:	HTMSS.h	

] (SRS_HTMSS_00003), (SRS_HTMSS_00004)

[SWS_HTMSS_00033][
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.
] (SRS_HTMSS_00003)

[SWS_HTMSS_00034][
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.
] (SRS_HTMSS_00003)

SWS_HTMSS_00035][
The function HTMSS_GetTestStatus shall set the HTMSS state to HTMSS_IDLE, if the MSTP provided status confirm the test completion.](SRS_HTMSS_00003)

[SWS_HTMSS_00036][
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.
](SRS_HTMSS_00003), (SRS_HTMSS_000010)

[SWS_HTMSS_00037][
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][
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][

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

] (SRS_BSW_00407)

8.4 Call-back 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][

API function	***Header File***	***Description***
Mcu_GetResetReason	Mcu.h	Service to read the reset type from the hardware, if supported.

] (SRS_BSW_00384)

8.6.2 Optional Interfaces

This chapter defines all interfaces, which are required to fulfill an optional functionality of the module.
[SWS_HTMSS_00041][

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][

Name:	GetTestStatus	
Comment	--	
IsService:	True	
Variation		
Possible Errors	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

GetTestStatus		
Comments	--	
Variation		
Parameters	GrpId	
	Comment	The test group type (e.g. start up or shut down)
	Type	HTMSS_TestGroupType
	Variation	--
	Direction	IN
	TestResultPtr	
	Comment	Pointer to provide Test results along with Test
	Type	HTMSS_TestResultType
	Variation	--
	Direction	OUT
Possible Errors	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

] (SRS_HTMSS_00004)

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_StartupTestErrorHook

[SWS_HTMSS_00043][

Service name:	HTMSS_ StartupTestErrorHook
Syntax:	void HTMSS_ StartupTestErrorHook(void)
Service ID [hex]:	0x07
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	none
Parameters (inout):	none
Parameters (out):	none
Return value:	none
Description:	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...
Available via:	HTMSS.h

] (SRS_HTMSS_00007)

8.8.2 HTMSS_ShutdownTestErrorHook

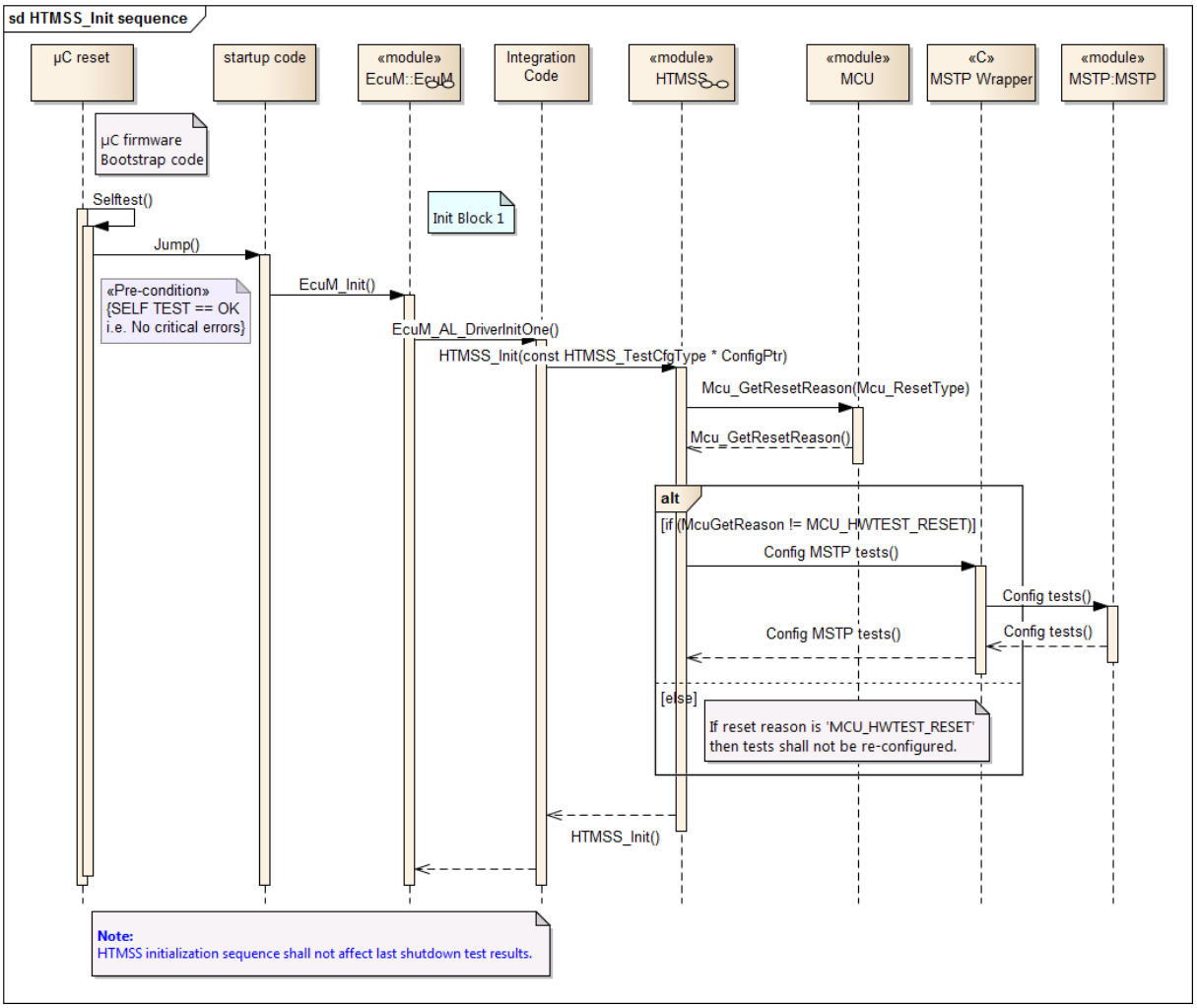
[SWS_HTMSS_00044][

Service name:	HTMSS_ ShutdownTestErrorHook
Syntax:	void HTMSS_ ShutdownTestErrorHook(void)
Service ID [hex]:	0x08
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	none
Parameters (inout):	none
Parameters (out):	none
Return value:	none
Description:	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...
Available via:	HTMSS.h

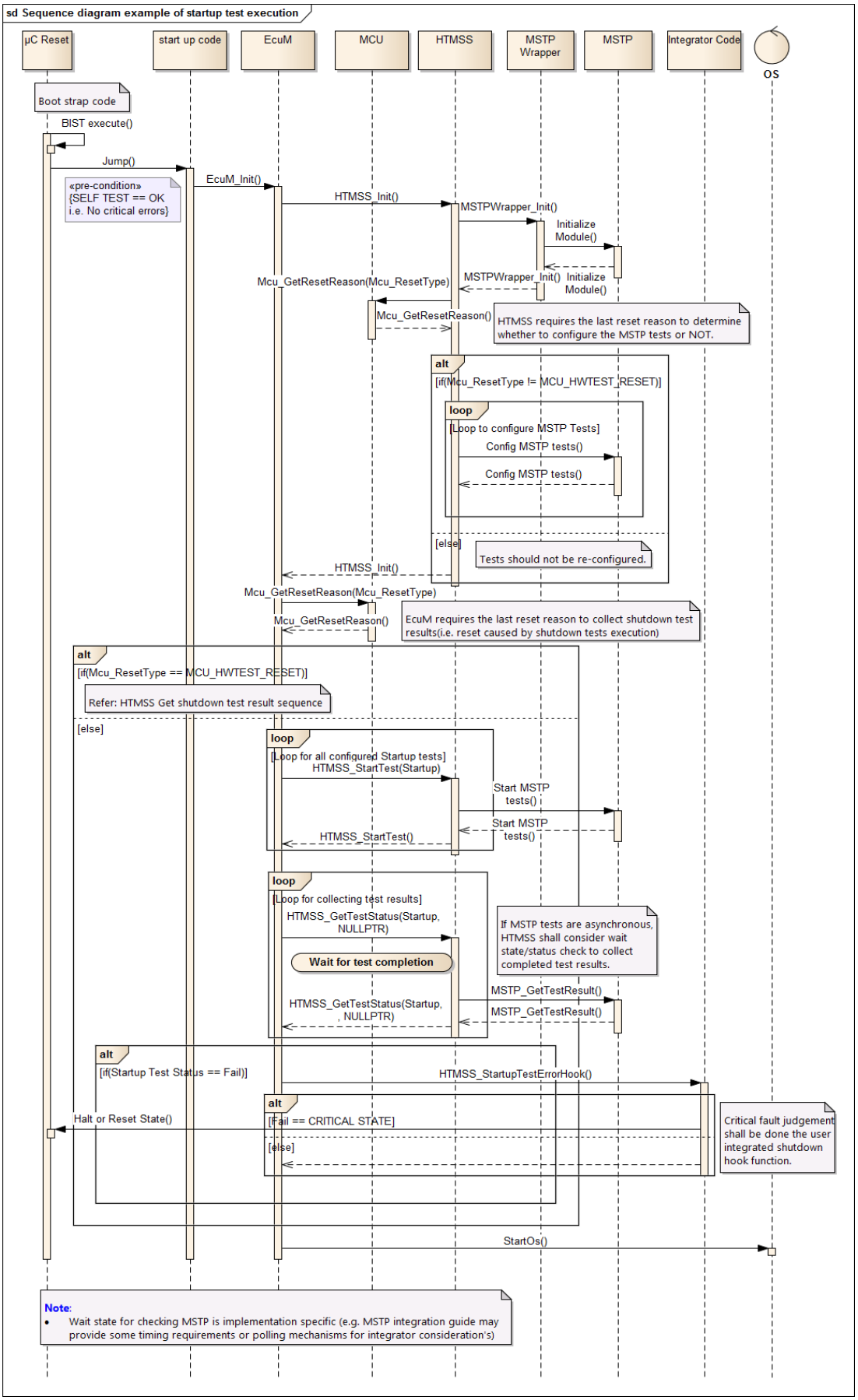
] (SRS_HTMSS_00007)

9 Sequence diagrams

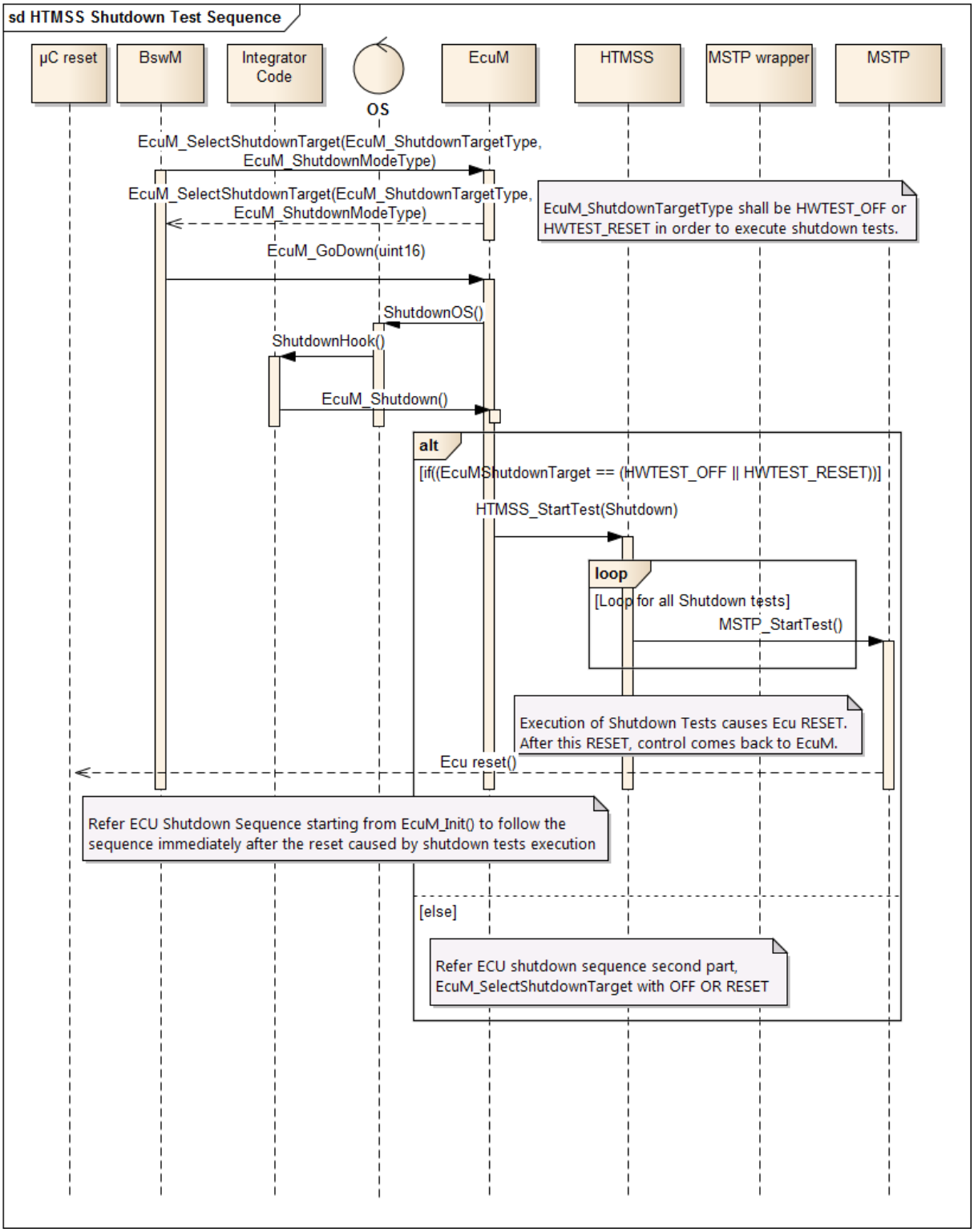
9.1.1 Sequence diagram example of HTMSS Initialization



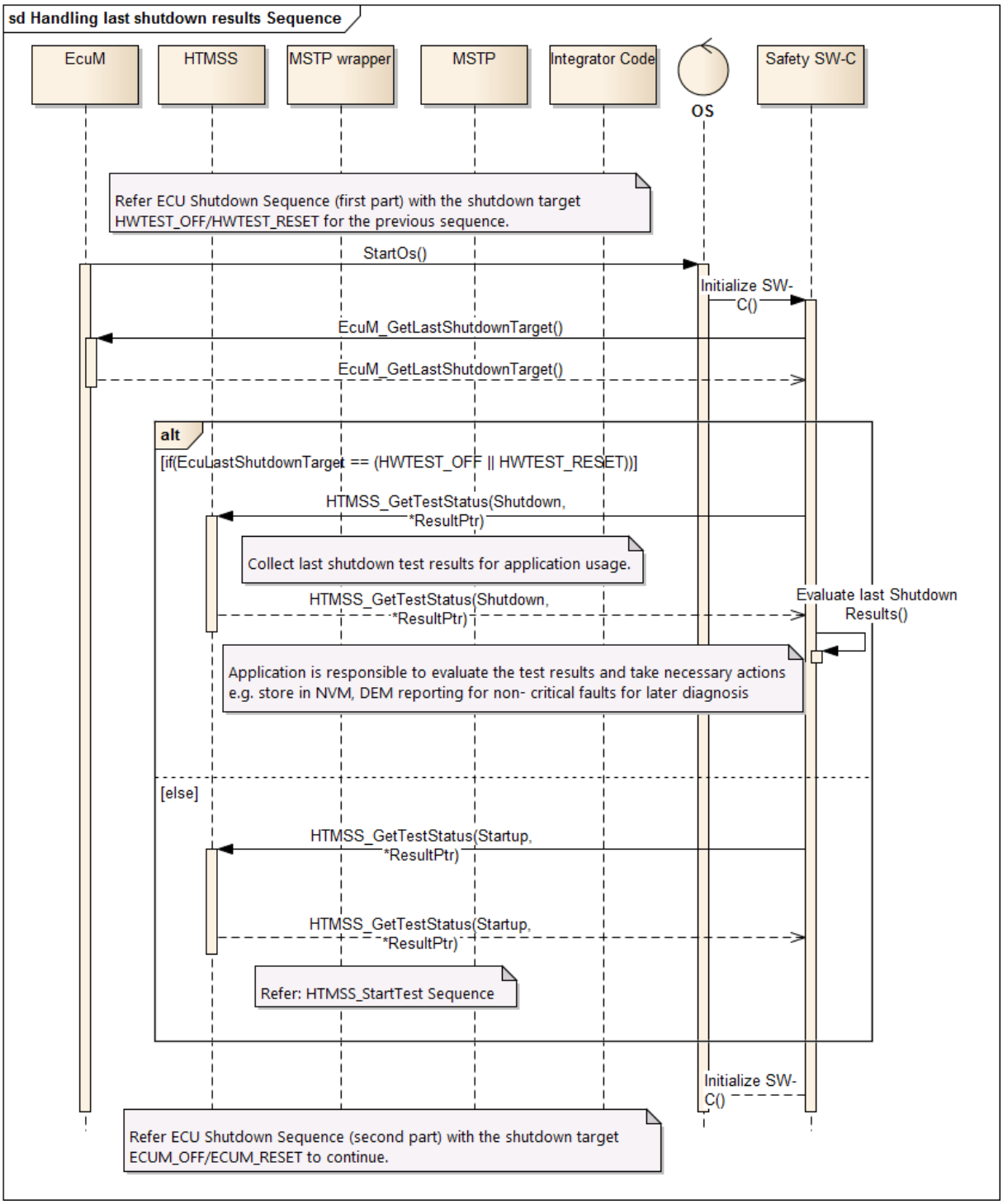
9.1.2 Sequence diagram example of startup test execution



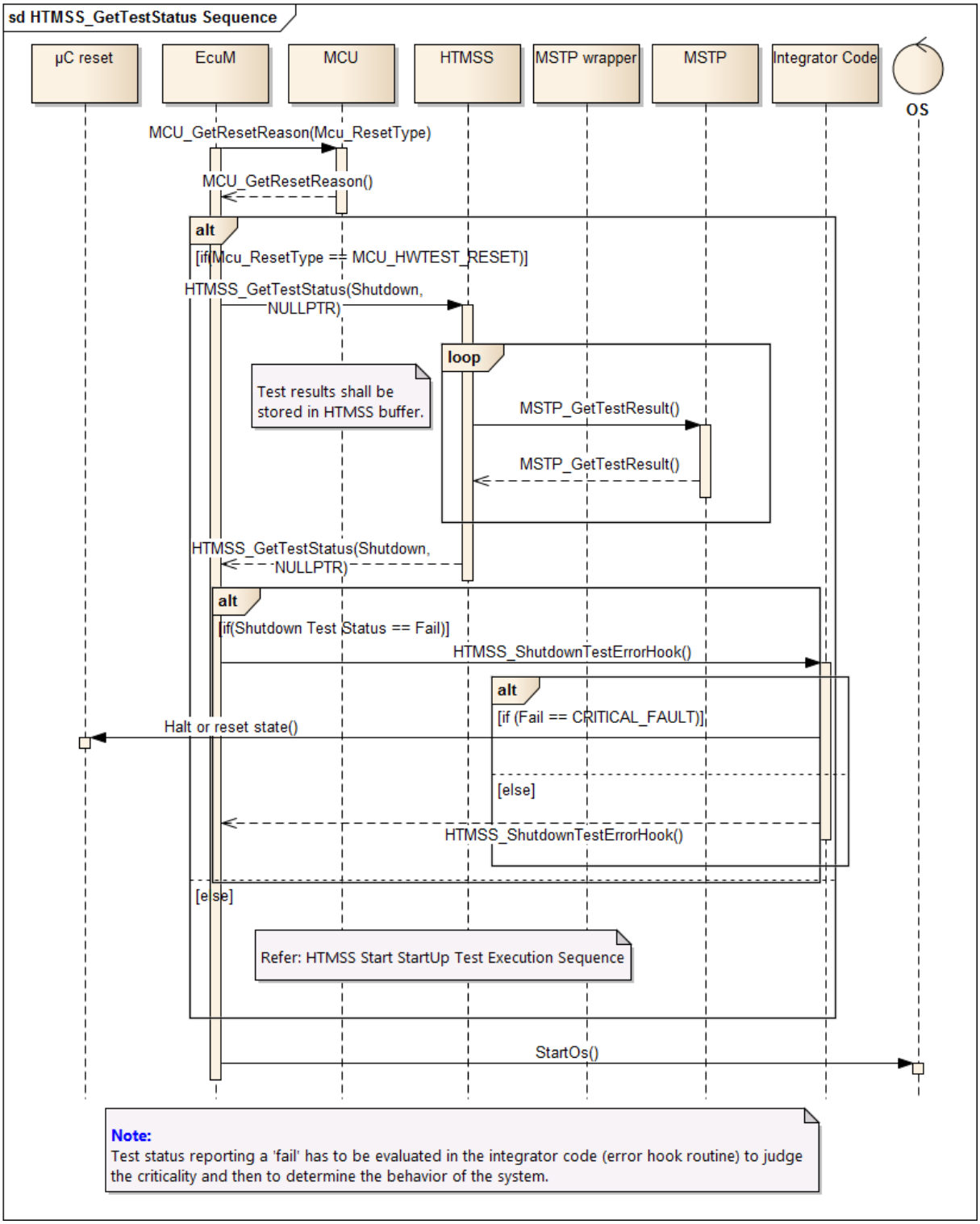
9.1.3 Sequence diagram example of shutdown test execution



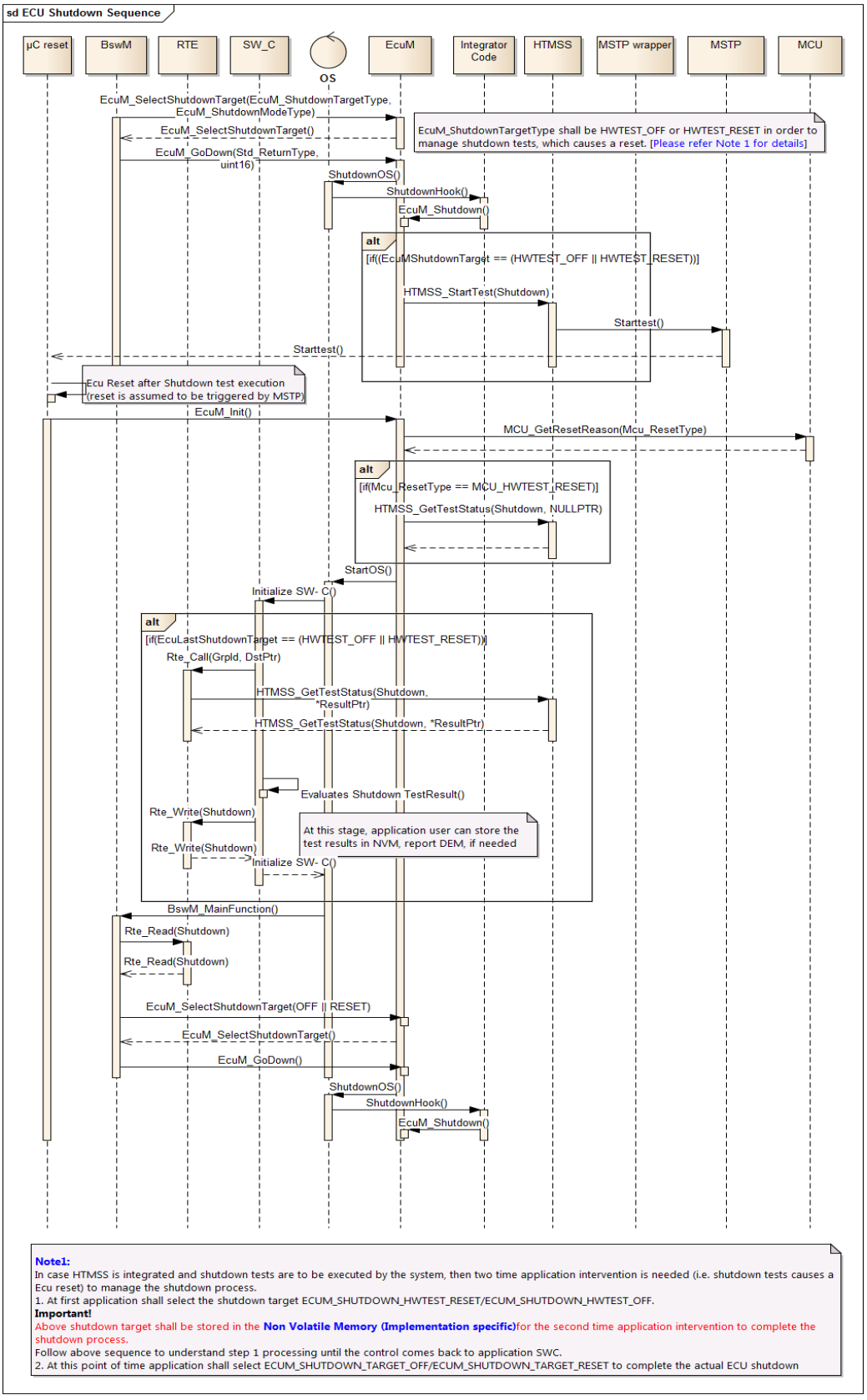
9.1.4 Sequence diagram example handling the last shutdown test results immediately after the ECU reset raised by MSTP module



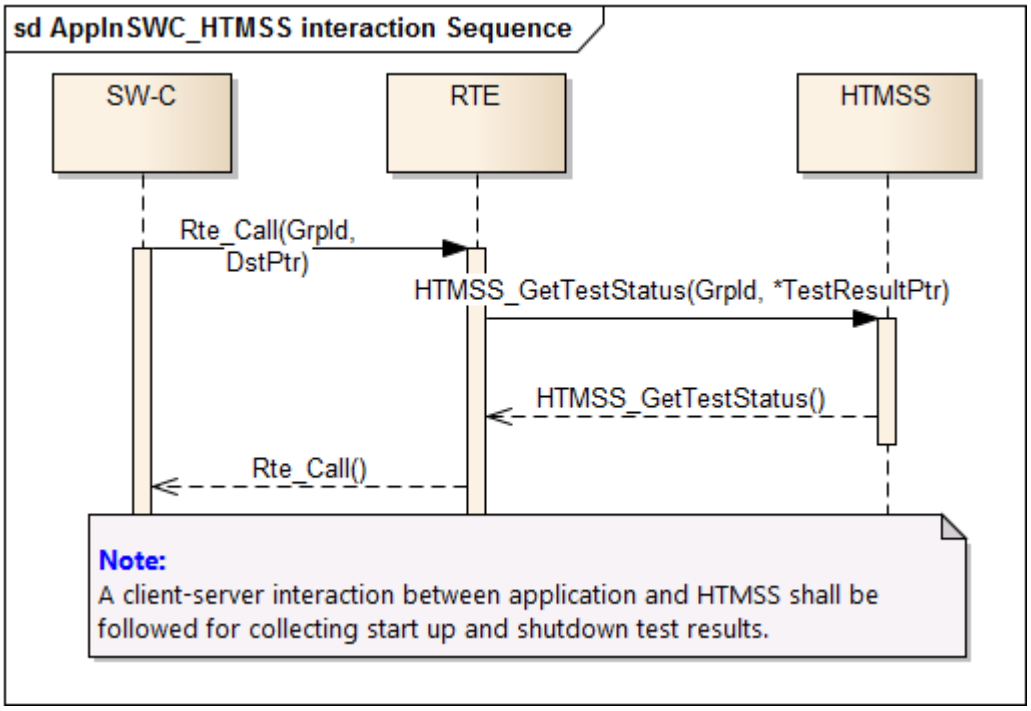
9.1.5 Sequence diagram example of collecting the shutdown test results



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



9.1.7 Sequence diagram example of 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.

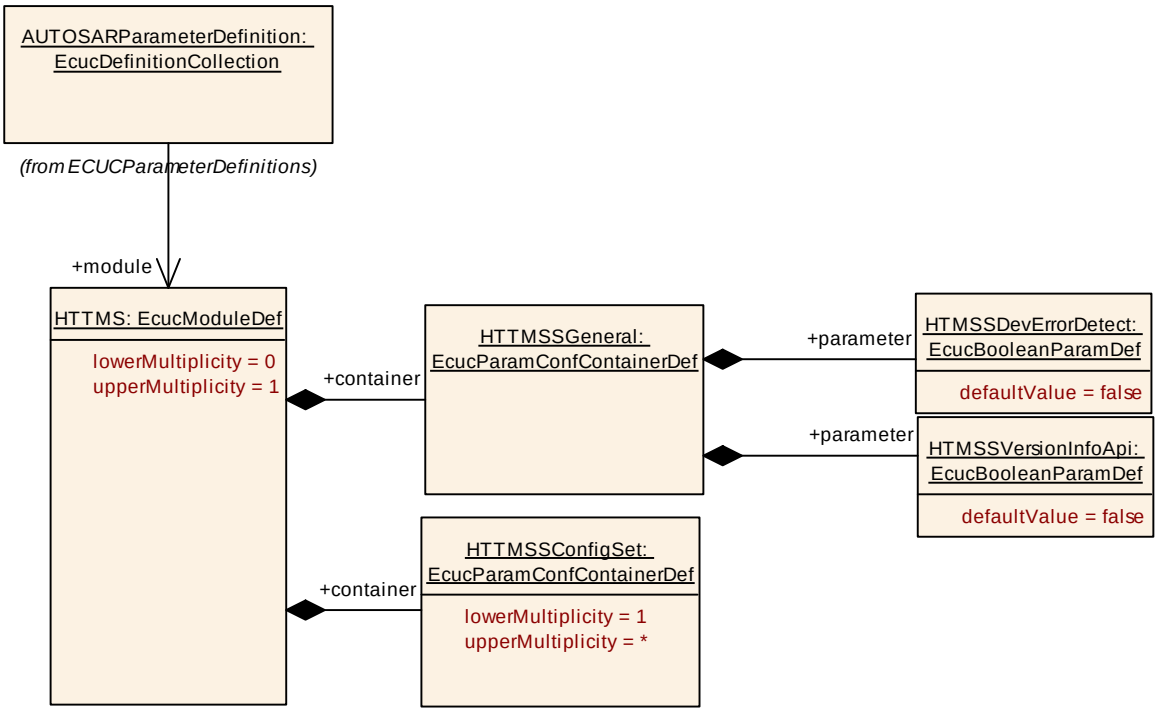
10.1 Containers and configuration parameters

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 HTTMS

SWS Item	**ECUC_HTTMS_00001 :**
Module Name	*HTTMS*
Module Description	Configuration of the Hardware Test Management start up and shutdown (HTMSS) module.
Post-Build Variant Support	false
Supported Config Variants	VARIANT-PRE-COMPILE

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
HTTMSSConfigSet	1..*	This is the base container that contains the configuration parameters & sub containers of HTMSS module.
HTTMSSGeneral	1	This container holds the general parameters of the HTMSS module.



10.1.2 HTTMSSGeneral

SWS Item	**ECUC_HTTMS_00619 :**
Container Name	HTTMSSGeneral
Description	This container holds the general parameters of the HTMSS module.

Configuration Parameters

SWS Item	**ECUC_HTTMS_00002 :**		
Name	HTMSSDevErrorDetect		
Parent Container	HTTMSSGeneral		
Description	Switch for enabling the DET.		
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	--	
Scope / Dependency	scope: local		

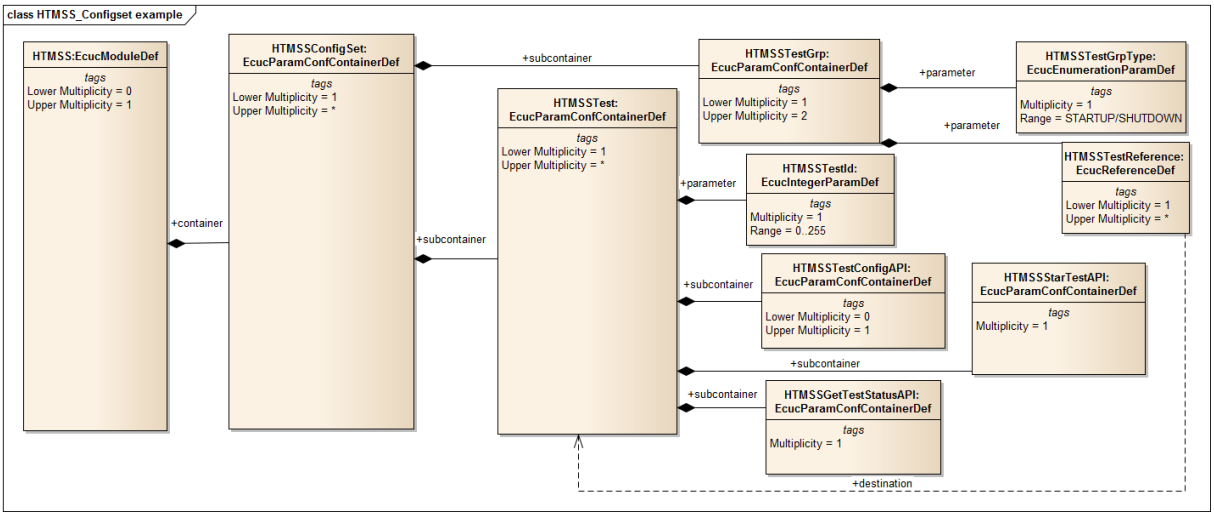
SWS Item	**ECUC_HTTMS_00003 :**		
Name	HTMSSVersionInfoApi		
Parent Container	HTTMSSGeneral		
Description	Activate/Deactivate the version information API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.1.3 HTTMSSConfigSet

SWS Item	**ECUC_HTTMS_00012 :**
Container Name	HTTMSSConfigSet
Description	This is the base container that contains the configuration parameters & sub containers of HTMSS module.
Configuration Parameters	

No Included Containers



10.2 Published Information

For details, refer to the chapter 10.3 “Published Information” in SWS_BSWGeneral.