Document Title	Specification of Core Test
Document Category	Software specification (SWS)
Document UID	259
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Removed the CorTst_GetVersionInfo API and its configuration parameter. • Classified CORTST_E_CORE_FAILURE as a production error and updated the related reporting requirements. • Aligned the API chapter structure with the updated SWS template. • Updated configuration information and traceability.
2025-11-27	R25-11	AUTOSAR Release Management	• Editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	• Removed [SWS_CorTst_01007] • Editorial changes
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes
2022-11-24	R22-11	AUTOSAR Release Management	• Changed [SWS_CorTst_00999] to [SWS_CorTst_NA_00999]
2021-11-25	R21-11	AUTOSAR Release Management	• Artifact inclusion based on ArtifactAnalysis corrected.
2020-11-30	R20-11	AUTOSAR Release Management	• Clean up of Error Classification chapter
2019-11-28	R19-11	AUTOSAR Release Management	• Incorporated changes to support MCALMulticoreDistribution • Changed Document Status from Final to published

△

2018-10-31	4.4.0	AUTOSAR Release Management	• Incorporated changes to support MCALMulticoreDistribution (Draft)
2017-12-08	4.3.1	AUTOSAR Release Management	• Minor corrections
2016-11-30	4.3.0	AUTOSAR Release Management	• Replaced Development Error Tracer with Default Error Tracer • Removed Debugging Support section • Removed Variants section
2015-07-31	4.2.2	AUTOSAR Release Management	• Correction of `CorTst_Init` prototype • Added `CorTst_ConfigType` and `CorTst_ResultType` • Debugging support marked as obsolete • Minor corrections
2014-10-31	4.2.1	AUTOSAR Release Management	• CORTST_E_CORE_FAILURE extended production error formalization, including healing. • Correction of `CorTst_GetCurrentStatus` prototype
2013-10-31	4.1.2	AUTOSAR Release Management	• Removed timing attribute of required [SWS_CorTst_00067] • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Alignment to new SWS_BSWGeneral document • Updated document for Extended Production Errors • Alignment to official naming in other Autosar documents
2011-12-22	4.0.3	AUTOSAR Administration	• Clarification of some requirements. • Typos correction. • Removed redundant and useless requirements.

▽

△

2010-09-30	3.1.5	AUTOSAR Administration	• Added new requirements for configuration and error detection. • Clarification of some requirements. • Added new configuration parameters. • Removed obsolete requirements. • Improvement of static error detection. • Removed unsued types.
2010-02-02	3.1.4	AUTOSAR Administration	• 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 7

2 Acronyms and Abbreviations 8

3 Related documentation 9
- 3.1 Input documents & related standards and norms . . . 9
- 3.2 Related specification . . . 9

4 Constraints and assumptions 10
- 4.1 Limitations . . . 10
- 4.2 Applicability to car domains . . . 10
- 4.3 Applicability to safety related environments . . . 10

5 Dependencies to other modules 11
- 5.1 File structure . . . 11
  - 5.1.1 Code file structure . . . 11

6 Requirements Tracing 12

7 Functional specification 15
- 7.1 General Behavior . . . 15
  - 7.1.1 Background & Rationale . . . 17
- 7.2 Error Classification . . . 17
  - 7.2.1 Development Errors . . . 18
  - 7.2.2 Runtime Errors . . . 18
  - 7.2.3 Production Errors . . . 19
    - 7.2.3.1 CORTST_E_CORE_FAILURE . . . 19
- 7.3 General Requirements . . . 19
- 7.4 Security Events . . . 19

8 API specification 20
- 8.1 Imported types . . . 20
- 8.2 Type definitions . . . 20
  - 8.2.1 CorTst_ConfigType . . . 20
  - 8.2.2 CorTst_CsumSignatureType . . . 21
  - 8.2.3 CorTst_CsumSignatureBgndType . . . 21
  - 8.2.4 CorTst_ErrOkType . . . 22
  - 8.2.5 CorTst_ResultType . . . 22
  - 8.2.6 CorTst_StateType . . . 23
  - 8.2.7 CorTst_TestIdFgndType . . . 23
- 8.3 Function definitions . . . 23
  - 8.3.1 CorTst_Init . . . 24
  - 8.3.2 CorTst_DeInit . . . 25
  - 8.3.3 CorTst_Abort . . . 26
  - 8.3.4 CorTst_GetState . . . 27

8.3.5 CorTst_GetCurrentStatus . . . 27
8.3.6 CorTst_GetSignature . . . 28
8.3.7 CorTst_GetFgndSignature . . . 29
8.3.8 CorTst_Start . . . 29
8.4 Scheduled functions . . . 30
8.5 Expected interfaces . . . 32
8.5.1 Mandatory interfaces . . . 32
8.5.2 Optional interfaces . . . 33
8.5.3 Configurable interfaces . . . 33
9 Sequence diagrams 35
9.1 Initialization . . . 35
9.2 Deinitialization . . . 35
9.3 Background Test . . . 36
9.3.1 Test Result Calculation within Core Test Module . . . 36
9.3.2 Core Test Signature provided to Calling Entity . . . 37
10 Configuration specification 38
10.1 How to read this chapter . . . 38
10.2 Containers and configuration parameters . . . 38
10.2.1 CorTst . . . 38
10.2.2 CorTstGeneral . . . 39
10.2.3 CorTstSelect . . . 42
10.2.4 CorTstBackgroundConfigSet . . . 45
10.2.5 CorTstForegroundConfigSet . . . 46
10.2.6 CorTstConfigApiServices . . . 47
10.2.7 CorTstDemEventParameterRefs . . . 50
10.3 Published Information . . . 51
A Not applicable requirements 52
B Change history of AUTOSAR traceable items 53
B.1 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 53
B.1.1 Added Specification Items in R25-11 . . . 53
B.1.2 Changed Specification Items in R25-11 . . . 53
B.1.3 Deleted Specification Items in R25-11 . . . 53
B.2 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 53
B.2.1 Added Specification Items in R26-11 . . . 53
B.2.2 Changed Specification Items in R26-11 . . . 53
B.2.3 Deleted Specification Items in R26-11 . . . 55

1 Introduction and functional overview

This specification specifies the functionality, API and configuration of the AUTOSAR Basic Software module called Core Test Driver. This specification is applicable to drivers for all kind of cores regardless if the driver is executing during power-on situations of an ECU or during ECU application runtime.

The Core Test Driver provides services for configuring, starting, polling, terminating and notifying the application about Core Test results. It also provides services for returning test results in a predefined way. Furthermore it provides several tests to verify dedicated core functionality like e.g. general purpose registers or Arithmetical and Logical Unit (ALU). It is assumed that every tested core hardware functionality can be exclusively accessed for testing purposes. It is up to the user of Core Test Driver API to choose suitable test combination and a scheduled execution order to fulfill the safety requirements of the system. The behaviour of those services is asynchronous or synchronous.

A Core Test driver accesses the microcontroller core directly without any intermediate software layers and is located in the Microcontroller Abstraction Layer (MCAL).

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the CoreTest module that are not included in the [1, AUTOSAR glossary].

Abbreviation / Acronym:	*Description:*
MCAL	Microcomputer Abstraction Layer
DEM	Diagnostic Event Manager
DET	Default Error Tracer
CPU	Central Processing Unit
MPU	Memory Protection Unit
L1	1st level memory
L2	2nd level memory
MCU	Microcontroller Unit
BIST	Built in Self Test
IRQ	Interrupt Request
Core	A CPU plus closely located functional resources
CSUM/Checksum/signature	A numerical representation of the result of a test execution.
Background test	Background test is called periodically by a SW-scheduler/RTOS.
Foreground test	A foreground test is a synchronous test and shall not be interrupted. It is called via user application calls.
'Golden (Ref.) Value'	Reference value used for comparison (e.g. Checksum/Signature) to a previously computed test result value.
'Good Case'	The execution finished without reporting an error
Atomic sequence/ atomic piece	An atomic sequence is a piece of test which shall not be interrupted.
External device	A physical external entity; e.g. a second microcontroller
Resource	A 'hardware resource' is the smallest unit (instance) that can be selected by a CORETest driver user. It can be tested in one or several atomic sequences. It is a core internal unit which executes a unique functionality (e.g. IRQ-controller).
Partial test (orange block in Figure3)	A partial test is defined as the test of one or more 'hardware resources'. (A partial test is interruptible because it is executed in background mode).
Entity/unit	Hardware functionality inside the core (e.g. CPU, MMU etc.)
Caller/calling entity	The caller/calling entity is located on a higher AUTOSAR or ISO layer [2]. It is the user of the API call.
test interval	CoreTest test Interval: the sum of all the partial tests (executed in background mode) on the hardware resources that are configured to make one complete Core test.
Test Interval Id	Identifier of a test interval, which shall be incremented by each start of a new test interval.

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

As this is a document from professionals for professionals, all other terms are expected to be known.

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] Layered Software Architecture
AUTOSAR_CP_EXP_LayeredSoftwareArchitecture

[3] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[4] Requirements on Core Test
AUTOSAR_CP_RS_CoreTest

[5] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[6] General Requirements on SPAL
AUTOSAR_CP_RS_SPALGeneral

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [3, SWS BSW General], which is also valid for CoreTest.

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

4 Constraints and assumptions

4.1 Limitations

A Core test module implementation might be limited to be executed during power-up/start-up time where core resources are not shared among different active AUTOSAR related software tasks or hardware-entities (e.g. IRQ-controller, DMA, Cache, MMU/MPU and MemoryIF)

-**OR**-

might be limited to test resources which are not shared during runtime software execution (e.g. ALU and CPU-registers). This is overall automotive system architecture dependent and cannot be covered in a MCAL Core Test SWS specification.

There must be a managing entity or architecture available who manages tasks like 'hardware-resource-access-managing' due to the inability of a MCAL-driver to handle such tasks on its own.

4.2 Applicability to car domains

No restrictions.

4.3 Applicability to safety related environments

This module can be used within safety related systems if the upper layer software provides mechanisms to handle the Core Test API results by:

- Checksum/signature protection
- Checking Core Test code integrity before using it
- Redundant storage of Checksum/signature
- External decision execution of Core Test results

and the Core Test module implementation is embedded into a system safety architecture concept.

5 Dependencies to other modules

The CoreTest module depends on the following modules:

- BSW scheduler is required to trigger main function in background mode

The Core Test library module and/or source code module is dependent on the microcontroller platform and therefore on the silicon manufacturers hardware implementation and even on a silicon revision.

The Core Test library module and/or source code module is dependent on an actively working core clock domain.

5.1 File structure

5.1.1 Code file structure

[SWS_CorTst_00002]

Upstream requirements: SRS_BSW_00164, SRS_CoreTst_14105

⌈The Core Test module shall provide interrupt service routines for test purposes only.⌋

6 Requirements Tracing

The following tables reference the requirements specified in [4], [5] and [6] 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_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_CorTst_00040] [SWS_CorTst_00041]
[SRS_BSW_00164]	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	[SWS_CorTst_00002]
[SRS_BSW_00304]	All AUTOSAR Basic Software Modules shall use only AUTOSAR data types instead of native C data types	[SWS_CorTst_00027]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_CorTst_00161]
[SRS_BSW_00327]	Error values naming convention	[SWS_CorTst_00016]
[SRS_BSW_00331]	All Basic Software Modules shall strictly separate error and status information	[SWS_CorTst_00037] [SWS_CorTst_00038] [SWS_CorTst_00039]
[SRS_BSW_00336]	Basic SW module shall be able to shutdown	[SWS_CorTst_00045] [SWS_CorTst_00046]
[SRS_BSW_00337]	Classification of development errors	[SWS_CorTst_00016]
[SRS_BSW_00339]	Reporting of production relevant error status	[SWS_CorTst_00154] [SWS_CorTst_00155] [SWS_CorTst_00177] [SWS_CorTst_01000] [SWS_CorTst_01001] [SWS_CorTst_01002]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/ disabling of detection and reporting of development errors.	[SWS_CorTst_00183]
[SRS_BSW_00357]	For success/failure of an API call a standard return type shall be defined	[SWS_CorTst_00064]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_CorTst_00040]
[SRS_BSW_00359]	Callback Function Return Types for AUTOSAR BSW	[SWS_CorTst_00076]
[SRS_BSW_00360]	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	[SWS_CorTst_00076]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_CorTst_00183]
[SRS_BSW_00385]	List possible error notifications	[SWS_CorTst_00016] [SWS_CorTst_01000]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_CorTst_01000]
[SRS_BSW_00406]	API handling in uninitialized state	[SWS_CorTst_00040] [SWS_CorTst_00044]



△

Requirement	Description	Satisfied by
[SRS_BSW_00409]	All production code error ID symbols are defined by the Dem module and shall be retrieved by the other BSW modules from Dem configuration	[SWS_CorTst_00154] [SWS_CorTst_00155] [SWS_CorTst_01001] [SWS_CorTst_01002]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_CorTst_00040] [SWS_CorTst_01003] [SWS_CorTst_01004]
[SRS_BSW_00422]	Pre-de-bouncing of error status information is done within the Dem	[SWS_CorTst_00154] [SWS_CorTst_00155] [SWS_CorTst_01000] [SWS_CorTst_01001] [SWS_CorTst_01002]
[SRS_BSW_00433]	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	[SWS_CorTst_00067]
[SRS_BSW_00469]	Fault detection and healing of production errors	[SWS_CorTst_00154] [SWS_CorTst_00155] [SWS_CorTst_01000] [SWS_CorTst_01001] [SWS_CorTst_01002]
[SRS_BSW_00471]	Do not cause dead-locks on detection of production errors - the ability to heal from previously detected production errors	[SWS_CorTst_01001] [SWS_CorTst_01002]
[SRS_CoreTst_14104]	Core Register Test Shall Be Available	[SWS_CorTst_00008]
[SRS_CoreTst_14105]	Core Interrupt and Exception Detection Tests Shall Be Available	[SWS_CorTst_00002] [SWS_CorTst_00009]
[SRS_CoreTst_14106]	Core ALU Test Shall Be Available	[SWS_CorTst_00010]
[SRS_CoreTst_14107]	Core Address Generator Test Shall Be Available	[SWS_CorTst_00011]
[SRS_CoreTst_14108]	Core Memory Interfaces Test Shall Be Available	[SWS_CorTst_00012]
[SRS_CoreTst_14109]	Memory Management/Protection Unit (MMU/MPU) Test Shall Be Available	[SWS_CorTst_00013]
[SRS_CoreTst_14110]	Cache Controller Test Shall Be Available	[SWS_CorTst_00014]
[SRS_CoreTst_14112]	There Shall Be a Single API for the Core Test Service	[SWS_CorTst_00064] [SWS_CorTst_00067] [SWS_CorTst_00144]
[SRS_CoreTst_14113]	The API Shall Have a Parameter to Select Which Component Shall Be Tested	[SWS_CorTst_00064] [SWS_CorTst_00160]
[SRS_CoreTst_14114]	A Main Function for the Core Test Shall Be Available	[SWS_CorTst_00067] [SWS_CorTst_00144]
[SRS_CoreTst_14115]	Test Metrics Shall Be Available to Caller	[SWS_CorTst_00057] [SWS_CorTst_00060]
[SRS_CoreTst_14116]	A Service shall be provided which returns a checksum/signature as test result	[SWS_CorTst_00057] [SWS_CorTst_00058] [SWS_CorTst_00060] [SWS_CorTst_00061] [SWS_CorTst_00176]
[SRS_CoreTst_14117]	Faults Shall Be Treated as Production Errors	[SWS_CorTst_00016] [SWS_CorTst_00021]
[SRS_CoreTst_14118]	The results of the Core test module shall be provided to the user	[SWS_CorTst_00053] [SWS_CorTst_00054]
[SRS_CoreTst_14119]	A Notification of Completion Shall Be Provided	[SWS_CorTst_00076] [SWS_CorTst_00077]
[SRS_CoreTst_14126]	It Shall Be Possible to Cancel a Running Test	[SWS_CorTst_00048] [SWS_CorTst_00050]
[SRS_CoreTst_14130]	Destructive Test Shall Restore Original State of tested Entity	[SWS_CorTst_00026]

▽

△

Requirement	Description	Satisfied by
[SRS_CoreTst_14131]	A Service shall be provided which returns a Pass/Fail Status Representation as a test result	[SWS_CorTst_00055] [SWS_CorTst_00056] [SWS_CorTst_01005]
[SRS_CoreTst_14133]	Each Core Test interval shall have an identifier	[SWS_CorTst_00137] [SWS_CorTst_00139]
[SRS_SPAL_00157]	All drivers and handlers of the AUTOSAR Basic Software shall implement notification mechanisms of drivers and handlers	[SWS_CorTst_00077]
[SRS_SPAL_12057]	All driver modules shall implement an interface for initialization	[SWS_CorTst_00041] [SWS_CorTst_00179]
[SRS_SPAL_12125]	All driver modules shall only initialize the configured resources	[SWS_CorTst_00179]
[SRS_SPAL_12163]	All driver modules shall implement an interface for de-initialization	[SWS_CorTst_00045]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 General Behavior

[SWS_CorTst_00008]

Upstream requirements: SRS_CoreTst_14104

⌈Core Test shall provide a procedure to test all CPU registers.⌋

[SWS_CorTst_00009]

Upstream requirements: SRS_CoreTst_14105

⌈The Core Test shall provide an Interrupt Controller and Exception detection test. Especially the detection of an interrupt itself and a branch to a valid interrupt service address shall be part of the test. It is regardless if the test is triggered by software exceptions or by a dedicated hardware unit built in silicon.⌋

[SWS_CorTst_00010]

Upstream requirements: SRS_CoreTst_14106

⌈The Core Test shall provide an Arithmetic and Logical Unit (ALU) test.⌋

[SWS_CorTst_00011]

Upstream requirements: SRS_CoreTst_14107

⌈The Core Test shall provide an address generation test.⌋

[SWS_CorTst_00012]

Upstream requirements: SRS_CoreTst_14108

⌈The Core Test shall provide a core memory interface test. This explicitly excludes tests on memory locations themselves which are connected external to a core itself or reside internal in a core.⌋

Note: Details of the required tests to be executed are provided in the corresponding HW documentation e.g. HW safety manual.

[SWS_CorTst_00013]

Upstream requirements: SRS_CoreTst_14109

⌈The Core Test shall provide a memory protection unit test (MPU). This is valid even if a Memory Management Unit (MMU) executes MPU functionality.⌋

[SWS_CorTst_00014]

Upstream requirements: SRS_CoreTst_14110

⌈The Core Test shall provide a Cache Controller Test. Especially the coherency and consistency between data or instructions located in memory outside the core and its appropriate cache entry representation shall be tested.⌋

[SWS_CorTst_00137]

Upstream requirements: SRS_CoreTst_14133

⌈Each Core Test Interval shall have an Identifier, which shall be incremented by each start of a new test interval in background mode.⌋

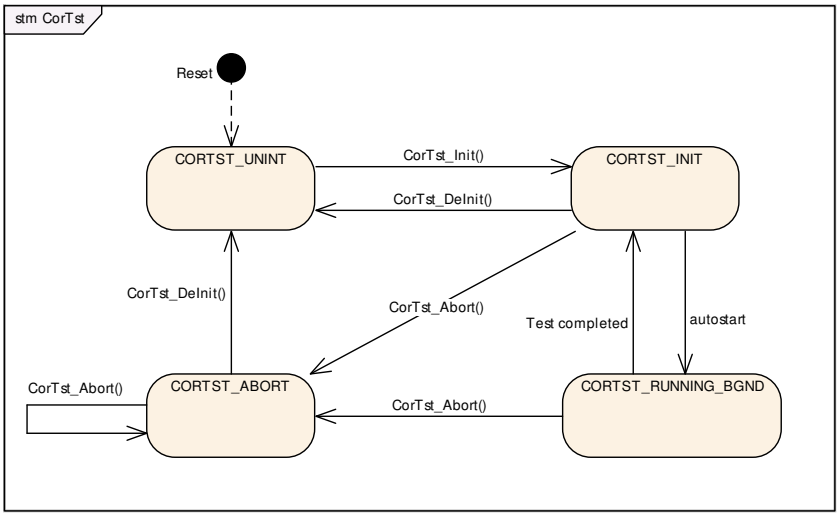
[SWS_CorTst_00144]

Upstream requirements: SRS_CoreTst_14112, SRS_CoreTst_14114

⌈Core Test module shall provide test execution services in background and foreground mode.⌋

Core Test states in background mode are described in [SWS_CorTst_00153]. The described states are driver states in background operation mode only.

[SWS_CorTst_00153] Core Test states in background mode ⌈



⌋

[SWS_CorTst_00145] ⌈Core Test is structured in partial tests (sets of hardware resource test) which can be interrupted by a higher priority task.⌋

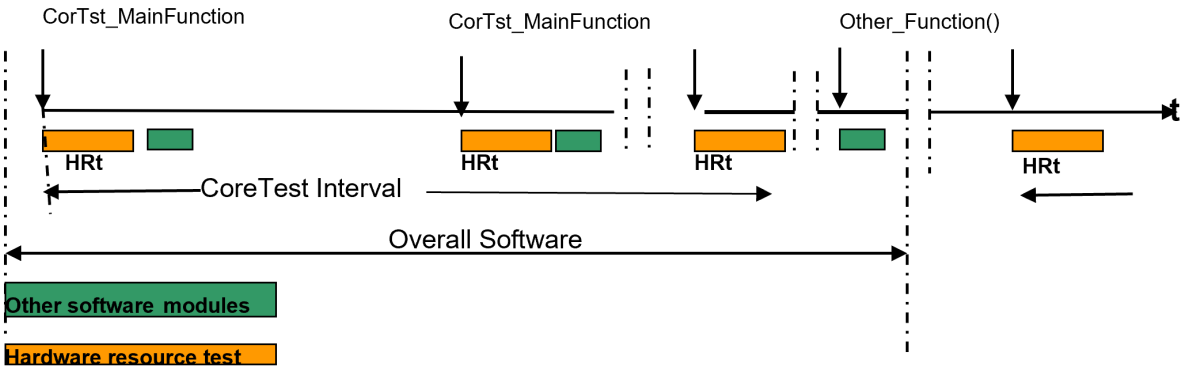


Figure 7.1: Background Test: Scheduling of Core Test (CorTst)

Each partial test is made up of atomic sequences which cannot be interrupted.

The following picture shows how `CorTst_MainFunction` is called by the scheduler, and how it can be interrupted between atomic pieces by higher priority tasks.

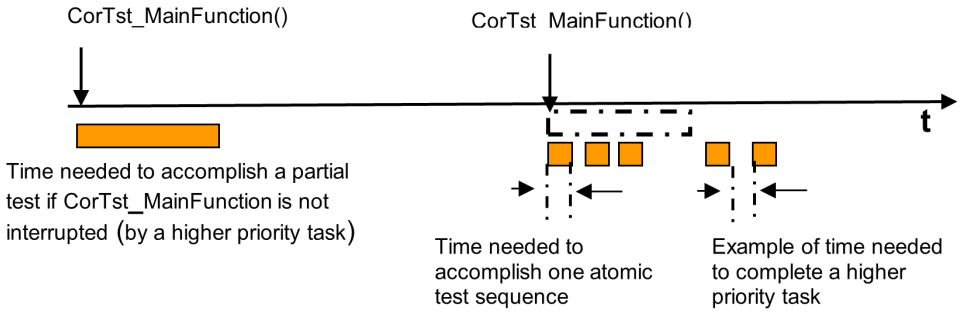


Figure 7.2: MainFunction called by scheduler

7.1.1 Background & Rationale

As described in [4], the Core Test is focused on testing the core, which includes the CPU and locally coupled units like e.g. MMU/MPU and Interrupt controller.

Due to complexity of a core implementation, a very deep knowledge of the core structure is a prerequisite to implement a Core Test. Therefore, it is assumed that a silicon manufacturer is the right entity to implement a Core Test by using an AUTOSAR API and provides the test as a library to user or application implementer.

Furthermore, it is assumed that a Core Test implementation may rarely be given away as a plain source code module from the silicon manufacturer to avoid IP draining.

7.2 Error Classification

Chapter [3, 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 following error types which may occur in BSW modules.

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

[SWS_CorTst_00021]

Upstream requirements: SRS_CoreTst_14117

⌈Except faults detected inside the CPU itself (e.g.ALU, MAC, etc.), which cannot be reliably reported by software. The errors that cannot be reliably reported by the `Dem_SetEventStatus` API shall be documented by the implementer.⌋

7.2.1 Development Errors

[SWS_CorTst_00016] Definition of development errors in module CorTst

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00327, SRS_CoreTst_14117

⌈

Type of error	*Related error code*	*Error value*
A particular API is called in an unexpected state	CORTST_E_STATUS_FAILURE	0x01
API service called with wrong parameter range	CORTST_E_PARAM_INVALID	0x11
API called without Core Test initialization	CORTST_E_UNINIT	0x20
API service CorTst_Init() called again without a CorTst_DeInit() inbetween	CORTST_E_ALREADY_INITIALIZED	0x23
API service called with a NULL pointer for CorTst_GetCurrentStatus()	CORTST_E_PARAM_POINTER	0x24

⌋

7.2.2 Runtime Errors

There are no runtime errors.

7.2.3 Production Errors

7.2.3.1 CORTST_E_CORE_FAILURE

[SWS_CorTst_01000] Core failure during tests

Upstream requirements: SRS_BSW_00339, SRS_BSW_00422, SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_00469

⌈

Diagnostic Event (Error Name)	CORTST_E_CORE_FAILURE
Description	This error indicates that the CorTst module detected a failure in a core.
Failed condition	PREFAILED is reported when CorTst_Start or CorTst_MainFunction detect a core failure. See SWS_CorTst_00154 and SWS_CorTst_00155.
Passed condition	PREPASSED is reported when CorTst_Start or CorTst_MainFunction could complete a core test without detecting an error. See SWS_CorTst_01001 and SWS_CorTst_01002.

⌋

7.3 General Requirements

[SWS_CorTst_00023] ⌈Due to the fact that Core Test is a MCAL driver module with no knowledge about the hardware/software system architecture, the tested entities and resources (e.g. ALU) shall be exclusively available prior start of test execution during runtime.⌋

[SWS_CorTst_00024] ⌈The Core Test implementer shall give an indication on the fault coverage achievements of a Core Test implementation.⌋

[SWS_CorTst_00026]

Upstream requirements: SRS_CoreTst_14130

⌈The Core Test shall be nondestructive to the tested entity. If Core Test modifies an entity setup, values, settings or selections on its own, it has to restore previous entity status before returning to calling service.⌋

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.

[SWS_CorTst_00027] Definition of imported datatypes of module CorTst

Upstream requirements: SRS_BSW_00304

⌈

Module	*Header File*	*Imported Type*
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type definitions

8.2.1 CorTst_ConfigType

[SWS_CorTst_01003] Definition of datatype CorTst_ConfigType

Upstream requirements: SRS_BSW_00414

⌈

Name	CorTst_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	–
Description	Configuration data structure of the CorTst module.	
Available via	CorTst.h	

⌋

8.2.2 CorTst_CsumSignatureType

[SWS_CorTst_00037] Definition of datatype CorTst_CsumSignatureType

Upstream requirements: SRS_BSW_00331

⌈

Name	CorTst_CsumSignatureType		
Kind	Type		
Derived from	***Basetype***	***Variation***	
	uint16	–	
	uint32	–	
Range	16..32 bit	–	Size depends on target platform.
Description	This is the type of the Core Test return value if a checksum/signature is returned from API to the caller of the API.		
Available via	CorTst.h		

⌋

8.2.3 CorTst_CsumSignatureBgndType

[SWS_CorTst_00176] Definition of datatype CorTst_CsumSignatureBgndType

Upstream requirements: SRS_CoreTst_14116

⌈

Name	CorTst_CsumSignatureBgndType	
Kind	Structure	
Elements	implementation specifc	
	Type	uint8, uint16, uint 32
	Comment	Implementation specific type
	0..<CorTstTestIntervalId EndValue>	
	Type	uint8, uint16, uint32
	Comment	value of CorTstTestIntervalId, which is incremented by each start of a test interval.
Description	Type for test signature in background mode	
Available via	CorTst.h	

⌋

8.2.4 CorTst_ErrOkType

[SWS_CorTst_00038] Definition of datatype CorTst_ErrOkType

Upstream requirements: SRS_BSW_00331

⌈

Name	CorTst_ErrOkType	
Kind	Structure	
Elements	0..<CorTstTestIntervalId EndValue>	
	Type	uint8, uint16, uint32
	Comment	value of CorTstTestIntervalId, which is incremented by each start of a test interval.
	returnvalue	
	Type	CorTst_ResultType
	Comment	CORTST_E_NOT_OK The Core Test detected at least one single test errors. CORTST_E_OKAY The Core test passed without errors. CORTST_E_NOT_TESTED There is no Core Test result available (default)
Description	This is the type of the Core Test test return if a status is retuned from API to the caller of the API.	
Available via	CorTst.h	

⌋

[SWS_CorTst_00138] ⌈For the type `CorTst_ErrOkType`, the enumeration value `CORTST_E_NOT_TESTED` shall be the default value after a reset. CorTstTestIntervalId shall have value zero per default.⌋

8.2.5 CorTst_ResultType

[SWS_CorTst_01005] Definition of datatype CorTst_ResultType

Upstream requirements: SRS_CoreTst_14131

⌈

Name	CorTst_ResultType		
Kind	Enumeration		
Range	CORTST_E_NOT_OK	0x00	The Core Test detected at least one single test errors.
	CORTST_E_OKAY	0x01	The Core test passed without errors.
	CORTST_E_NOT_TESTED	0x02	There is no Core Test result available (default)
Description	This is the type of the Core Test test return if a status is retuned from API to the caller of the API.		
Available via	CorTst.h		

⌋

8.2.6 CorTst_StateType

[SWS_CorTst_00039] Definition of datatype CorTst_StateType

Upstream requirements: SRS_BSW_00331

⌈

Name	CorTst_StateType		
Kind	Enumeration		
Range	CORTST_ABORT	0x00	The Core Test has been cancelled by API CorTst_Abort().
	CORTST_INIT	0x01	The Core Test is initialized and can be started.
	CORTST_UNINIT	0x02	The Core Test can be initialized.
	CORTST_RUNNING_BGND	0x03	The Core Test is currently executed
Description	This is a status value returned by the API CorTst_GetState().		
Available via	CorTst.h		

⌋

8.2.7 CorTst_TestIdFgndType

[SWS_CorTst_00160] Definition of datatype CorTst_TestIdFgndType

Upstream requirements: SRS_CoreTst_14113

⌈

Name	CorTst_TestIdFgndType		
Kind	Type		
Derived from	***Basetype***	***Variation***	
	uint16	–	
	uint32	–	
	uint8	–	
Range	8..32 bit	–	Size depends on target platform.
Description	This is the type of the parameter (Id) used for a specific foreground test configuration to run. (The Id shall be used in the call to the API CorTst_Start(CorTst_TestIdFgndType TestId)).		
Available via	CorTst.h		

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

This is a list of functions provided for calling services and upper layer modules.

Since Core Test module is a MCAL driver module, it does not provide any call-back functions for lower layered modules.

8.3.1 CorTst_Init

[SWS_CorTst_00040] Definition of API function CorTst_Init

Upstream requirements: SRS_BSW_00101, SRS_BSW_00406, SRS_BSW_00358, SRS_BSW_00414

⌈

Service Name	CorTst_Init	
Syntax	`void CorTst_Init ( const CorTst_ConfigType* ConfigPtr )`	
Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to the selected configuration set.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service for initialization and change of state of the Core Test	
Available via	CorTst.h	

⌋

[SWS_CorTst_01004]

Upstream requirements: SRS_BSW_00414

⌈The configuration pointer `ConfigPtr` shall always have `NULL_PTR` value.⌋

Note: The configuration pointer `ConfigPtr` is currently not used and shall therefore be set `NULL_PTR` value.

[SWS_CorTst_00041]

Upstream requirements: SRS_BSW_00101, SRS_SPAL_12057

⌈The function `CorTst_Init` shall initialize all CorTst relevant data structures, global variables, registers and special test hardware (if existing) with appropriate values used for core test.⌋

[SWS_CorTst_00179]

Upstream requirements: SRS_SPAL_12057, SRS_SPAL_12125

⌈The function `CorTst_Init` shall only initialize the configured resources and shall not touch resources that are not configured in the configuration file.⌋

[SWS_CorTst_00042] ⌈Execution state will be changed to `CORTST_INIT` if the driver is called while in state `CORTST_UNINIT`.⌋

[SWS_CorTst_00178] ⌈If `CorTst_Init` is called again while not in state `CORTST_UNINIT` a development error `CORTST_E_ALREADY_INITIALIZED` is reported. Execution state remains unchanged, the API call `CorTst_Init` is ignored.⌋

[SWS_CorTst_00044]

Upstream requirements: SRS_BSW_00406

⌈The function `CorTst_Init` shall be called first before calling any other CoreTest functions except the function `CorTst_GetState`.

If this sequence is not respected, the error code `CORTST_E_UNINIT` shall be reported to the Default Error Tracer (if development error detection is enabled).⌋

8.3.2 CorTst_DeInit

[SWS_CorTst_00045] Definition of API function CorTst_DeInit

Upstream requirements: SRS_BSW_00336, SRS_SPAL_12163

⌈

Service Name	CorTst_DeInit
Syntax	`void CorTst_DeInit ( void )`
Service ID [hex]	0x01
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Service to change from CORTST_ABORT or CORTST_INIT to CORTST_UNINIT state
Available via	CorTst.h

⌋

[SWS_CorTst_00046]

Upstream requirements: SRS_BSW_00336

⌈The function API `CorTst_DeInit` shall initialize all data structures, global variables, registers and special test hardware (if existing) with the default values after running the startup software (variable/structures) or power-on (HW-default).⌋

[SWS_CorTst_00047] ⌈If in state `CORTST_INIT`: The state shall be changed from `CORTST_INIT` to `CORTST_UNINIT` state.⌋

[SWS_CorTst_00136] ⌈If in state `CORTST_ABORT`: The state shall be changed from `CORTST_ABORT` to `CORTST_UNINIT` state.⌋

[SWS_CorTst_00149] ⌈If the DET is enabled and the status of the CORE Test module is `CORTST_RUNNING_BGND`, the function `CorTst_DeInit` shall report the error value `CORTST_E_STATUS_FAILURE` to the DET, and then immediately return.⌋

8.3.3 CorTst_Abort

[SWS_CorTst_00048] Definition of API function CorTst_Abort

Upstream requirements: SRS_CoreTst_14126

⌈

Service Name	CorTst_Abort
Syntax	`void CorTst_Abort ( void )`
Service ID [hex]	0x02
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Service to change from CORTST_INIT to CORTST_ABORT state
Available via	CorTst.h

⌋

[SWS_CorTst_00049] ⌈If the current state is `CORTST_INIT` the state shall be changed from `CORTST_INIT` to `CORTST_ABORT` state.⌋

[SWS_CorTst_00105] ⌈If the current state is `CORTST_RUNNING_BGND` the state shall be changed from `CORTST_RUNNING_BGND` to `CORTST_ABORT` state.⌋

[SWS_CorTst_00050]

Upstream requirements: SRS_CoreTst_14126

⌈When the `CorTst_Abort` function is called, `CorTst_MainFunction` shall finish the current atomic sequence it is executing plus shall provide already finished atomic test sequence results, before changing from `CORTST_RUNNING_BGND` to `CORTST_ABORT` state.⌋

[SWS_CorTst_00051] ⌈After a call to `CorTst_Abort`, `CorTst_MainFunction` shall not begin testing again when called by the scheduler before a complete re-initialization of the Core test module takes place by calling `CorTst_DeInit` and `CorTst_Init` again.⌋

[SWS_CorTst_00052] ⌈A call to `CorTst_Abort` while already being in state `CORTST_ABORT` does not change the state.⌋

[SWS_CorTst_00152] ⌈A call to `CorTst_Abort` shall set the result of function `CorTst_GetCurrentStatus` to return `CORTST_E_NOT_TESTED`.⌋

8.3.4 CorTst_GetState

[SWS_CorTst_00053] Definition of API function CorTst_GetState

Upstream requirements: SRS_CoreTst_14118

⌈

Service Name	CorTst_GetState	
Syntax	`CorTst_StateType CorTst_GetState ( void )`	
Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	CorTst_StateType	See type definition
Description	Service for Core Test to immediately return status on currently executed Core Test.	
Available via	CorTst.h	

⌋

[SWS_CorTst_00054]

Upstream requirements: SRS_CoreTst_14118

⌈The function `CorTst_GetState` shall return the current Core Test execution state regardless which state is currently executed. It is allowed to call this function in any execution state.⌋

8.3.5 CorTst_GetCurrentStatus

[SWS_CorTst_00055] Definition of API function CorTst_GetCurrentStatus

Upstream requirements: SRS_CoreTst_14131

⌈

Service Name	CorTst_GetCurrentStatus	
Syntax	`void CorTst_GetCurrentStatus ( CorTst_ErrOkType* ErrOk )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	ErrOk	See type definition
Return value	None	

▽

△

Description	Service for Core Test to get indicator of the last executed Core Test result
Available via	CorTst.h

⌋

[SWS_CorTst_00056]

Upstream requirements: SRS_CoreTst_14131

⌈The function `CorTst_GetCurrentStatus` shall return the result of the last completed Core Test run plus it shall return the Test Interval Id of the last background test.⌋

[SWS_CorTst_00120] ⌈The function `CorTst_GetCurrentStatus` shall return `CORTST_E_NOT_TESTED` per default if no result is available.⌋

8.3.6 CorTst_GetSignature

[SWS_CorTst_00057] Definition of API function CorTst_GetSignature

Upstream requirements: SRS_CoreTst_14115, SRS_CoreTst_14116

⌈

Service Name	CorTst_GetSignature	
Syntax	`CorTst_CsumSignatureBgndType CorTst_GetSignature ( void )`	
Service ID [hex]	0x05	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	CorTst_CsumSignatureBgndType	Implementation specific
Description	Service to get signature of the last executed Core Test in background mode.	
Available via	CorTst.h	

⌋

[SWS_CorTst_00058]

Upstream requirements: SRS_CoreTst_14116

⌈The function `CorTst_GetSignature` shall return currently pending Core Test result signature and Core Test Interval Id of the last completed test run in background mode.⌋

[SWS_CorTst_00121] ⌈The function `CorTst_GetSignature` shall return value zero per default as signature until a first initial Core Test run has successfully been executed which will provide a first valid signature representation.⌋

8.3.7 CorTst_GetFgndSignature

[SWS_CorTst_00060] Definition of API function CorTst_GetFgndSignature

Upstream requirements: SRS_CoreTst_14115, SRS_CoreTst_14116

⌈

Service Name	CorTst_GetFgndSignature	
Syntax	`CorTst_CsumSignatureType CorTst_GetFgndSignature ( void )`	
Service ID [hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	CorTst_CsumSignature Type	Implementation specific
Description	Service to get signature of the last executed Core Test in foreground mode.	
Available via	CorTst.h	

⌋

[SWS_CorTst_00061]

Upstream requirements: SRS_CoreTst_14116

⌈The function `CorTst_GetFgndSignature` shall return Core Test result signature type as Core Test result of the last completed test run in foreground mode.⌋

[SWS_CorTst_00122] ⌈The function `CorTst_GetFgndSignature` shall return value zero per default as signature until a first initial Core Test run has successfully been executed which will provide first valid signature representation.⌋

8.3.8 CorTst_Start

[SWS_CorTst_00064] Definition of API function CorTst_Start

Upstream requirements: SRS_BSW_00357, SRS_CoreTst_14112, SRS_CoreTst_14113

⌈

Service Name	CorTst_Start	
Syntax	`Std_ReturnType CorTst_Start ( CorTst_TestIdFgndType TestId )`	
Service ID [hex]	0x07	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	TestId	Id of the foreground test configuration to be executed.

▽

△

Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Foreground test processed E_NOT_OK: Foreground test not accepted
Description	Service for executing foreground Core Test.	
Available via	CorTst.h	

⌋

[SWS_CorTst_00065] ⌈The function CorTst_Start is only applicable for Foreground mode Core Test operation.⌋

[SWS_CorTst_00109] ⌈If the execution state is CORTST_RUNNING_BGND while this function API is called, the function shall return without any action and the return value shall be E_OK.⌋

[SWS_CorTst_00154]

Upstream requirements: SRS_BSW_00339, SRS_BSW_00422, SRS_BSW_00409, SRS_BSW_00469

⌈In case an error occurs during test, the CorTst_Start function shall report the production error CORTST_E_CORE_FAILURE (see [ECUC_CorTst_00157]) as DEM_EVENT_STATUS_PREFAILED to the DEM if the core can still report errors reliably by software.⌋

[SWS_CorTst_01001]

Upstream requirements: SRS_BSW_00339, SRS_BSW_00422, SRS_BSW_00409, SRS_BSW_00469, SRS_BSW_00471

⌈In case no errors occured during test, the CorTst_Start function shall report the production error CORTST_E_CORE_FAILURE (see [ECUC_CorTst_00157]) as DEM_EVENT_STATUS_PREPASSED to the DEM.⌋

[SWS_CorTst_00161]

Upstream requirements: SRS_BSW_00323

⌈If development error detection is enabled and the parameter TestId is out of the range, the DET error value CORTST_E_PARAM_INVALID shall be raised and the function shall return without any action with return value E_NOT_OK.⌋

8.4 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

[SWS_CorTst_00067] Definition of scheduled function CorTst_MainFunction

Upstream requirements: SRS_BSW_00433, SRS_CoreTst_14112, SRS_CoreTst_14114

⌈

Service Name	CorTst_MainFunction
Syntax	`void CorTst_MainFunction ( void )`
Service ID [hex]	0x0b
Description	Cyclically called by scheduler to perform processing of Core Test.
Available via	SchM_CorTst.h

⌋

[SWS_CorTst_00068] ⌈The function `CorTst_MainFunction` shall set state to `CORTST_INIT`, if all work within a Core Test interval has been finished.⌋

[SWS_CorTst_00069] ⌈The function `CorTst_MainFunction` shall set state to `CORTST_INIT`, if no work within a Core Test needs to be done.⌋

[SWS_CorTst_00070] ⌈If the CoreTest module is in the state `CORTST_INIT`, a call to the API `CorTst_MainFunction` shall change the state of the module to `CORTST_RUNNING_BGND`.⌋

[SWS_CorTst_00071] ⌈`CorTst_MainFunction` shall test all selected core hardware entities as configured in [ECUC_CorTst_00087].⌋

[SWS_CorTst_00072] ⌈The function `CorTst_MainFunction` shall set Core Test result status to `CORTST_E_OKAY` or `CORTST_E_NOT_OK` after each complete test cycle - which may consist itself of many different atomic test cycles - depending on the result of Core Test.⌋

[SWS_CorTst_00073] ⌈`CORTST_E_OKAY` shall be set as status from `CorTst_MainFunction` processing only in the case that every selected atomic part of `CorTst_MainFunction` has been successfully executed without any kind of errors. In all other cases `CORTST_E_NOT_OK` is returned as current status. Status can be checked by calling `CorTst_GetCurrentStatus`.⌋

[SWS_CorTst_00074] ⌈`CorTst_MainFunction` shall set `CORTST_E_NOT_OK` status after first detected error in a sequence of atomic parts of Core Test module. Status can be checked by calling `CorTst_GetCurrentStatus`.⌋

[SWS_CorTst_00139]

Upstream requirements: SRS_CoreTst_14133

⌈The function `CorTst_MainFunction` shall increment Test Interval Id before start of a new test interval. The first test interval shall always have the Test Interval Id = "0" (=zero). If Test Interval Id becomes greater than or equal to `CorTstTestIntervalIdEndValue` Test Interval Id shall start again with value "0" (=zero) for the next test

interval. The value shall be provided as part of the return values of `CorTst_GetSignature` and `CorTst_GetCurrentStatus` in background mode.⌋

[SWS_CorTst_00155]

Upstream requirements: SRS_BSW_00339, SRS_BSW_00422, SRS_BSW_00409, SRS_BSW_00469

⌈In case an error occurs during test, the `CorTst_MainFunction` function shall report the production error `CORTST_E_CORE_FAILURE` (see [ECUC_CorTst_00157]) as `DEM_EVENT_STATUS_PREFAILED` to the DEM if the core can still report errors reliably by software.⌋

[SWS_CorTst_01002]

Upstream requirements: SRS_BSW_00339, SRS_BSW_00422, SRS_BSW_00409, SRS_BSW_00469, SRS_BSW_00471

⌈In case a core test is completed during a `CorTst_MainFunction` invocation and no errors occured during this test, the `CorTst_MainFunction` function shall report the production error `CORTST_E_CORE_FAILURE` (see [ECUC_CorTst_00157]) as `DEM_EVENT_STATUS_PREPASSED` to the DEM.⌋

8.5 Expected interfaces

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

8.5.1 Mandatory interfaces

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

[SWS_CorTst_00177] Definition of mandatory interfaces required by module Cor Tst

Upstream requirements: SRS_BSW_00339

⌈

API Function	*Header File*	*Description*
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEvent ReportingType} == STANDARD_REPORTING)

⌋

8.5.2 Optional interfaces

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

[SWS_CorTst_00183] Definition of optional interfaces requested by module Cor Tst

Upstream requirements: SRS_BSW_00369, SRS_BSW_00350

⌈

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

⌋

8.5.3 Configurable interfaces

In this section, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of this kind of interfaces are not fixed because they are configurable.

[SWS_CorTst_00076] Definition of API function CorTst_TestCompletedNotification

Upstream requirements: SRS_BSW_00359, SRS_BSW_00360, SRS_CoreTst_14119

⌈

Service Name	CorTst_TestCompletedNotification	
Syntax	`void CorTst_TestCompletedNotification ( CorTst_ErrOkType ResultOfLastCorTstRun )`	
Service ID [hex]	0x0c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ResultOfLastCorTstRun	CORTST_E_OKAY Last Core Test execution successfully finished with no errors CORTST_E_NOT_OK Last Core Test execution finished with errors.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The function CorTst_TestCompletedNotification shall be called every time when a complete test cycle has been executed.	
Available via	CorTst.h	

⌋

[SWS_CorTst_00077]

Upstream requirements: SRS_CoreTst_14119, SRS_SPAL_00157

⌈The Core Test module shall call the callback notification `CorTst_TestCompleted-Notification` every time when it has executed a complete Core Test cycle based on a combination of atomic parts of Core Test in background mode.⌋

[SWS_CorTst_00140] ⌈The call of function `CorTst_TestCompleted-Notification` shall be pre compile time configurable by the configuration parameter `CorTstNotificationSupported`.⌋

9 Sequence diagrams

9.1 Initialization

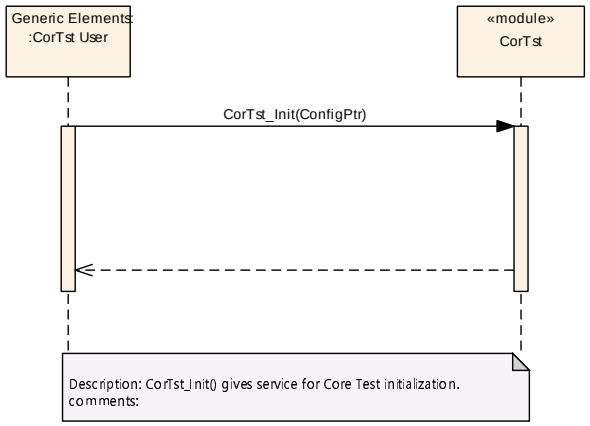


Figure 9.1: Core Test Init

9.2 Deinitialization

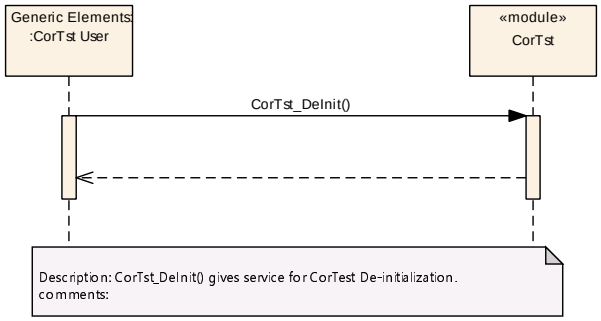


Figure 9.2: Core Test De-initialization

9.3 Background Test

9.3.1 Test Result Calculation within Core Test Module

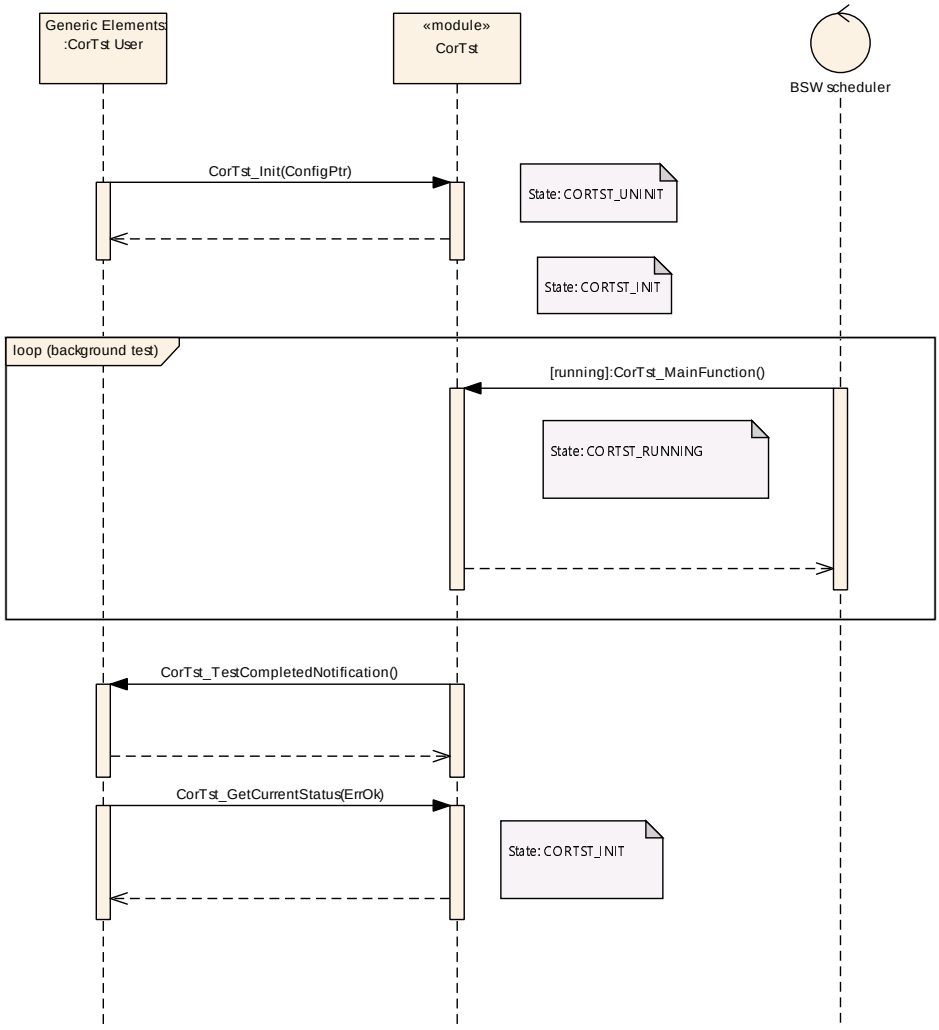


Figure 9.3: Result Calculation within Core Test Driver

9.3.2 Core Test Signature provided to Calling Entity

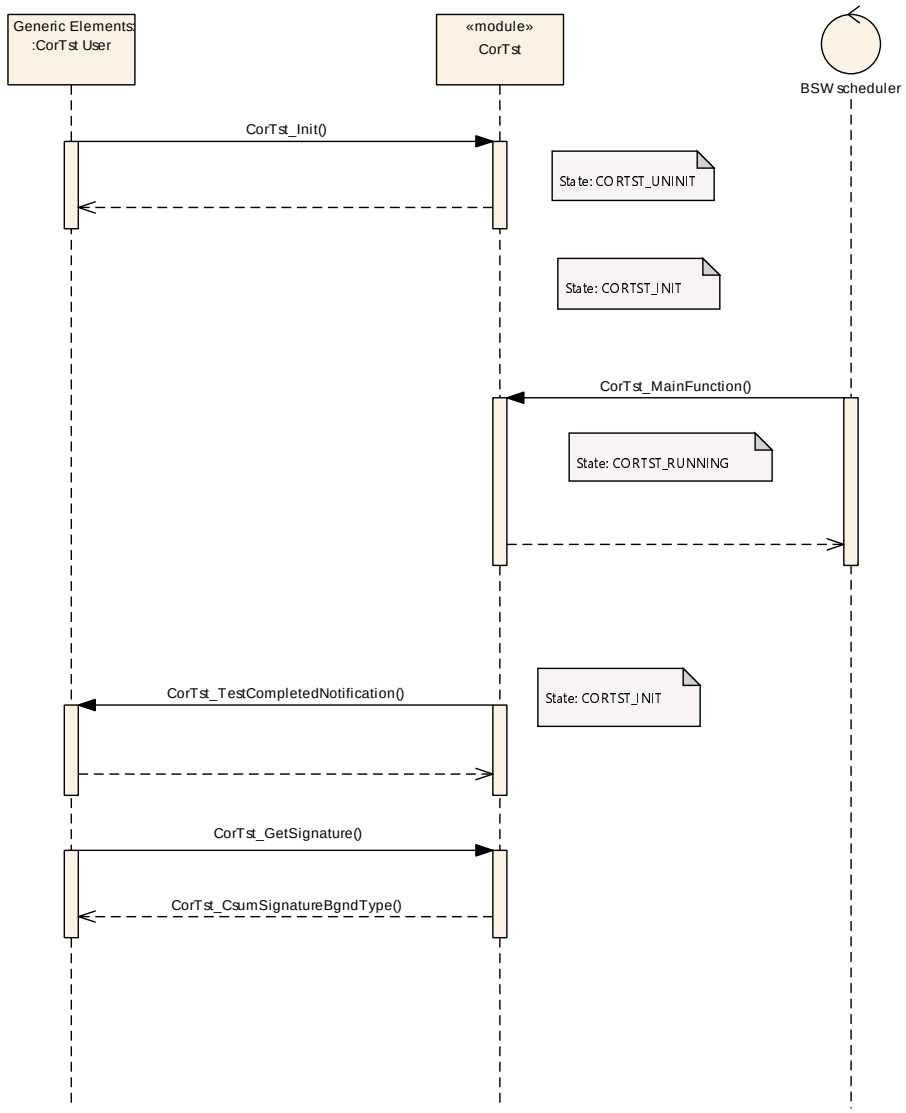


Figure 9.4: Result Calculation on Calling Entity

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module CoreTest.

Chapter 10.3 specifies published information of the module CoreTest.

10.1 How to read this chapter

For details refer to [3] Chapter 10.1 *"Introduction to configuration specification"*. in [3].

[SWS_CorTst_01006] ⌈The Core Test module shall reject configurations with partition mappings which are not supported by the implementation.⌋

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 7 and Chapter 8.

10.2.1 CorTst

[ECUC_CorTst_00125] Definition of EcucModuleDef CorTst ⌈

Module Name	CorTst
Description	Configuration of the CorTst module.
Post-Build Selectable Support	false
Post-Build Loadable Support	false

Included Containers		
Container Name	**Multiplicity**	**Description**
CorTstBackgroundConfigSet	0..*	Multiple Configuration Set Container, defines background mode.
CorTstConfigApiServices	1	Configure which CorTst service APIs are available
CorTstDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
CorTstForegroundConfigSet	1..*	Multiple Configuration Set Container , defines foreground mode.
CorTstGeneral	1	Contains the general configuration parameters of the module.

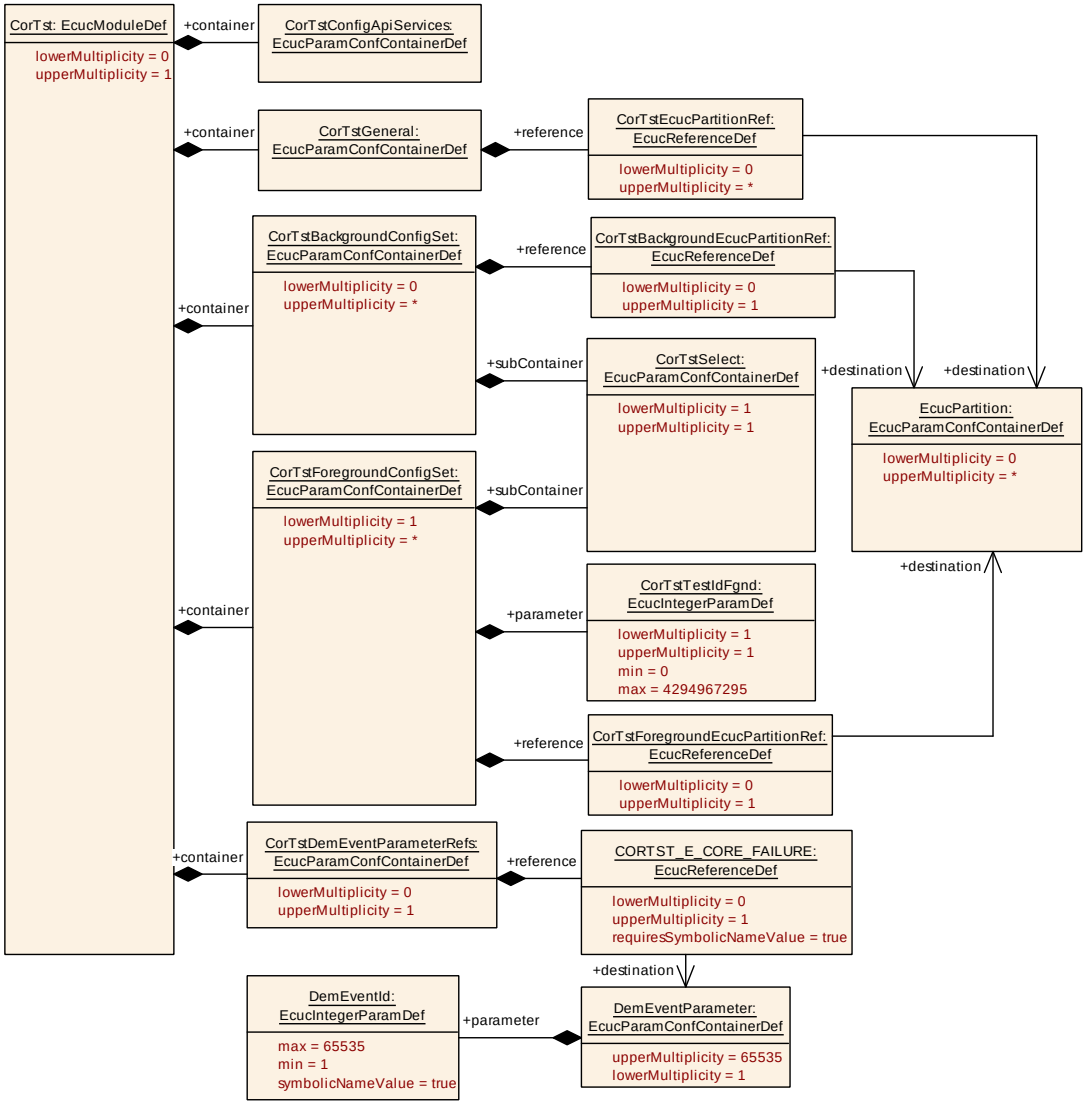


Figure 10.1: CorTst

10.2.2 CorTstGeneral

[ECUC_CorTst_00081] Definition of EcucParamConfContainerDef CorTstGeneral

Container Name	CorTstGeneral
Parent Container	CorTst
Upstream mapping defined	–
Description	Contains the general configuration parameters of the module.
Multiplicity	1

△

Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CorTstDevErrorDetect	1	[ECUC_CorTst_00082]
CorTstFgndTestNumber	1	[ECUC_CorTst_00159]
CorTstNotificationSupported	1	[ECUC_CorTst_00083]
CorTstTestIntervalIdEndValue	0..1	[ECUC_CorTst_00143]
CorTstTestResultMode	1	[ECUC_CorTst_00086]
CorTstEcucPartitionRef	0..*	[ECUC_CorTst_00160]

No Included Containers

⌋

[ECUC_CorTst_00082] Definition of EcucBooleanParamDef CorTstDevErrorDetect ⌈

Parameter Name	CorTstDevErrorDetect
Parent Container	CorTstGeneral
Upstream mapping defined	–
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_CorTst_00159] Definition of EcucIntegerParamDef CorTstFgndTestNumber ⌈

Parameter Name	CorTstFgndTestNumber
Parent Container	CorTstGeneral
Upstream mapping defined	–
Description	This parameter holds the number of test configurations available for the foreground tests as defined in this configuration.
Multiplicity	1
Type	EcucIntegerParamDef
Range	1 .. 4294967295
Default value	–
Pre-Compile only	true

⌋

[ECUC_CorTst_00083] Definition of EcucBooleanParamDef CorTstNotification Supported ⌈

Parameter Name	CorTstNotificationSupported
Parent Container	CorTstGeneral
Upstream mapping defined	–
Description	Switch to indicate that the notification is supported.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	true
Pre-Compile only	true

⌋

[ECUC_CorTst_00143] Definition of EcucIntegerParamDef CorTstTestIntervalId EndValue ⌈

Parameter Name	CorTstTestIntervalIdEndValue
Parent Container	CorTstGeneral
Upstream mapping defined	–
Description	Defines the end value of the Test Interval Id.
Multiplicity	0..1
Type	EcucIntegerParamDef
Range	0 .. 4294967295
Default value	–
Pre-Compile only	true

⌋

[ECUC_CorTst_00086] Definition of EcucBooleanParamDef CorTstTestResult Mode ⌈

Parameter Name	CorTstTestResultMode
Parent Container	CorTstGeneral
Upstream mapping defined	–
Description	Switch for enabling test result comparison within the Core test driver. In this mode a core test result OK or NOTOK shall not be calculated from the core test driver. Within core test driver no comparison against the reference value is processed.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_CorTst_00160] Definition of EcucReferenceDef CorTstEcucPartitionRef ⌈

Parameter Name	CorTstEcucPartitionRef
Parent Container	CorTstGeneral
Upstream mapping defined	–

▽

△

Description	Maps the core test to zero or multiple ECUC partitions to make the modules API available in this partition.
Multiplicity	0..*
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

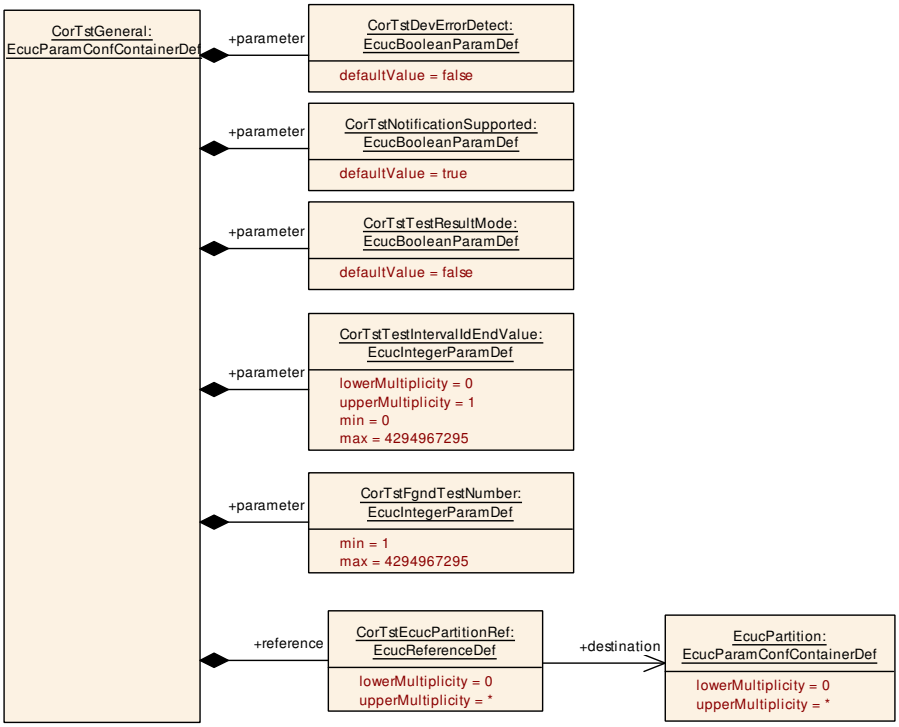


Figure 10.2: CorTstGeneral

10.2.3 CorTstSelect

[ECUC_CorTst_00089] Definition of EcucParamConfContainerDef CorTstSelect

⌈

Container Name	CorTstSelect
Parent Container	CorTstBackgroundConfigSet, CorTstForegroundConfigSet
Upstream mapping defined	–
Description	This container specifies configuration parameters to select individual tests for foreground mode and background mode. The availability is hardware and implementation specific.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CorTstAddress	1	[ECUC_CorTst_00130]
CorTstAlu	1	[ECUC_CorTst_00129]
CorTstCache	1	[ECUC_CorTst_00133]
CorTstInterrupt	1	[ECUC_CorTst_00128]
CorTstMemoryIf	1	[ECUC_CorTst_00131]
CorTstMpu	1	[ECUC_CorTst_00132]
CorTstRegister	1	[ECUC_CorTst_00127]

No Included Containers

⌋

[ECUC_CorTst_00130] Definition of EcucBooleanParamDef CorTstAddress ⌈

Parameter Name	CorTstAddress
Parent Container	CorTstSelect
Upstream mapping defined	–
Description	Enable/Disables core address test.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_CorTst_00129] Definition of EcucBooleanParamDef CorTstAlu ⌈

Parameter Name	CorTstAlu
Parent Container	CorTstSelect
Upstream mapping defined	–
Description	Enable/Disables core ALU test.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_CorTst_00133] Definition of EcucBooleanParamDef CorTstCache ⌈

Parameter Name	CorTstCache
Parent Container	CorTstSelect
Upstream mapping defined	–
Description	Enable/Disables core cache test.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false

▽

△

Pre-Compile only	true

⌋

[ECUC_CorTst_00128] Definition of EcucBooleanParamDef CorTstInterrupt ⌈

Parameter Name	CorTstInterrupt
Parent Container	CorTstSelect
Upstream mapping defined	–
Description	Enable/Disables core interrupt test
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_CorTst_00131] Definition of EcucBooleanParamDef CorTstMemoryIf ⌈

Parameter Name	CorTstMemoryIf
Parent Container	CorTstSelect
Upstream mapping defined	–
Description	Enable/Disables core memory interface test
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_CorTst_00132] Definition of EcucBooleanParamDef CorTstMpu ⌈

Parameter Name	CorTstMpu
Parent Container	CorTstSelect
Upstream mapping defined	–
Description	Enable/Disables core MPU test
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_CorTst_00127] Definition of EcucBooleanParamDef CorTstRegister ⌈

Parameter Name	CorTstRegister
Parent Container	CorTstSelect
Upstream mapping defined	–
Description	Enable/Disables core register test

▽

△

Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

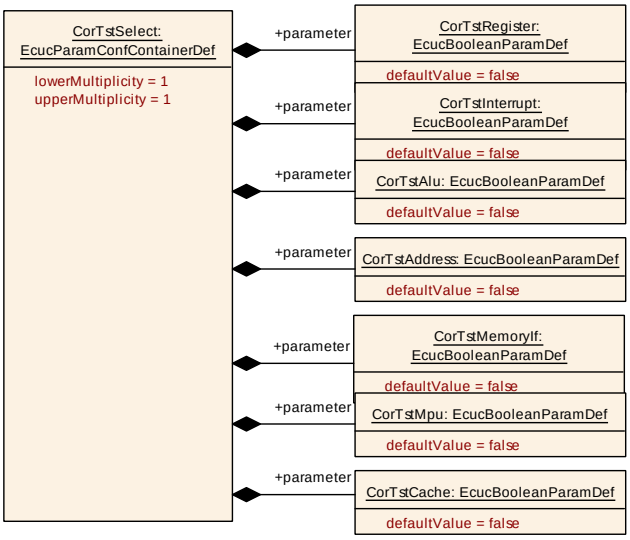


Figure 10.3: CorTstSelect

10.2.4 CorTstBackgroundConfigSet

[ECUC_CorTst_00087] Definition of EcucParamConfContainerDef CorTstBackgroundConfigSet ⌈

Container Name	CorTstBackgroundConfigSet
Parent Container	CorTst
Upstream mapping defined	–
Description	Multiple Configuration Set Container, defines background mode.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CorTstBackgroundEcucPartitionRef	0..1	[ECUC_CorTst_00161]

Included Containers		
Container Name	**Multiplicity**	**Description**
CorTstSelect	1	This container specifies configuration parameters to select individual tests for foreground mode and background mode. The availability is hardware and implementation specific.

⌋

[ECUC_CorTst_00161] Definition of EcucReferenceDef CorTstBackgroundEcucPartitionRef ⌈

Parameter Name	CorTstBackgroundEcucPartitionRef
Parent Container	CorTstBackgroundConfigSet
Upstream mapping defined	–
Description	Maps the background test configuration to zero or one ECUC partitions. The ECUC partition referenced is a subset of the ECUC partitions where the CorTst driver is mapped to.
Multiplicity	0..1
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

[SWS_CorTst_01008] ⌈The ECUC partitions referenced by `CorTstBackgroundEcucPartitionRef` shall be a subset of the ECUC partitions referenced by `CorTstEcucPartitionRef`.⌋

[SWS_CorTst_01010] ⌈If `CorTstEcucPartitionRef` references one or more ECUC partitions, `CorTstBackgroundEcucPartitionRef` shall have a multiplicity of one and reference one of these ECUC partitions as well.⌋

10.2.5 CorTstForegroundConfigSet

[ECUC_CorTst_00088] Definition of EcucParamConfContainerDef CorTstForegroundConfigSet ⌈

Container Name	CorTstForegroundConfigSet
Parent Container	CorTst
Upstream mapping defined	–
Description	Multiple Configuration Set Container , defines foreground mode.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CorTstTestIdFgnd	1	[ECUC_CorTst_00158]
CorTstForegroundEcucPartitionRef	0..1	[ECUC_CorTst_00162]

Included Containers		
Container Name	**Multiplicity**	**Description**
CorTstSelect	1	This container specifies configuration parameters to select individual tests for foreground mode and background mode. The availability is hardware and implementation specific.

⌋

[ECUC_CorTst_00158] Definition of EcucIntegerParamDef CorTstTestIdFgnd ⌈

Parameter Name	CorTstTestIdFgnd	
Parent Container	CorTstForegroundConfigSet	
Upstream mapping defined	–	
Description	This is the Id of this specific foreground test configuration. The value shall be used in the call to the API CorTst_Start(CorTst_TestIdFgndType TestId).	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_CorTst_00162] Definition of EcucReferenceDef CorTstForegroundEcucPartitionRef ⌈

Parameter Name	CorTstForegroundEcucPartitionRef
Parent Container	CorTstForegroundConfigSet
Upstream mapping defined	–
Description	Maps the foreground test configuration to zero or one ECUC partitions. The ECUC partition referenced is a subset of the ECUC partitions where the CorTst driver is mapped to.
Multiplicity	0..1
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

[SWS_CorTst_01011] ⌈The ECUC partitions referenced by `CorTstForeground-EcucPartitionRef` shall be a subset of the ECUC partitions referenced by `Cor-TstEcucPartitionRef`.⌋

[SWS_CorTst_01012] ⌈If `CorTstEcucPartitionRef` references one or more ECUC partitions, `CorTstForegroundEcucPartitionRef` shall have a multiplicity of one and reference one of these ECUC partitions as well.⌋

10.2.6 CorTstConfigApiServices

[ECUC_CorTst_00092] Definition of EcucParamConfContainerDef CorTstConfig ApiServices ⌈

Container Name	CorTstConfigApiServices
Parent Container	CorTst
Upstream mapping defined	–
Description	Configure which CorTst service APIs are available
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
CorTstAbortApi	1	[ECUC_CorTst_00094]
CorTstGetCurrentStatus	1	[ECUC_CorTst_00104]
CorTstGetFgndSignature	1	[ECUC_CorTst_00103]
CorTstGetSignature	1	[ECUC_CorTst_00097]
CorTstGetStateApi	1	[ECUC_CorTst_00096]
CorTstStartApi	1	[ECUC_CorTst_00093]

No Included Containers

⌋

[ECUC_CorTst_00094] Definition of EcucBooleanParamDef CorTstAbortApi ⌈

Parameter Name	CorTstAbortApi
Parent Container	CorTstConfigApiServices
Upstream mapping defined	–
Description	Adds / removes the service CorTst_Abort() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_CorTst_00104] Definition of EcucBooleanParamDef CorTstGetCurrentStatus ⌈

Parameter Name	CorTstGetCurrentStatus
Parent Container	CorTstConfigApiServices
Upstream mapping defined	–
Description	Adds / removes the service CorTst_GetCurrentStatus() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_CorTst_00103] Definition of EcucBooleanParamDef CorTstGetFgndSignature ⌈

Parameter Name	CorTstGetFgndSignature
Parent Container	CorTstConfigApiServices
Upstream mapping defined	–
Description	Adds / removes the service CorTst_GetFgndSignature() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_CorTst_00097] Definition of EcucBooleanParamDef CorTstGetSignature ⌈

Parameter Name	CorTstGetSignature
Parent Container	CorTstConfigApiServices
Upstream mapping defined	–
Description	Adds / removes the service CorTst_GetSignature() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_CorTst_00096] Definition of EcucBooleanParamDef CorTstGetStateApi ⌈

Parameter Name	CorTstGetStateApi
Parent Container	CorTstConfigApiServices
Upstream mapping defined	–
Description	Adds / removes the service CorTst_GetState() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_CorTst_00093] Definition of EcucBooleanParamDef CorTstStartApi ⌈

Parameter Name	CorTstStartApi
Parent Container	CorTstConfigApiServices
Upstream mapping defined	–
Description	Adds / removes the service CorTst_Start() from the code.
Multiplicity	1
Type	EcucBooleanParamDef

▽

△

Default value	false
Pre-Compile only	true

⌋

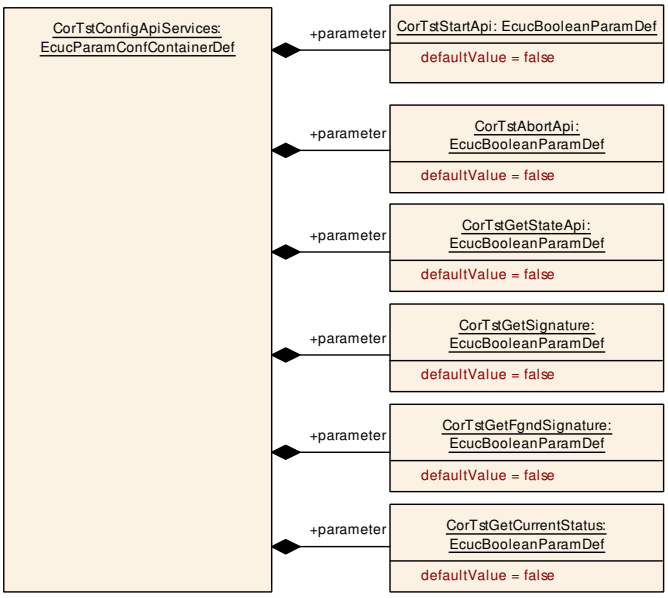


Figure 10.4: CorTstConfigApiServices

10.2.7 CorTstDemEventParameterRefs

[ECUC_CorTst_00156] Definition of EcucParamConfContainerDef CorTstDemEventParameterRefs ⌈

Container Name	CorTstDemEventParameterRefs
Parent Container	CorTst
Upstream mapping defined	–
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The Event Id is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CORTST_E_CORE_FAILURE	0..1	[ECUC_CorTst_00157]

No Included Containers

⌋

[ECUC_CorTst_00157] Definition of EcucReferenceDef CORTST_E_CORE_FAILURE ⌈

Parameter Name	CORTST_E_CORE_FAILURE
Parent Container	CorTstDemEventParameterRefs
Upstream mapping defined	–
Description	Reference to the DemEventParameter which shall be issued when the error "CORE failure" has occured. Dependency: Dem
Multiplicity	0..1
Type	Symbolic name reference to DemEventParameter
Pre-Compile only	true

⌋

10.3 Published Information

For details refer to [3] Chapter 10.3 *“Published Information”*.

A Not applicable requirements

[SWS_CorTst_NA_00999]

Upstream requirements: SRS_BSW_00167, SRS_BSW_00168, SRS_BSW_00344, SRS_BSW_00375, SRS_BSW_00383, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00437, SRS_BSW_00438, SRS_BSW_00170, SRS_BSW_00171, SRS_CoreTst_14125, SRS_CoreTst_14124, SRS_CoreTst_14121, SRS_CoreTst_14127, SRS_CoreTst_14128, SRS_BSW_00490, SRS_BSW_00491

⌈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 hyperlinks in the document.

B.1 Traceable item history of this document according to AUTOSAR Release R25-11

B.1.1 Added Specification Items in R25-11

none

B.1.2 Changed Specification Items in R25-11

none

B.1.3 Deleted Specification Items in R25-11

none

B.2 Traceable item history of this document according to AUTOSAR Release R26-11

B.2.1 Added Specification Items in R26-11

none

B.2.2 Changed Specification Items in R26-11

Number	Heading
[ECUC_CorTst_00081]	Definition of EcucParamConfContainerDef CorTstGeneral
[ECUC_CorTst_00082]	Definition of EcucBooleanParamDef CorTstDevErrorDetect
[ECUC_CorTst_00083]	Definition of EcucBooleanParamDef CorTstNotificationSupported

▽

△

Number	Heading
[ECUC_CorTst_00086]	Definition of EcucBooleanParamDef CorTstTestResultMode
[ECUC_CorTst_00087]	Definition of EcucParamConfContainerDef CorTstBackgroundConfigSet
[ECUC_CorTst_00088]	Definition of EcucParamConfContainerDef CorTstForegroundConfigSet
[ECUC_CorTst_00089]	Definition of EcucParamConfContainerDef CorTstSelect
[ECUC_CorTst_00092]	Definition of EcucParamConfContainerDef CorTstConfigApiServices
[ECUC_CorTst_00093]	Definition of EcucBooleanParamDef CorTstStartApi
[ECUC_CorTst_00094]	Definition of EcucBooleanParamDef CorTstAbortApi
[ECUC_CorTst_00096]	Definition of EcucBooleanParamDef CorTstGetStateApi
[ECUC_CorTst_00097]	Definition of EcucBooleanParamDef CorTstGetSignature
[ECUC_CorTst_00103]	Definition of EcucBooleanParamDef CorTstGetFgndSignature
[ECUC_CorTst_00104]	Definition of EcucBooleanParamDef CorTstGetCurrentStatus
[ECUC_CorTst_00125]	Definition of EcucModuleDef CorTst
[ECUC_CorTst_00127]	Definition of EcucBooleanParamDef CorTstRegister
[ECUC_CorTst_00128]	Definition of EcucBooleanParamDef CorTstInterrupt
[ECUC_CorTst_00129]	Definition of EcucBooleanParamDef CorTstAlu
[ECUC_CorTst_00130]	Definition of EcucBooleanParamDef CorTstAddress
[ECUC_CorTst_00131]	Definition of EcucBooleanParamDef CorTstMemoryIf
[ECUC_CorTst_00132]	Definition of EcucBooleanParamDef CorTstMpu
[ECUC_CorTst_00133]	Definition of EcucBooleanParamDef CorTstCache
[ECUC_CorTst_00143]	Definition of EcucIntegerParamDef CorTstTestIntervalIdEndValue
[ECUC_CorTst_00156]	Definition of EcucParamConfContainerDef CorTstDemEventParameterRefs

▽

△

Number	Heading
[ECUC_CorTst_00157]	Definition of EcucReferenceDef CORTST_E_CORE_FAILURE
[ECUC_CorTst_00158]	Definition of EcucIntegerParamDef CorTstTestIdFgnd
[ECUC_CorTst_00159]	Definition of EcucIntegerParamDef CorTstFgndTestNumber
[ECUC_CorTst_00160]	Definition of EcucReferenceDef CorTstEcucPartitionRef
[ECUC_CorTst_00161]	Definition of EcucReferenceDef CorTstBackgroundEcucPartitionRef
[ECUC_CorTst_00162]	Definition of EcucReferenceDef CorTstForegroundEcucPartitionRef
[SWS_CorTst_00016]	Definition of development errors in module CorTst
[SWS_CorTst_00027]	Definition of imported datatypes of module CorTst
[SWS_CorTst_00044]	
[SWS_CorTst_00153]	Core Test states in background mode
[SWS_CorTst_00154]	
[SWS_CorTst_00155]	
[SWS_CorTst_01000]	Core failure during tests
[SWS_CorTst_01001]	
[SWS_CorTst_01002]	

Table B.1: Changed Specification Items in R26-11

B.2.3 Deleted Specification Items in R26-11

Number	Heading
[ECUC_CorTst_00098]	Definition of EcucBooleanParamDef CorTstVersionInfoApi
[SWS_CorTst_00112]	Definition of API function CorTst_GetVersionInfo
[SWS_CorTst_00118]	

Table B.2: Deleted Specification Items in R26-11