

<b>Document Title</b>	Specification of Core Test
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	259
<b>Document Status</b>	Final
<b>Part of AUTOSAR Standard</b>	Classic Platform
<b>Part of Standard Release</b>	4.3.1

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2017-12-08	4.3.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Minor corrections</li> </ul>
2016-11-30	4.3.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Replaced Development Error Tracer with Default Error Tracer</li> <li>Removed Debugging Support section</li> <li>Removed Variants section</li> </ul>
2015-07-31	4.2.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Correction of CorTst_Init prototype</li> <li>Added CorTst_ConfigType and CorTst_ResultType</li> <li>Debugging support marked as obsolete</li> <li>Minor corrections</li> </ul>
2014-10-31	4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>CORTST_E_CORE_FAILURE extended production error formalization, including healing.</li> <li>Correction of CorTst_GetCurrentStatus prototype</li> </ul>
2013-10-31	4.1.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Removed timing attribute of requirement SWS_CorTst_00067</li> <li>Editorial changes</li> <li>Removed chapter(s) on change documentation</li> </ul>
2013-03-15	4.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Alignment to the new SWS_BSWGeneral document</li> <li>Updated document for Extended Production Errors</li> <li>Alignment to official naming in other Autosar documents</li> </ul>

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2017-12-08	4.3.1	AUTOSAR Release Management	<ul style="list-style-type: none"><li>• Minor corrections</li></ul>
2011-12-22	4.0.3	AUTOSAR Administration	<ul style="list-style-type: none"><li>• Clarification of some requirements.</li><li>• Typos correction.</li><li>• Removed redundant and useless requirements.</li></ul>
2010-09-30	3.1.5	AUTOSAR Administration	<ul style="list-style-type: none"><li>• Added new requirements for configuration and error detection.</li><li>• Clarification of some requirements.</li><li>• Added new configuration parameters.</li><li>• Removed obsolete requirements.</li><li>• Improvement of static error detection.</li><li>• Removed unused types.</li></ul>
2010-02-02	3.1.4	AUTOSAR Administration	<ul style="list-style-type: none"><li>• Initial release</li></ul>

## Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## Table of Contents

1	Introduction and functional overview .....	6
2	Acronyms and Abbreviations .....	7
3	Related documentation.....	8
3.1	Input documents.....	8
3.2	Related standards and norms .....	8
3.3	Related specification .....	8
4	Constraints and assumptions .....	9
4.1	Limitations .....	9
4.2	Applicability to car domains.....	9
4.3	Applicability to safety related environments .....	9
5	Dependencies to other modules.....	10
5.1	File structure .....	10
5.1.1	Code file structure .....	10
5.1.2	Header file structure .....	10
6	Requirements traceability .....	12
7	Functional specification .....	19
7.1	General Behavior .....	19
7.1.1	Background & Rationale .....	21
7.2	Error classification .....	21
7.2.1	Development Errors .....	21
7.2.2	Runtime Errors.....	21
7.2.3	Transient Faults .....	22
7.2.4	Production Errors .....	22
7.2.5	Extended Production Errors .....	22
7.3	Error notification .....	22
7.4	General Requirements .....	23
8	API specification.....	24
8.1	Imported types.....	24
8.2	Type definitions .....	24
8.2.1	CorTst_ConfigType .....	24
8.2.2	CorTst_CsumSignatureType.....	24
8.2.3	CorTst_CsumSignatureBgndType .....	24
8.2.4	CorTst_ErrOkType .....	25
8.2.5	CorTst_ResultType .....	25
8.2.6	CorTst_StateType .....	25
8.2.7	CorTst_TestIdFgndType .....	26
8.3	Function definitions .....	27
8.3.1	CorTst_Init.....	27
8.3.2	CorTst_DeInit .....	28
8.3.3	CorTst_Abort.....	29
8.3.4	CorTstGetState .....	29

8.3.5	CorTst_GetCurrentStatus.....	30
8.3.6	CorTstGetSignature .....	30
8.3.7	CorTst_GetFgndSignature .....	31
8.3.8	CorTst_Start.....	32
8.3.9	CorTst_GetVersionInfo .....	34
8.4	Call-back notifications .....	34
8.5	Scheduled functions .....	34
8.5.1	CorTst_MainFunction.....	34
8.6	Expected Interfaces.....	36
8.6.1	Mandatory Interfaces .....	36
8.6.2	Optional Interfaces .....	36
8.6.3	Configurable interfaces .....	36
9	Sequence diagrams .....	38
9.1	Initialization .....	38
9.2	Deinitialization .....	39
9.3	Background Test .....	40
9.3.1	Test Result Calculation within Core Test Module.....	40
9.3.2	Core Test Signature provided to Calling Entity.....	41
10	Configuration specification.....	42
10.1	How to read this chapter .....	42
10.2	Containers and configuration parameters .....	43
10.2.1	CorTst .....	43
10.2.2	CorTstGeneral.....	44
10.2.3	CorTstSelect .....	46
10.2.4	CorTstBackgroundConfigSet.....	49
10.2.5	CorTstForegroundConfigSet .....	49
10.2.6	CorTstConfigApiServices .....	50
10.2.7	CorTstDemEventParameterRefs.....	52
10.3	Published Information.....	53
11	Not applicable requirements .....	54

## 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

<b>Abbreviation / Acronym:</b>	<b>Description:</b>
MCAL	Microcomputer Abstraction Layer
DEM	Diagnostic Event Manager
DET	Default Error Tracer
CPU	Central Processing Unit
MPU	Memory Protection Unit
L1	1 <sup>st</sup> level memory
L2	2 <sup>nd</sup> 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.

<b>Term:</b>	<b>Description:</b>
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. It is the user of the API call.
test interval	<i>CoreTest test Interval</i> : the sum of all the <i>partial tests</i> (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.

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

## 3 Related documentation

### 3.1 Input documents

- [1] List of Basic Software Modules  
AUTOSAR\_TR\_BSWModuleList.pdf
- [2] Layered Software Architecture  
AUTOSAR\_EXP\_LayeredSoftwareArchitecture.pdf
- [3] General Requirements on Basic Software Modules  
AUTOSAR\_SRS\_BSWGeneral.pdf
- [4] Specification of BSW Scheduler  
AUTOSAR\_SWS\_BSW\_Scheduler.pdf
- [5] ECU Configuration Specification  
AUTOSAR\_SWS\_ECUCStateManager.pdf
- [6] Specification of Memory Mapping  
AUTOSAR\_SWS\_MemoryMapping.pdf
- [7] Requirement on Core Test  
AUTOSAR\_SRS\_CoreTest.pdf
- [8] AUTOSAR Basic Software Module Description Template  
AUTOSAR\_RS\_BSWModuleDescriptionTemplate.pdf
- [9] General Specification of Basic Software Modules  
AUTOSAR\_SWS\_BSWGeneral.pdf

### 3.2 Related standards and norms

- [10] ISO DIS 26262, [www.iso.org](http://www.iso.org)

### 3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [9] (SWS BSW General), which is also valid for Core Test.

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



## 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]

[ The Core Test module shall provide interrupt service routines for test purposes only. ] (SRS\_BSW\_00164, SRS\_CoreTst\_14105)

-

#### 5.1.2 Header file structure

The Core Test inclusion structure for the source code shall be as follows:

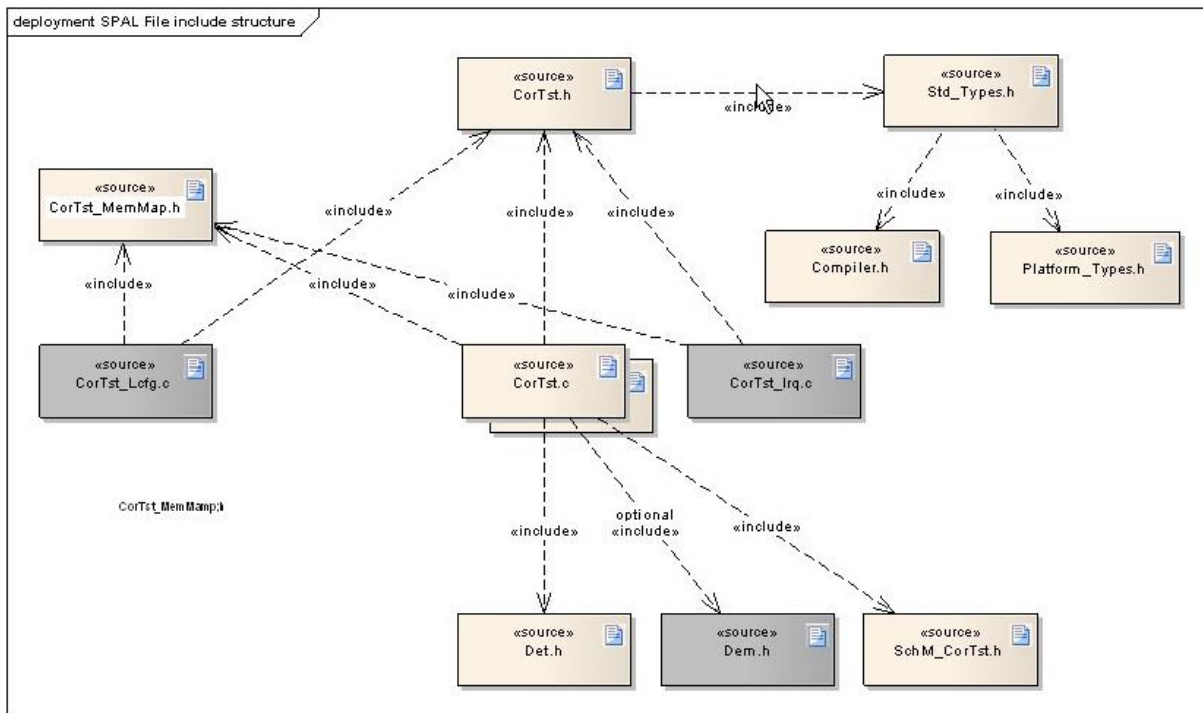


Figure 1 – File structure

## 6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00003	All software modules shall provide version and identification information	SWS_CorTst_00112
SRS_BSW_00004	All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files	SWS_CorTst_00112
SRS_BSW_00005	Modules of the $\mu$ C Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_CorTst_00999
SRS_BSW_00006	The source code of software modules above the $\mu$ C Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_CorTst_00999
SRS_BSW_00009	All Basic SW Modules shall be documented according to a common standard.	SWS_CorTst_00999
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_CorTst_00999
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_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_CorTst_00999
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_CorTst_00999
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_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_CorTst_00999
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_CorTst_00999
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_CorTst_00999
SRS_BSW_00171	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	SWS_CorTst_00999
SRS_BSW_00172	The scheduling strategy that is built inside the Basic Software Modules	SWS_CorTst_00999

	shall be compatible with the strategy used in the system	
SRS_BSW_00301	All AUTOSAR Basic Software Modules shall only import the necessary information	SWS_CorTst_00999
SRS_BSW_00302	All AUTOSAR Basic Software Modules shall only export information needed by other modules	SWS_CorTst_00999
SRS_BSW_00304	All AUTOSAR Basic Software Modules shall use the following data types instead of native C data types	SWS_CorTst_00027
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_CorTst_00999
SRS_BSW_00308	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	SWS_CorTst_00999
SRS_BSW_00309	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	SWS_CorTst_00999
SRS_BSW_00310	API naming convention	SWS_CorTst_00999
SRS_BSW_00312	Shared code shall be reentrant	SWS_CorTst_00999
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_CorTst_00999
SRS_BSW_00318	Each AUTOSAR Basic Software Module file shall provide version numbers in the header file	SWS_CorTst_00999
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	SWS_CorTst_00999
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_CorTst_00161
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_CorTst_00999
SRS_BSW_00327	Error values naming convention	SWS_CorTst_00016
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_CorTst_00999
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_CorTst_00999
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_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_CorTst_00999

SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the meta data	SWS_CorTst_00999
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_00338	-	SWS_CorTst_00183
SRS_BSW_00339	Reporting of production relevant error status	SWS_CorTst_00154, SWS_CorTst_00155, SWS_CorTst_00177, SWS_CorTst_00999, SWS_CorTst_01000, SWS_CorTst_01001, SWS_CorTst_01002
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_CorTst_00999
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_CorTst_00999
SRS_BSW_00346	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	SWS_CorTst_00999
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	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	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_00371	The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules	SWS_CorTst_00999
SRS_BSW_00374	All Basic Software Modules shall provide a readable module vendor identification	SWS_CorTst_00999
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_CorTst_00999
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_CorTst_00999
SRS_BSW_00379	All software modules shall provide a module identifier in the header file and	SWS_CorTst_00999

	in the module XML description file.	
SRS_BSW_00383	The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description	SWS_CorTst_00999
SRS_BSW_00385	List possible error notifications	SWS_CorTst_00016, SWS_CorTst_01000
SRS_BSW_00386	The BSW shall specify the configuration for detecting an error	SWS_CorTst_00999, SWS_CorTst_01000
SRS_BSW_00398	The link-time configuration is achieved on object code basis in the stage after compiling and before linking	SWS_CorTst_00999
SRS_BSW_00399	Parameter-sets shall be located in a separate segment and shall be loaded after the code	SWS_CorTst_00999
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_CorTst_00999
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_CorTst_00999
SRS_BSW_00406	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	SWS_CorTst_00040, SWS_CorTst_00044
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_CorTst_00112, SWS_CorTst_00118
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_00999, SWS_CorTst_01001, SWS_CorTst_01002
SRS_BSW_00411	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	SWS_CorTst_00112
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_CorTst_00999
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_00416	The sequence of modules to be initialized shall be configurable	SWS_CorTst_00999
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_CorTst_00999
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_CorTst_00154, SWS_CorTst_00155, SWS_CorTst_00999, SWS_CorTst_01000,

		SWS_CorTst_01001, SWS_CorTst_01002
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_CorTst_00999
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_CorTst_00999
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_CorTst_00999
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_CorTst_00999
SRS_BSW_00428	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	SWS_CorTst_00999
SRS_BSW_00429	Access to OS is restricted	SWS_CorTst_00999
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_CorTst_00999
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_00437	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	SWS_CorTst_00999
SRS_BSW_00438	Configuration data shall be defined in a structure	SWS_CorTst_00999
SRS_BSW_00466	Classification of extended production errors	SWS_CorTst_00154, SWS_CorTst_00155, SWS_CorTst_01000, SWS_CorTst_01001, SWS_CorTst_01002
SRS_BSW_00469	Fault detection and healing of production errors and extended production errors	SWS_CorTst_00154, SWS_CorTst_00155, SWS_CorTst_01000, 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 [approved]	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 [approved]	SWS_CorTst_00064, SWS_CorTst_00160
SRS_CoreTst_14114	A Main Function for the Core Test Shall Be Available [approved]	SWS_CorTst_00067, SWS_CorTst_00144
SRS_CoreTst_14115	Test Metrics Shall Be Available to Caller [approved]	SWS_CorTst_00057, SWS_CorTst_00060
SRS_CoreTst_14116	A Service shall be provided which returns a checksum/signature as test result [approved]	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 [approved]	SWS_CorTst_00053, SWS_CorTst_00054
SRS_CoreTst_14119	A Notification of Completion Shall Be Provided [approved]	SWS_CorTst_00076, SWS_CorTst_00077
SRS_CoreTst_14124	The implementation of the Core test shall have to comply with the IEC61508 [rejected]	SWS_CorTst_00999
SRS_CoreTst_14125	Diagnostic Coverage [rejected]	SWS_CorTst_00999
SRS_CoreTst_14126	It Shall Be Possible to Cancel a Running Test [approved]	SWS_CorTst_00048, SWS_CorTst_00050
SRS_CoreTst_14130	Destructive Test Shall Restore Original State of tested Entity [approved]	SWS_CorTst_00026
SRS_CoreTst_14131	A Service shall be provided which returns a Pass/Fail Status Representation as a test result [approved]	SWS_CorTst_00055, SWS_CorTst_00056, SWS_CorTst_01005
SRS_CoreTst_14133	Each Core Test interval shall have an identifier [approved]	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



## 7 Functional specification

### 7.1 General Behavior

[SWS\_CorTst\_00008]

[ Core Test shall provide a procedure to test all CPU registers.  
] (SRS\_CoreTst\_14104)

[SWS\_CorTst\_00009] [

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. ] (SRS\_CoreTst\_14105)

[SWS\_CorTst\_00010]

[ The Core Test shall provide an Arithmetic and Logical Unit (ALU) test.  
] (SRS\_CoreTst\_14106)

[SWS\_CorTst\_00011]

[ The Core Test shall provide an address generation test. ] (SRS\_CoreTst\_14107)

[SWS\_CorTst\_00012]

[ 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. ] (SRS\_CoreTst\_14108)

[SWS\_CorTst\_00013]

[ The Core Test shall provide a memory protection unit test (MPU). This is valid even if a Memory Management Unit (MMU) executes MPU functionality.  
] (SRS\_CoreTst\_14109)

[SWS\_CorTst\_00014]

[ 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. ] (SRS\_CoreTst\_14110)

[SWS\_CorTst\_00137]

[ Each Core Test Interval shall have an Identifier, which shall be incremented by each start of a new test interval in background mode. ] (SRS\_CoreTst\_14133)

[SWS\_CorTst\_00144]

[ Core Test module shall provide test execution services in background and foreground mode. ] (SRS\_CoreTst\_14112, SRS\_CoreTst\_14114)

Core Test states in background mode are described in Figure 2. The described states are driver states in background operation mode only.

[SWS\_CorTst\_00153] [

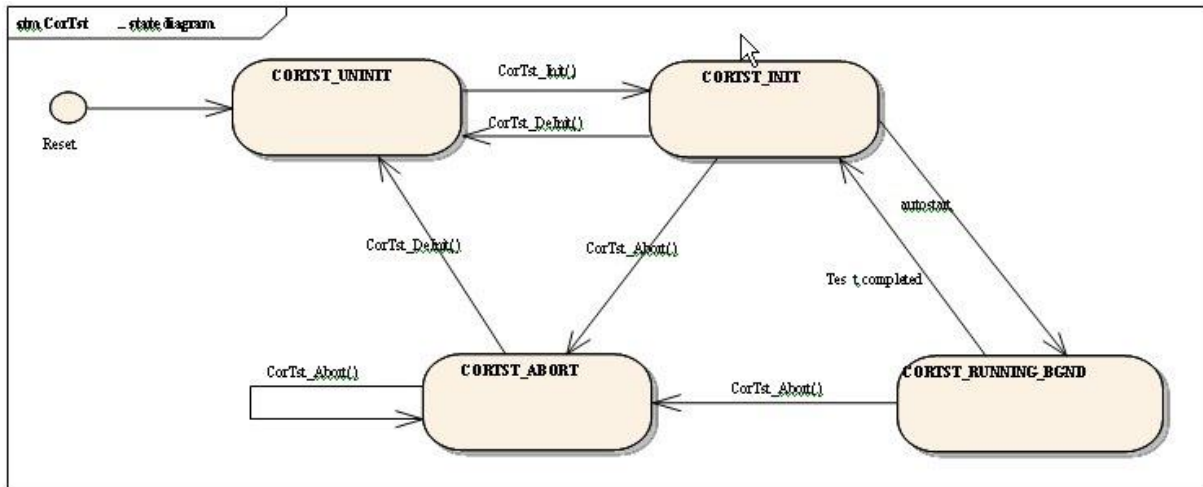


Figure 2 - State Diagram ] ()

[SWS\_CorTst\_00145] [

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

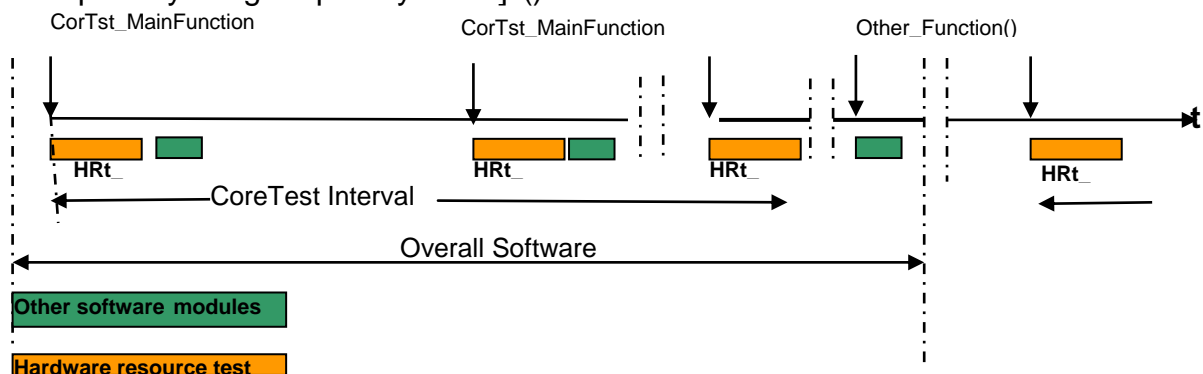
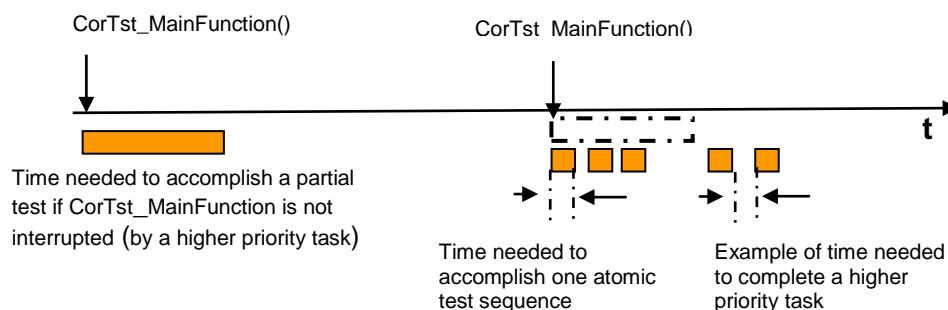


Figure 3 – Background Test: Scheduling of Core Test (CorTst)

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.



### 7.1.1 Background & Rationale

As described in the Core Test SRS, 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

### 7.2.1 Development Errors

[SWS\_CorTst\_00016]

└ The Core Test shall detect the following API parameter errors depending on its build options:

<b>ID:</b>	<b>Type of error</b>	<b>Relevance</b>	<b>Related error code</b>	<b>Value [hex]</b>
SWS_CorTst_00169	API service called with wrong parameter range	Development	CORTST_E_PARAM_INV ALID	0x11
SWS_CorTst_00170	API called without Core Test initialization	Development	CORTST_E_UNINIT	0x20
SWS_CorTst_00172	API service CorTst_Init() called again without a CorTst_Delnit() in-between	Development	CORTST_E_ALREADY_INITIALIZED	0x23
SWS_CorTst_00180	API service called with a NULL pointer for CorTst_GetVersionInfo() and CorTst_GetCurrentStatus()	Development	CORTST_E_PARAM_POINTER	0x24
SWS_CorTst_00181	A particular API is called in an unexpected state	Development	CORTST_E_STATUS_FAILURE	0x01

└ (SRS\_BSW\_00337, SRS\_BSW\_00385, SRS\_BSW\_00327, SRS\_CoreTst\_14117)

### 7.2.2 Runtime Errors

There are no runtime errors.

### 7.2.3 Transient Faults

There are no transient faults.

### 7.2.4 Production Errors

This module does not specify any production errors.

### 7.2.5 Extended Production Errors

#### 7.2.5.1 CORTST\_E\_CORE\_FAILURE

[SWS\_CorTst\_01000]

<b>Error Name:</b>	CORTST_E_CORE_FAILURE	
<b>Short Description:</b>	Core failure during tests	
<b>Long Description:</b>	This error indicates that the CorTst module detected a failure in a core.	
<b>Detection Criteria:</b>	Fail	PREFAILED is reported when CorTst_Start or CorTst_MainFunction detect a core failure.  See <a href="#">SWS_CorTst_00154</a> , <a href="#">SWS_CorTst_00155</a> .
	Pass	PREPASSED is reported when CorTst_Start or CorTst_MainFunction could complete a core test without detecting an error.  See <a href="#">SWS_CorTst_01001</a> , and <a href="#">SWS_CorTst_01002</a> .
<b>Secondary Parameters:</b>	The PREPASSED and PREFAILED detection is always active. However PREFAILED status may not be reported if the core is not in a state where errors can reliably be reported by software.  See <a href="#">SWS_CorTst_00154</a> and <a href="#">SWS_CorTst_00155</a> .	
<b>Time Required:</b>	The time required to detect a failure depends on the frequency of the CorTst_Start or CorTst_MainFunction invocation and the number of foreground or background tests (see <a href="#">ECUC_CorTst_00125</a> )  The time required to recover from a failure may be longer as transient hardware failures from a core should be considered as failures.	
<b>Monitor Frequency</b>	Continuous See <a href="#">SWS_CorTst_00154</a> , <a href="#">SWS_CorTst_00155</a> , <a href="#">SWS_CorTst_01001</a> , and <a href="#">SWS_CorTst_01002</a> .	

] ( SRS\_BSW\_00339, SRS\_BSW\_00422, SRS\_BSW\_00385, SRS\_BSW\_00386, SRS\_BSW\_00466, SRS\_BSW\_00469)

## 7.3 Error notification

[SWS\_CorTst\_00021]

[ 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.

] (SRS\_CoreTst\_14117)

## 7.4 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]

[ 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. ] (SRS\_CoreTst\_14130)

## 8 API specification

### 8.1 Imported types

This chapter lists all types included from other BSW modules.

#### [SWS\_CorTst\_00027] [

<i>Module</i>	<i>Imported Type</i>
Dem	Dem_EventIdType
	Dem_EventStatusType
Std_Types	Std_ReturnType
	Std_VersionInfoType

] (SRS\_BSW\_00304)

### 8.2 Type definitions

#### 8.2.1 CorTst\_ConfigType

##### [SWS\_CorTst\_01003] [

<b>Name:</b>	CorTst_ConfigType	
<b>Type:</b>	Structure	
<b>Range:</b>	implementation specific	--
<b>Description:</b>	Configuration data structure of the CorTst module.	

] (SRS\_BSW\_00414)

#### 8.2.2 CorTst\_CsumSignatureType

##### [SWS\_CorTst\_00037] [

<b>Name:</b>	CorTst_CsumSignatureType	
<b>Type:</b>	uint16, uint32	
<b>Range:</b>	16..32 bit	-- Size depends on target platform.
<b>Description:</b>	This is the type of the Core Test return value if a checksum/signature is returned from API to the caller of the API.	

] (SRS\_BSW\_00331)

#### 8.2.3 CorTst\_CsumSignatureBgndType

##### [SWS\_CorTst\_00176] [

<b>Name:</b>	CorTst_CsumSignatureBgndType		
<b>Type:</b>	Structure		
<b>Element:</b>	uint8, uint16, uint32	implementation specific	Implementation specific type
	uint8, uint16, uint32	0.<CorTstTestIntervalId EndValue>	value of CorTstTestIntervalId, which is incremented by each start of a test interval.
<b>Description:</b>	Type for test signature in background mode		



] (SRS\_CoreTst\_14116)

## 8.2.4 CorTst\_ErrOkType

[SWS\_CorTst\_00038] [

<b>Name:</b>	CorTst_ErrOkType		
<b>Type:</b>	Structure		
<b>Element:</b>	uint8, uint16, uint32	0..<CorTstTestIntervalId EndValue>	value of CorTstTestIntervalId, which is incremented by each start of a test interval.
	CorTst_ResultType	returnvalue	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)
<b>Description:</b>	This is the type of the Core Test test return if a status is returned from API to the caller of the API.		

] (SRS\_BSW\_00331)

[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] [

<b>Name:</b>	CorTst_ResultType		
<b>Type:</b>	Enumeration		
<b>Range:</b>	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)
<b>Description:</b>	This is the type of the Core Test test return if a status is returned from API to the caller of the API.		

] (SRS\_CoreTst\_14131)

## 8.2.6 CorTst\_StateType

[SWS\_CorTst\_00039] [

<b>Name:</b>	CorTst_StateType		
<b>Type:</b>	Enumeration		
<b>Range:</b>	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
<b>Description:</b>	This is a status value returned by the API CorTst_GetState().		

] (SRS\_BSW\_00331)

### 8.2.7 CorTst\_TestIdFgndType

[SWS\_CorTst\_00160] [

<b>Name:</b>	CorTst_TestIdFgndType		
<b>Type:</b>	uint8, uint16, uint32		
<b>Range:</b>	8..32 bit	--	Size depends on target platform.
<b>Description:</b>	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)).		

] (SRS\_CoreTst\_14113)

## 8.3 Function definitions

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

### 8.3.1 CorTst\_Init

#### [SWS\_CorTst\_00040] [

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

] (SRS\_BSW\_00101, SRS\_BSW\_00406, SRS\_BSW\_00358, SRS\_BSW\_00414)

#### [SWS\_CorTst\_01004]

[ The configuration pointer `ConfigPtr` shall always have `NULL_PTR` value. ] ( SRS\_BSW\_00414)

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

#### [SWS\_CorTst\_00041]

[ 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. ] (SRS\_BSW\_00101, SRS\_SPAL\_12057)

#### [SWS\_CorTst\_00179]

[ The function `CorTst_Init` shall only initialize the configured resources and shall not touch resources that are not configured in the configuration file. ] (SRS\_SPAL\_12057, SRS\_SPAL\_12125)

#### [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]

[ The function `CorTst_Init` shall be called first before calling any other CoreTest functions except the functions `CorTst_GetState` and `CorTst_GetVersionInfo`. 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). ] (SRS\_BSW\_00406)

### 8.3.2 CorTst\_DeInit

[SWS\_CorTst\_00045] [

<b>Service name:</b>	CorTst_DeInit
<b>Syntax:</b>	void CorTst_DeInit( void )
<b>Service ID[hex]:</b>	0x01
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	Service to change from <code>CORTST_ABORT</code> or <code>CORTST_INIT</code> to <code>CORTST_UNINIT</code> state

] (SRS\_BSW\_00336, SRS\_SPAL\_12163)

[SWS\_CorTst\_00046]

[ 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). ] (SRS\_BSW\_00336)

[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] [

<b>Service name:</b>	CorTst_Abort
<b>Syntax:</b>	void CorTst_Abort( void )
<b>Service ID[hex]:</b>	0x02
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	Service to change from CORTST_INIT to CORTST_ABORT state

] (SRS\_CoreTst\_14126)

#### [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]

[ 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. ] (SRS\_CoreTst\_14126)

#### [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 CorTstGetState

#### [SWS\_CorTst\_00053] [

<b>Service name:</b>	CorTst_GetState
<b>Syntax:</b>	CorTst_StateType CorTst_GetState (

	void
	)
<b>Service ID[hex]:</b>	0x03
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	CorTst_StateType   See type definition
<b>Description:</b>	Service for Core Test to immediately return status on currently executed Core Test.

] (SRS\_CoreTst\_14118)

[SWS\_CorTst\_00054]

[ 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. ] (SRS\_CoreTst\_14118)

### 8.3.5 CorTst\_GetCurrentStatus

[SWS\_CorTst\_00055] [

<b>Service name:</b>	CorTst_GetCurrentStatus
<b>Syntax:</b>	void CorTst_GetCurrentStatus ( CorTst_ErrOkType* ErrOk )
<b>Service ID[hex]:</b>	0x04
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	ErrOk   See type definition
<b>Return value:</b>	None
<b>Description:</b>	Service for Core Test to get indicator of the last executed Core Test result

] (SRS\_CoreTst\_14131)

[SWS\_CorTst\_00056]

[ 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. ] (SRS\_CoreTst\_14131)

[SWS\_CorTst\_00120]

[ The function `CorTst_GetCurrentStatus` shall return `CORTST_E_NOT_TESTED` per default if no result is available. ] ( )

### 8.3.6 CorTstGetSignature

[SWS\_CorTst\_00057] [

<b>Service name:</b>	CorTst_GetSignature
<b>Syntax:</b>	CorTst_CsumSignatureBgndType CorTst_GetSignature ( void

	)
<b>Service ID[hex]:</b>	0x05
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	CorTst_CsumSignatureBgndType   Implementation specific
<b>Description:</b>	Service to get signature of the last executed Core Test in background mode.

] (SRS\_CoreTst\_14115, SRS\_CoreTst\_14116)

[SWS\_CorTst\_00058]

[ 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. ] (SRS\_CoreTst\_14116)

[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] [

<b>Service name:</b>	CorTst_GetFgndSignature
<b>Syntax:</b>	CorTst_CsumSignatureType CorTst_GetFgndSignature ( void )
<b>Service ID[hex]:</b>	0x06
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	CorTst_CsumSignatureType   Implementation specific
<b>Description:</b>	Service to get signature of the last executed Core Test in foreground mode.

] (SRS\_CoreTst\_14115, SRS\_CoreTst\_14116)

[SWS\_CorTst\_00061]

[ The function `CorTst_GetFgndSignature` shall return Core Test result signature type as Core Test result of the last completed test run in foreground mode. ] (SRS\_CoreTst\_14116)

[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] [

<b>Service name:</b>	CorTst_Start	
<b>Syntax:</b>	Std_ReturnType CorTst_Start( CorTst_TestIdFgndType TestId )	
<b>Service ID[hex]:</b>	0x07	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	TestId	Id of the foreground test configuration to be executed.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: Foreground test processed E_NOT_OK: Foreground test not accepted
<b>Description:</b>	Service for executing foreground Core Test.	

] (SRS\_BSW\_00357, SRS\_CoreTst\_14112, SRS\_CoreTst\_14113)

#### [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]

[ In case an error occurs during test, the `CorTst_Start` function shall report the extended 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. ] (SRS\_BSW\_00339, SRS\_BSW\_00422, SRS\_BSW\_00409, SRS\_BSW\_00466, SRS\_BSW\_00469)

#### [SWS\_CorTst\_01001]

[ In case no errors occurred during test, the `CorTst_Start` function shall report the extended production error `CORTST_E_CORE_FAILURE` (see [ECUC\\_CorTst\\_00157](#)) as `DEM_EVENT_STATUS_PREPASSED` to the DEM. ] (SRS\_BSW\_00339, SRS\_BSW\_00422, SRS\_BSW\_00409, SRS\_BSW\_00466, SRS\_BSW\_00469)

#### [SWS\_CorTst\_00161]

[ 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.  
J (SRS\_BSW\_00323)

### 8.3.9 CorTst\_GetVersionInfo

#### [SWS\_CorTst\_00112] [

<b>Service name:</b>	CorTst_GetVersionInfo
<b>Syntax:</b>	void CorTst_GetVersionInfo( Std_VersionInfoType* versioninfo )
<b>Service ID[hex]:</b>	0x08
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	versioninfo   Pointer to where to store the version information of this module.
<b>Return value:</b>	None
<b>Description:</b>	Service returns the version information of this module.

] (SRS\_BSW\_00004, SRS\_BSW\_00407, SRS\_BSW\_00003, SRS\_BSW\_00411)

#### [SWS\_CorTst\_00118] [

If the function CorTst\_GetVersionInfo is called with a NULL pointer as parameter, it shall return immediately without any further action. If DET is enabled, this function shall report the error value CORTST\_E\_PARAM\_POINTER to the DET module, before returning without any further action. ](SRS\_BSW\_00407)

## 8.4 Call-back notifications

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

## 8.5 Scheduled functions

For details refer to the chapter 8.5 “Scheduled functions” in [SWS\\_BSWGeneral](#)

### 8.5.1 CorTst\_MainFunction

#### [SWS\_CorTst\_00067] [

<b>Service name:</b>	CorTst_MainFunction
<b>Syntax:</b>	void CorTst_MainFunction( void )
<b>Service ID[hex]:</b>	0x0b
<b>Description:</b>	Cyclically called by scheduler to perform processing of Core Test.

] (SRS\_BSW\_00433, SRS\_CoreTst\_14112, SRS\_CoreTst\_14114)

#### [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]

[ 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. ] (SRS\_CoreTst\_14133)

[SWS\_CorTst\_00155]

[ In case an error occurs during test, the `CorTest_MainFunction` function shall report the extended 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. ] (SRS\_BSW\_00339, SRS\_BSW\_00422, SRS\_BSW\_00409, SRS\_BSW\_00466, SRS\_BSW\_00469)

[SWS\_CorTst\_01002]

[ In case a core test is completed during a `CorTst_MainFunction` invocation and no errors occurred during this test, the `CorTst_MainFunction` function shall report the extended production error `CORTST_E_CORE_FAILURE` (see [ECUC\\_CorTst\\_00157](#)) as `DEM_EVENT_STATUS_PREPASSED` to the DEM.] (SRS\_BSW\_00339, SRS\_BSW\_00422, SRS\_BSW\_00409, SRS\_BSW\_00466, SRS\_BSW\_00469)

## 8.6 Expected Interfaces

This chapter lists all functions the Core Test module requires from other modules.

### 8.6.1 Mandatory Interfaces

This chapter lists all functions the Core Test module requires to fulfill its task.

[SWS\_CorTst\_00177] [

API function	Description
Dem_SetEventStatus	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling <code>Dem_SetEventStatus</code> can safely ignore the return value.

] (SRS\_BSW\_00339)

### 8.6.2 Optional Interfaces

This chapter lists all functions the Core Test module requires to fulfill an optional functionality.

[SWS\_CorTst\_00183] [

API function	Description
Det_ReportError	Service to report development errors.

] (SRS\_BSW\_00369, SRS\_BSW\_00350)

### 8.6.3 Configurable interfaces

In this chapter, all interfaces are listed where the target function could be configured. The target function is usually a callback function.

#### 8.6.3.1 CorTst Test Completed Notification

[SWS\_CorTst\_00076] [

<b>Service name:</b>	CorTst_TestCompletedNotification	
<b>Syntax:</b>	void CorTst_TestCompletedNotification( CorTst_ErrOkType ResultOfLastCorTstRun )	
<b>Service ID[hex]:</b>	0x0c	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	ResultOfLastCorTstRun	CORTST_E_OKAY Last Core Test execution successfully finished with no errors

		CORTST_E_NOT_OK Last Core Test execution finished with errors.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	The function CorTst_TestCompletedNotification shall be called every time when a complete test cycle has been executed.	

] (SRS\_BSW\_00359, SRS\_BSW\_00360, SRS\_CoreTst\_14119)

[SWS\_CorTst\_00077]

[ The Core Test module shall call the callback notification

CorTst\_TestCompletedNotification every time when it has executed a complete Core Test cycle based on a combination of atomic parts of Core Test in background mode. ] (SRS\_CoreTst\_14119, SRS\_SPAL\_00157)

[SWS\_CorTst\_00140]

[ The call of function CorTst\_TestCompletedNotification shall be pre compile time configurable by the configuration parameter

CorTstNotificationSupported. ] ()

## 9 Sequence diagrams

### 9.1 Initialization

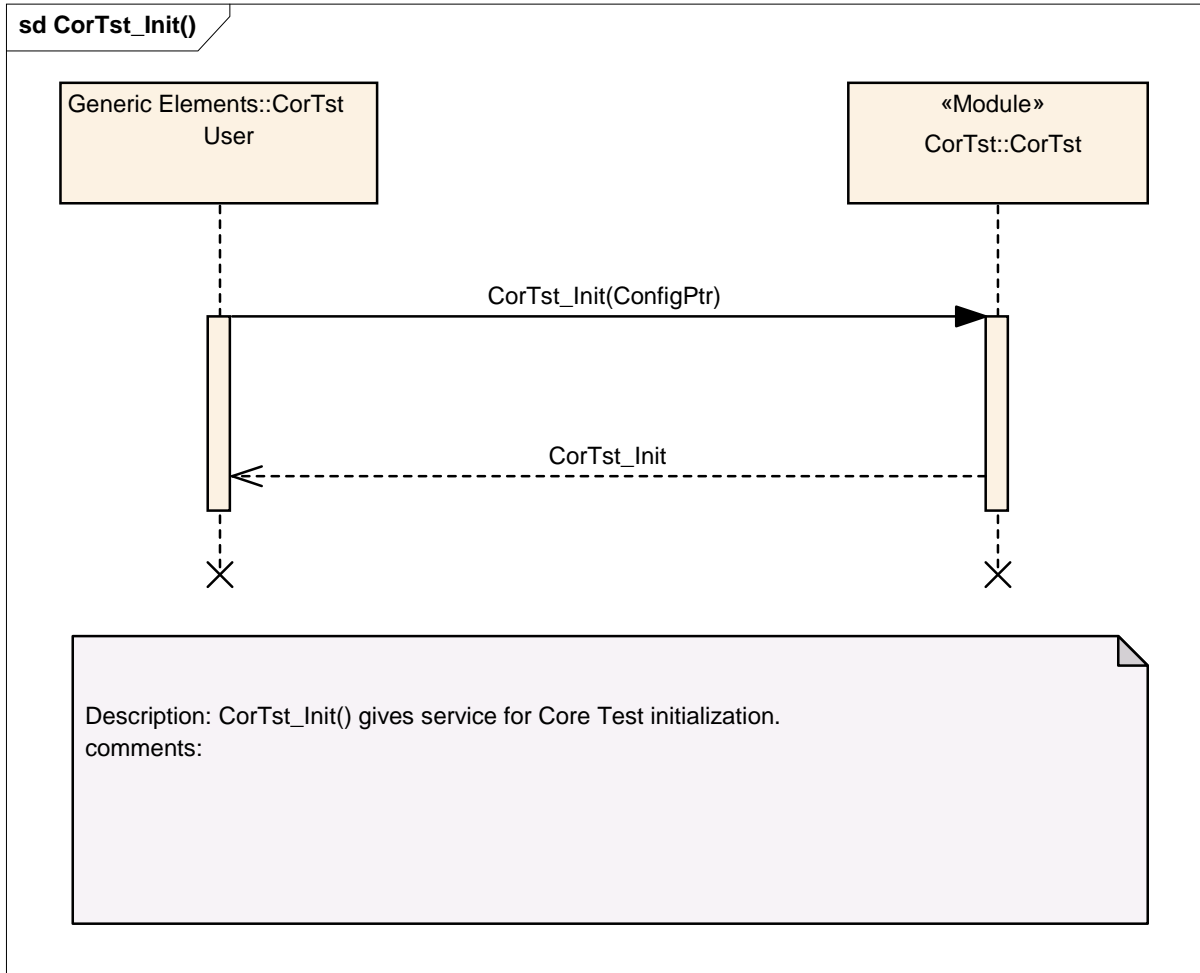


Figure 4 – Core Test Init

## 9.2 Deinitialization

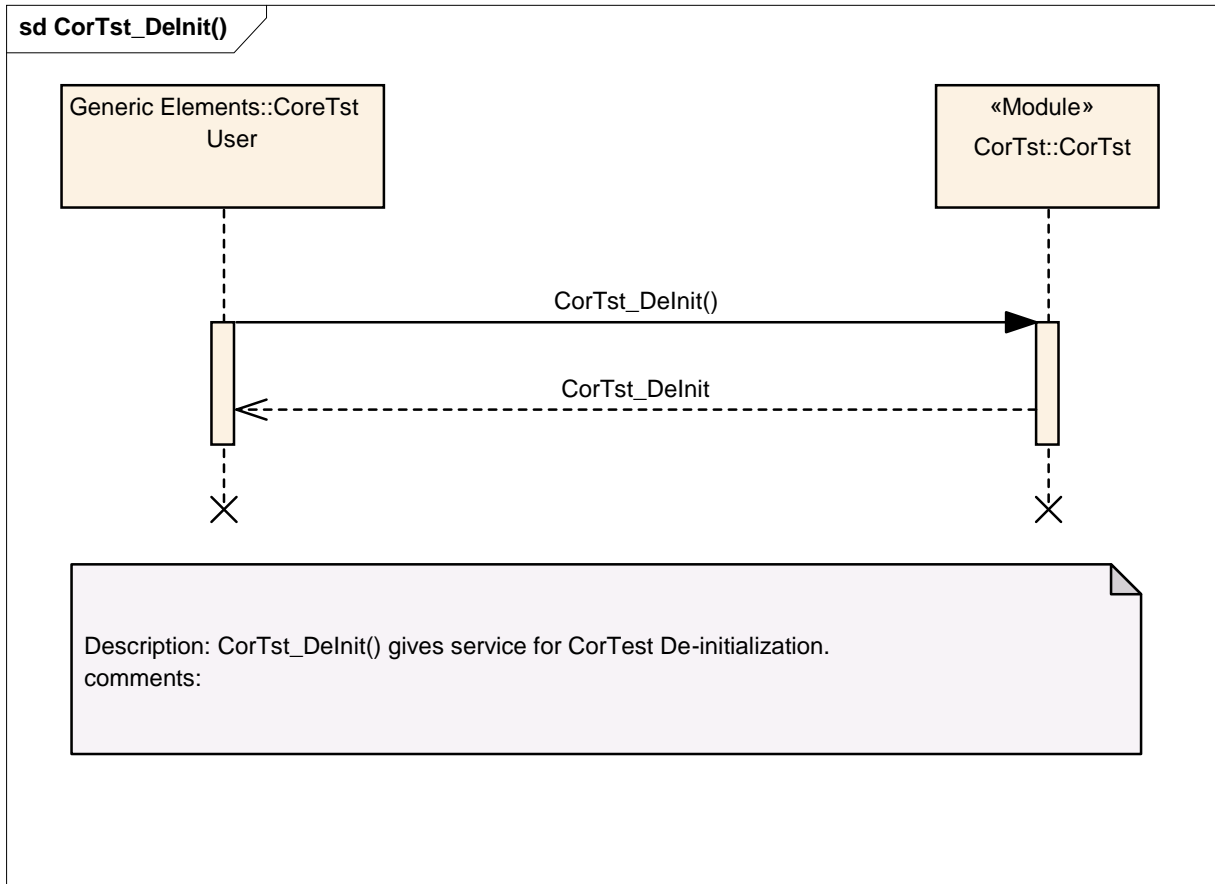


Figure 5 – Core Test De-initialization

### 9.3 Background Test

#### 9.3.1 Test Result Calculation within Core Test Module

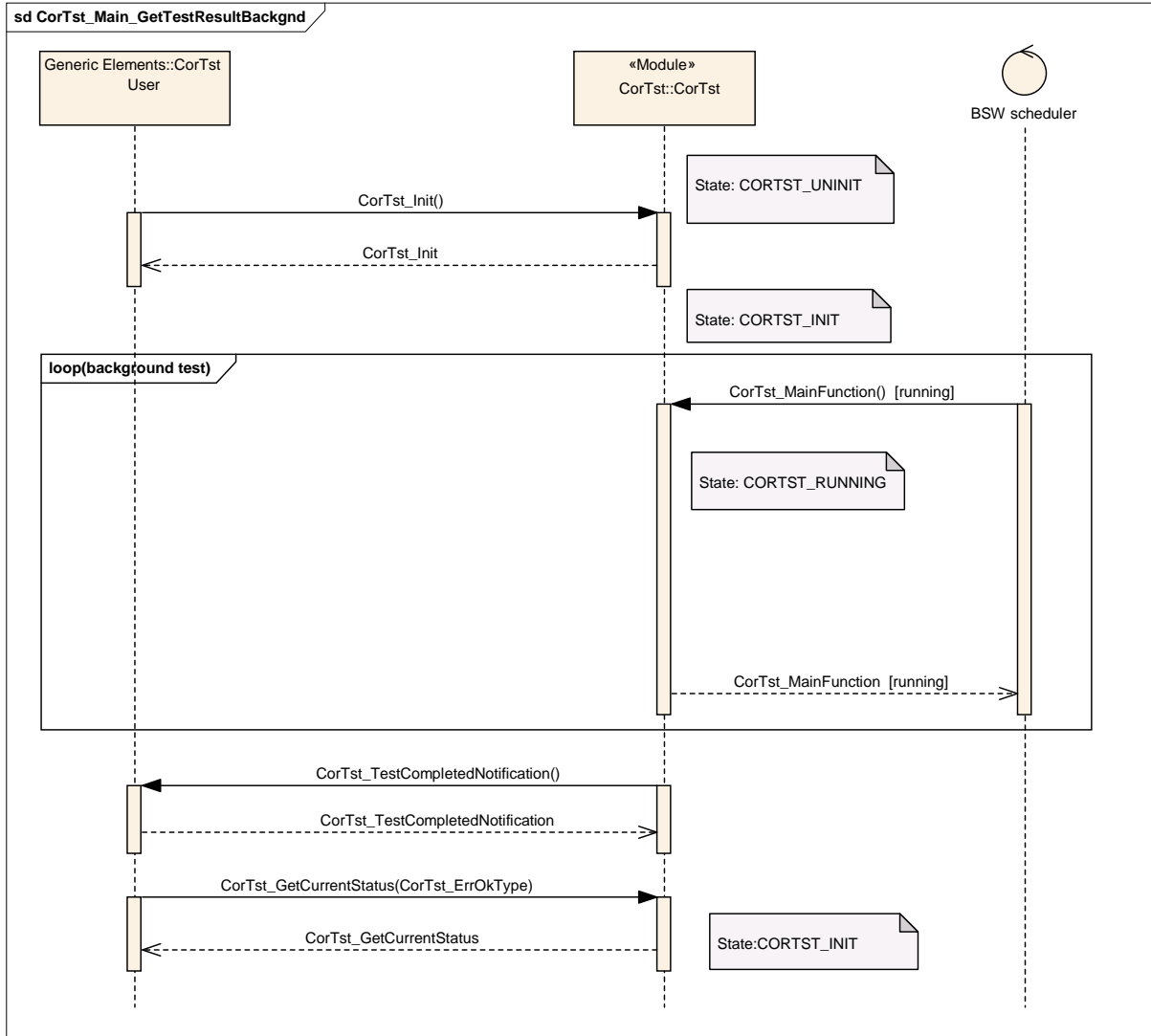


Figure 6 – Result Calculation within Core Test Driver



**9.3.2 Core Test Signature provided to Calling Entity**

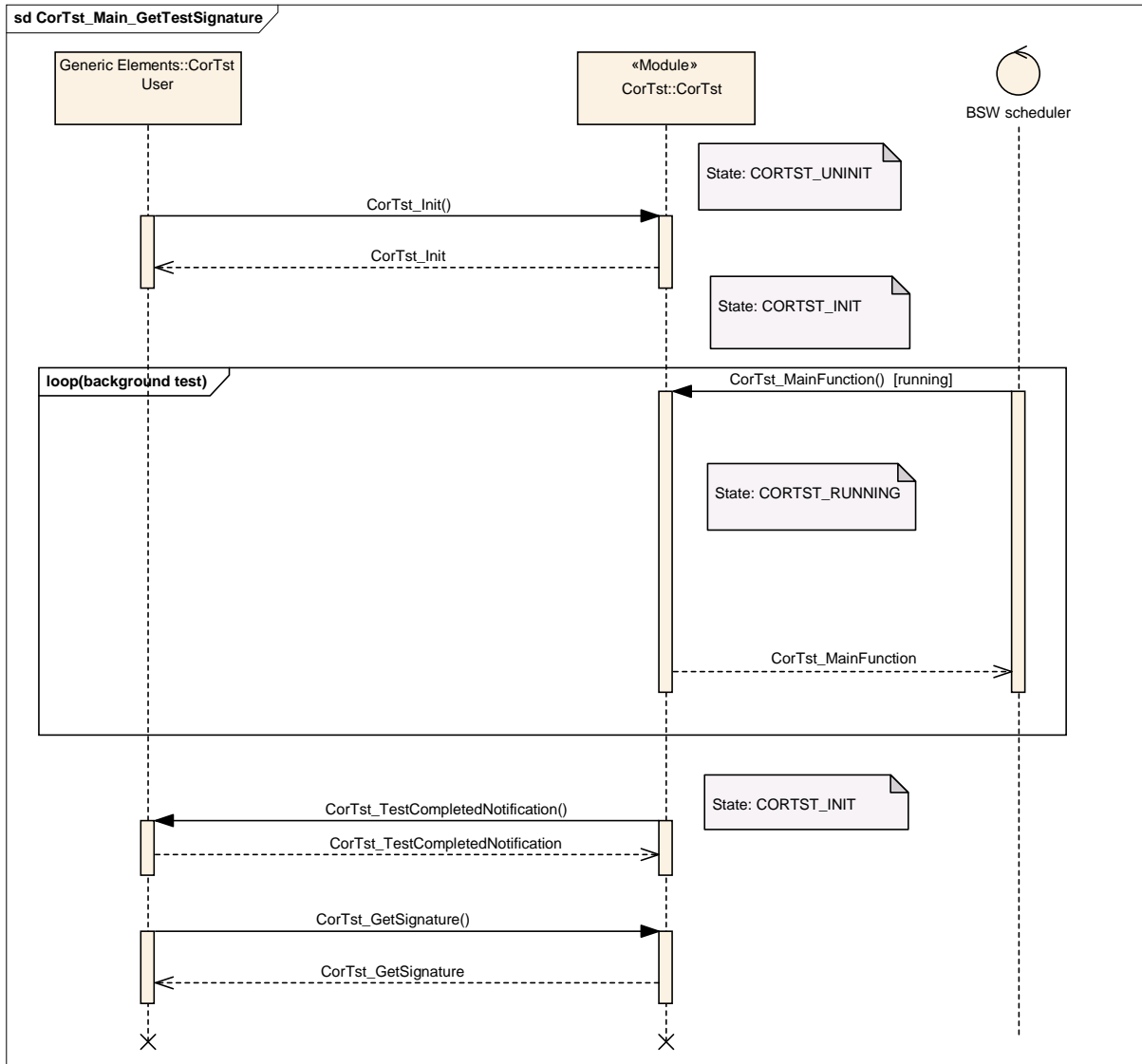


Figure 7 – Result Calculation on Calling Entity

## 10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers.

### 10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in [SWS\\_BSWGeneral](#)

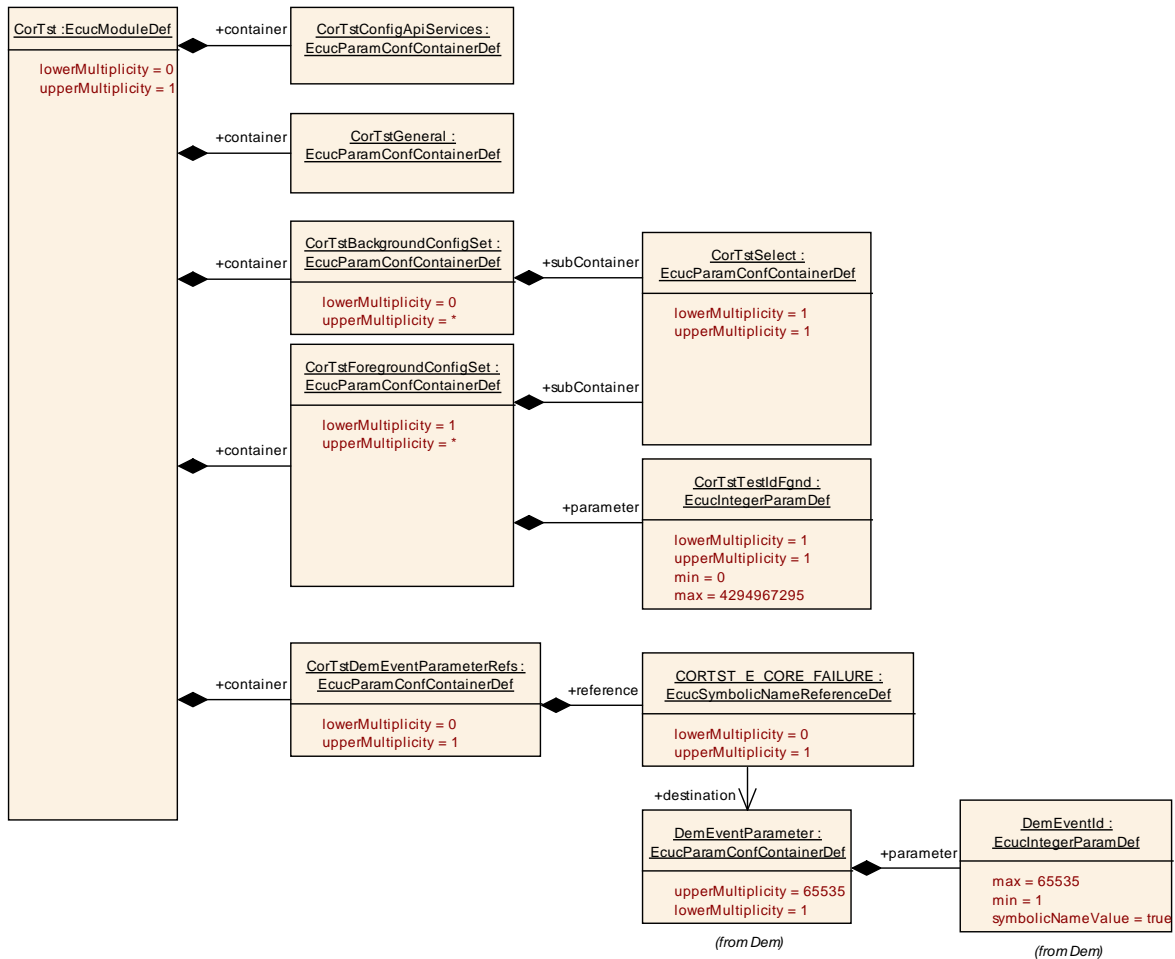
## 10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter [Functional specification](#) and Chapter [API specification](#).

### 10.2.1 CorTst

<b>SWS Item</b>	<b>ECUC_CorTst_00125 :</b>
<b>Module Name</b>	<i>CorTst</i>
<b>Module Description</b>	Configuration of the CorTst module.
<b>Post-Build Variant Support</b>	false
<b>Supported Config Variants</b>	VARIANT-LINK-TIME, VARIANT-PRE-COMPILE

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
CorTstBackgroundConfigSet	0..*	Multiple Configuration Set Container, defines background mode.
CorTstConfigApiServices	1	--
CorTstDemEventParameterReferences	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	--



### 10.2.2 CorTstGeneral

<b>SWS Item</b>	<b>ECUC_CorTst_00081 :</b>
<b>Container Name</b>	CorTstGeneral
<b>Description</b>	--
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_CorTst_00082 :</b>		
<b>Name</b>	CorTstDevErrorDetect		
<b>Parent Container</b>	CorTstGeneral		
<b>Description</b>	Switches the development error detection and notification on or off. <ul style="list-style-type: none"> <li>• true: detection and notification is enabled.</li> <li>• false: detection and notification is disabled.</li> </ul>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CorTst_00159 :</b>
<b>Name</b>	CorTstFgndTestNumber
<b>Parent Container</b>	CorTstGeneral

<b>Description</b>	This parameter holds the number of test configurations available for the foreground tests as defined in this configuration.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	1 .. 4294967295		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

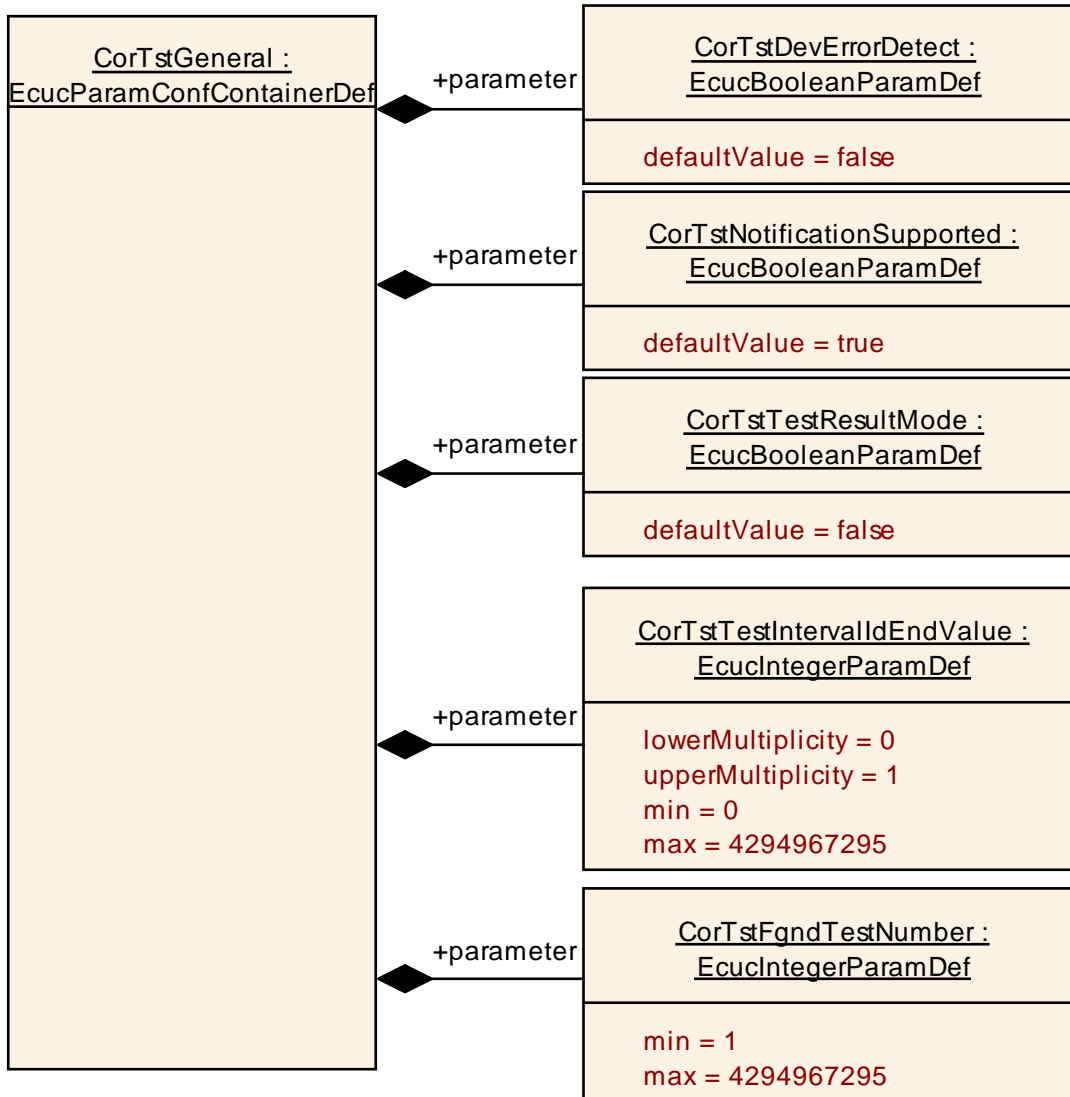
<b>SWS Item</b>	<b>ECUC_CorTst_00083 :</b>		
<b>Name</b>	CorTstNotificationSupported		
<b>Parent Container</b>	CorTstGeneral		
<b>Description</b>	Switch to indicate that the notification is supported.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	true		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CorTst_00143 :</b>		
<b>Name</b>	CorTstTestIntervalIdEndValue		
<b>Parent Container</b>	CorTstGeneral		
<b>Description</b>	Defines the end value of the Test Interval Id.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 4294967295		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CorTst_00086 :</b>		
<b>Name</b>	CorTstTestResultMode		
<b>Parent Container</b>	CorTstGeneral		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants

	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**



### 10.2.3 CorTstSelect

<b>SWS Item</b>	<b>ECUC_CorTst_00089 :</b>
<b>Container Name</b>	CorTstSelect
<b>Description</b>	This container specifies configuration parameters to select individual tests for foreground mode and background mode. The availability is hardware and implementation specific.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_CorTst_00130 :</b>
<b>Name</b>	CorTstAddress
<b>Parent Container</b>	CorTstSelect
<b>Description</b>	Enable/Disables core address test.
<b>Multiplicity</b>	1

<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CorTst_00129 :</b>		
<b>Name</b>	CorTstAlu		
<b>Parent Container</b>	CorTstSelect		
<b>Description</b>	Enable/Disables core ALU test.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CorTst_00133 :</b>		
<b>Name</b>	CorTstCache		
<b>Parent Container</b>	CorTstSelect		
<b>Description</b>	Enable/Disables core cache test.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CorTst_00128 :</b>		
<b>Name</b>	CorTstInterrupt		
<b>Parent Container</b>	CorTstSelect		
<b>Description</b>	Enable/Disables core interrupt test		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CorTst_00131 :</b>		
<b>Name</b>	CorTstMemoryIf		
<b>Parent Container</b>	CorTstSelect		
<b>Description</b>	Enable/Disables core memory interface test		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		

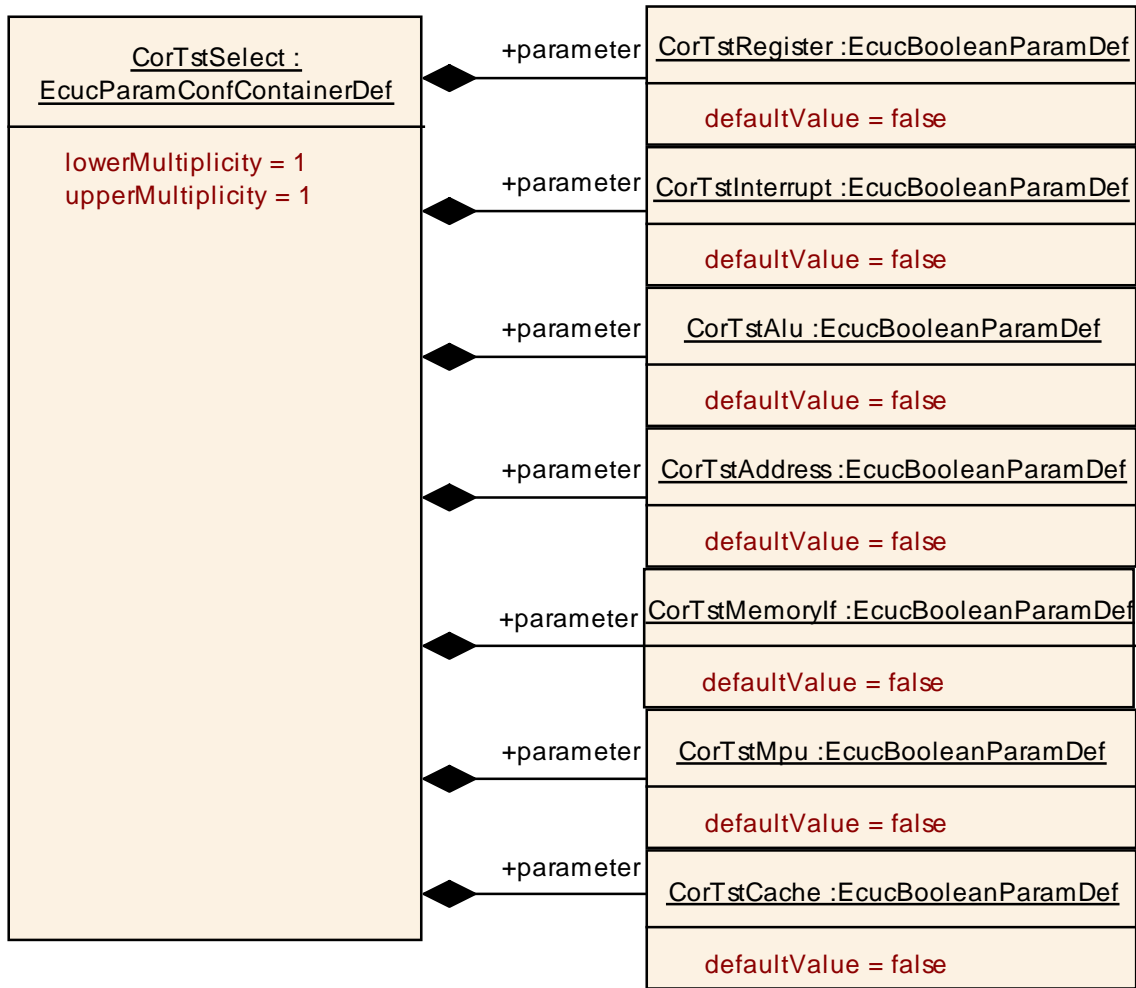
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CorTst_00132 :</b>		
<b>Name</b>	CorTstMpu		
<b>Parent Container</b>	CorTstSelect		
<b>Description</b>	Enable/Disables core MPU test		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CorTst_00127 :</b>		
<b>Name</b>	CorTstRegister		
<b>Parent Container</b>	CorTstSelect		
<b>Description</b>	Enable/Disables core register test		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>No Included Containers</b>
-------------------------------





### 10.2.4 CorTstBackgroundConfigSet

<b>SWS Item</b>	ECUC_CorTst_00087 :
<b>Container Name</b>	CorTstBackgroundConfigSet
<b>Description</b>	Multiple Configuration Set Container, defines background mode.
<b>Configuration Parameters</b>	

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

### 10.2.5 CorTstForegroundConfigSet

<b>SWS Item</b>	ECUC_CorTst_00088 :
<b>Container Name</b>	CorTstForegroundConfigSet
<b>Description</b>	Multiple Configuration Set Container , defines foreground mode.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	ECUC_CorTst_00158 :
<b>Name</b>	CorTstTestIdFgnd

<b>Parent Container</b>	CorTstForegroundConfigSet		
<b>Description</b>	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).		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 4294967295		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
CorTstSelect	1	This container specifies configuration parameters to select individual tests for foreground mode and background mode. The availability is hardware and implementation specific.

### 10.2.6 CorTstConfigApiServices

<b>SWS Item</b>	ECUC_CorTst_00092 :		
<b>Container Name</b>	CorTstConfigApiServices		
<b>Description</b>	--		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	ECUC_CorTst_00094 :		
<b>Name</b>	CorTstAbortApi		
<b>Parent Container</b>	CorTstConfigApiServices		
<b>Description</b>	Adds / removes the service CorTst_Abort() from the code.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	ECUC_CorTst_00104 :		
<b>Name</b>	CorTstGetCurrentStatus		
<b>Parent Container</b>	CorTstConfigApiServices		
<b>Description</b>	Adds / removes the service CorTst_GetCurrentStatus() from the code.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	ECUC_CorTst_00103 :		
<b>Name</b>	CorTstGetFgndSignature		
<b>Parent Container</b>	CorTstConfigApiServices		
<b>Description</b>	Adds / removes the service CorTst_GetFgndSignature() from the code.		

<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CorTst_00097 :</b>		
<b>Name</b>	CorTstGetSignature		
<b>Parent Container</b>	CorTstConfigApiServices		
<b>Description</b>	Adds / removes the service CorTst_GetSignature() from the code.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

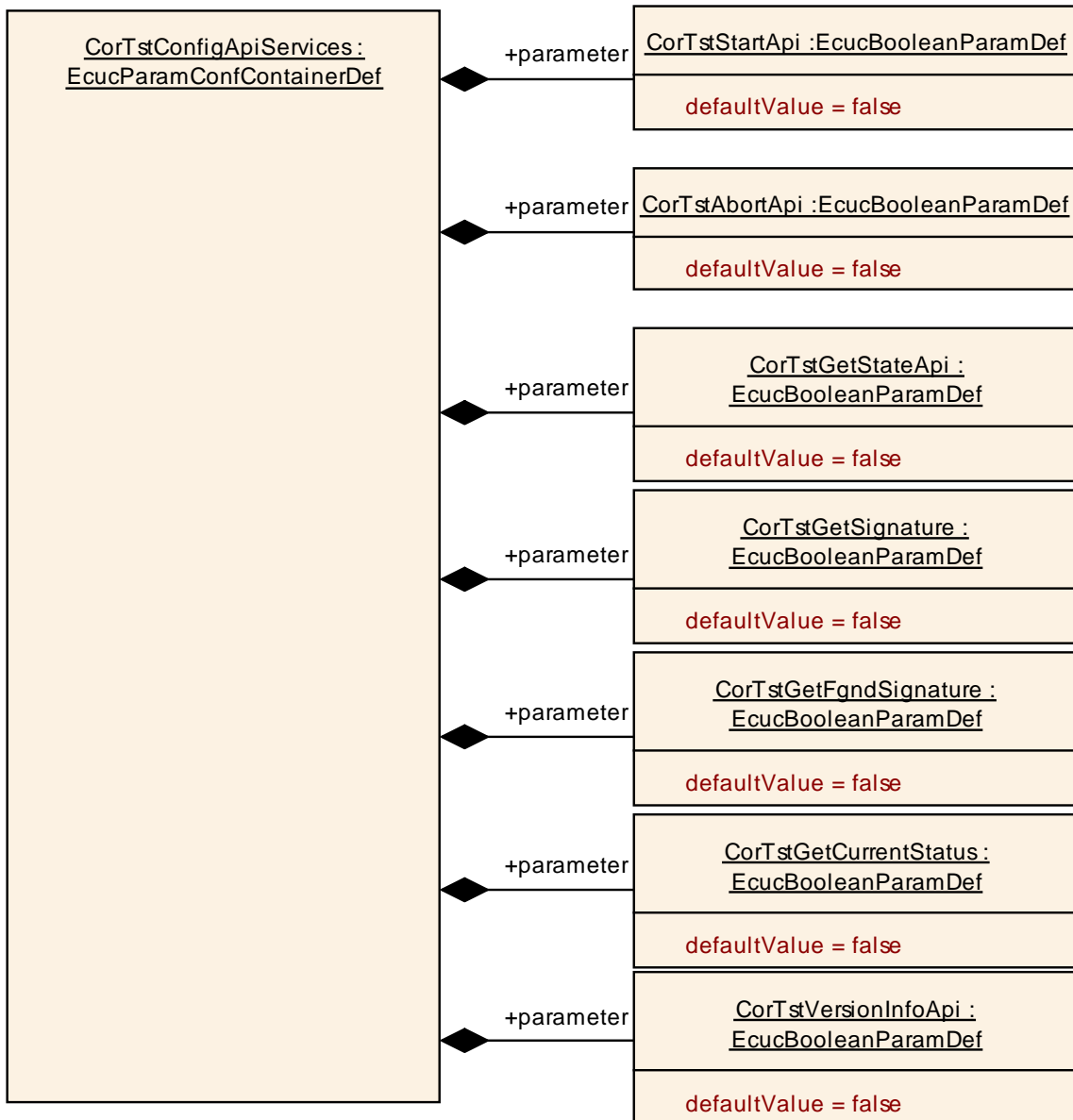
<b>SWS Item</b>	<b>ECUC_CorTst_00096 :</b>		
<b>Name</b>	CorTstGetStateApi		
<b>Parent Container</b>	CorTstConfigApiServices		
<b>Description</b>	Adds / removes the service CorTst_GetState() from the code.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CorTst_00093 :</b>		
<b>Name</b>	CorTstStartApi		
<b>Parent Container</b>	CorTstConfigApiServices		
<b>Description</b>	Adds / removes the service CorTst_Start() from the code.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CorTst_00098 :</b>		
<b>Name</b>	CorTstVersionInfoApi		
<b>Parent Container</b>	CorTstConfigApiServices		
<b>Description</b>	Adds / removes the service CorTst_GetVersionInfo() from the code.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		

<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**



### 10.2.7 CorTstDemEventParameterRefs

<b>SWS Item</b>	ECUC_CorTst_00156 :
<b>Container Name</b>	CorTstDemEventParameterRefs
<b>Description</b>	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.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_CorTst_00157 :</b>		
<b>Name</b>	CORTST_E_CORE_FAILURE		
<b>Parent Container</b>	CorTstDemEventParameterRefs		
<b>Description</b>	Reference to the DemEventParameter which shall be issued when the error "CORE failure" has occurred.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: Dem		

<b>No Included Containers</b>
-------------------------------

### 10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in [SWS\\_BSWGeneral](#)

## 11 Not applicable requirements

**[SWS\_CorTst\_00999]** [ These requirements are not applicable to this specification. ] (SRS\_BSW\_00167, SRS\_BSW\_00168, SRS\_BSW\_00339, SRS\_BSW\_00344, SRS\_BSW\_00375, SRS\_BSW\_00383, SRS\_BSW\_00386, SRS\_BSW\_00398, SRS\_BSW\_00399, SRS\_BSW\_00404, SRS\_BSW\_00405, SRS\_BSW\_00409, SRS\_BSW\_00416, SRS\_BSW\_00417, SRS\_BSW\_00422, 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\_00005, SRS\_BSW\_00006, SRS\_BSW\_00009, SRS\_BSW\_00010, SRS\_BSW\_00161, SRS\_BSW\_00162, SRS\_BSW\_00170, SRS\_BSW\_00171, SRS\_BSW\_00172, SRS\_BSW\_00301, SRS\_BSW\_00302, SRS\_BSW\_00306, SRS\_BSW\_00308, SRS\_BSW\_00309, SRS\_BSW\_00310, SRS\_BSW\_00312, SRS\_BSW\_00314, SRS\_BSW\_00318, SRS\_BSW\_00321, SRS\_BSW\_00325, SRS\_BSW\_00328, SRS\_BSW\_00330, SRS\_BSW\_00333, SRS\_BSW\_00334, SRS\_BSW\_00341, SRS\_BSW\_00346, SRS\_BSW\_00371, SRS\_BSW\_00374, SRS\_BSW\_00378, SRS\_BSW\_00379, SRS\_BSW\_00413, SRS\_CoreTst\_14125, SRS\_CoreTst\_14124)