Document Title	Specification of Core Test
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	259
Document Classification	Standard

Document Version	1.2.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History

Date	Version	Changed by	Change Description
23.09.2011	1.2.0	AUTOSAR Administration	• Clarification of some requirements. • Typos correction. • Removed redundant and useless requirements.
15.11.2010	1.1.0	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 unused types.
30.11.2009	1.0.0	AUTOSAR Administration	Initial release

Disclaimer

Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 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

Abbreviation / Acronym:	*Description:*
MCAL	Microcomputer Abstraction Layer
DEM	Diagnostic Event Manager
DET	Development 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.

Term:	*Description:*
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	*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.

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_ECUStateManager.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

3.2 Related standards and norms

[9] ISO DIS 26262, www.iso.org

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:

- DET: Development Error Tracer: DET services will be called in case of Development errors.
- Production Errors will be reported to Diagnostic Event Manager (DEM)
- 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

[CorTst002]

「 The Core Test module shall provide interrupt service routines for test purposes only. 」(BSW164, BSW14105)

[CorTst003]

「 The Core Test source code module file shall be named

- CorTst.c – for source code of the core test module 」()

5.1.2 Header file structure

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

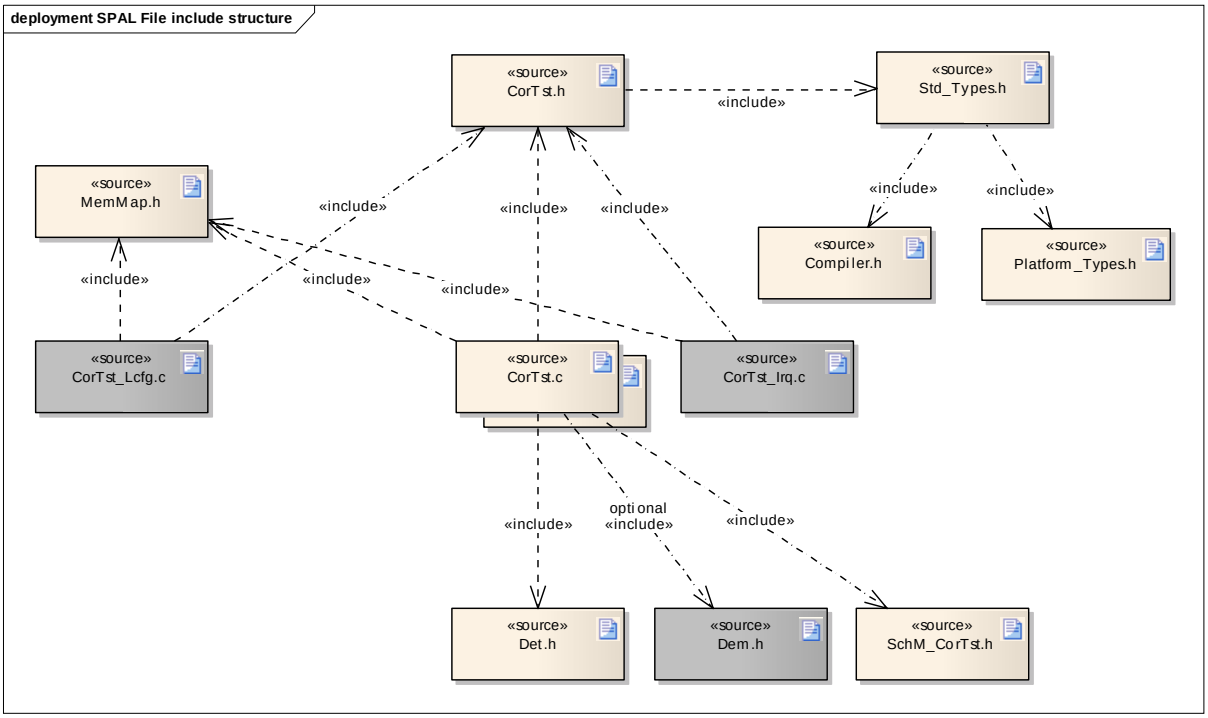


Figure 1 – File structure

[CorTst006]
「 The file CorTst.h only contains external declarations of constants, global data, type definitions and services that are specified in the Core Test source code module driver SWS. 」(BSW00380, BSW00381)

[CorTst007] 「
Constants, global data types and functions that are only used by Core Test driver source code module internally, are declared in CorTst.c. 」(BSW00380, BSW00381)

6 Requirements traceability

Requirement	Satisfied by
-	CorTst050
-	CorTst152
-	CorTst077
-	CorTst024
-	CorTst144
-	CorTst145
-	CorTst074
-	CorTst003
-	CorTst061
-	CorTst010
-	CorTst109
-	CorTst065
-	CorTst147
-	CorTst013
-	CorTst049
-	CorTst0174
-	CorTst072
-	CorTst070
-	CorTst146
-	CorTst0176
-	CorTst160
-	CorTst149
-	CorTst068
-	CorTst073
-	CorTst138
-	CorTst023
-	CorTst042
-	CorTst122
-	CorTst056
-	CorTst148
-	CorTst153
-	CorTst115
-	CorTst051
-	CorTst045
-	CorTst118
-	CorTst0178
-	CorTst011
-	CorTst121

-	CorTst105
-	CorTst136
-	CorTst008
-	CorTst120
-	CorTst0175
-	CorTst041
-	CorTst140
-	CorTst022
-	CorTst0179
-	CorTst012
-	CorTst052
-	CorTst154
-	CorTst009
-	CorTst113
-	CorTst020
-	CorTst014
-	CorTst069
-	CorTst155
-	CorTst071
-	CorTst054
-	CorTst047
-	CorTst021
-	CorTst044
-	CorTst058
BSW003	CorTst112
BSW00301	CorTst999
BSW00302	CorTst999
BSW00304	CorTst027
BSW00306	CorTst999
BSW00308	CorTst999
BSW00309	CorTst999
BSW00310	CorTst999
BSW00312	CorTst999
BSW00314	CorTst999
BSW00318	CorTst999
BSW00321	CorTst999
BSW00323	CorTst161
BSW00325	CorTst999
BSW00327	CorTst016
BSW00328	CorTst999
BSW00329	CorTst999
BSW00330	CorTst999

BSW00331	CorTst038, CorTst039, CorTst037
BSW00333	CorTst999
BSW00334	CorTst999
BSW00336	CorTst046
BSW00337	CorTst016, CorTst015
BSW00338	CorTst183, CorTst017
BSW00339	CorTst999
BSW00341	CorTst999
BSW00344	CorTst999
BSW00346	CorTst999
BSW00350	CorTst183
BSW00355	CorTst999
BSW00357	CorTst064
BSW00358	CorTst040
BSW00359	CorTst076
BSW00360	CorTst076
BSW00369	CorTst183, CorTst017, CorTst019
BSW00370	CorTst999
BSW00371	CorTst999
BSW00374	CorTst999
BSW00375	CorTst999
BSW00378	CorTst999
BSW00379	CorTst999
BSW00380	CorTst006, CorTst007
BSW00381	CorTst006, CorTst007
BSW00383	CorTst999
BSW00385	CorTst016
BSW00386	CorTst999
BSW00398	CorTst999
BSW00399	CorTst999
BSW004	CorTst112
BSW00404	CorTst999
BSW00405	CorTst999
BSW00406	CorTst018, CorTst040
BSW00407	CorTst112
BSW00409	CorTst999
BSW00411	CorTst112
BSW00413	CorTst999
BSW00414	CorTst040
BSW00416	CorTst999
BSW00417	CorTst999
BSW00422	CorTst999

BSW00423	CorTst999
BSW00424	CorTst999
BSW00425	CorTst999
BSW00426	CorTst999
BSW00428	CorTst999
BSW00429	CorTst999
BSW00431	CorTst999
BSW00432	CorTst999
BSW00433	CorTst067
BSW00434	CorTst999
BSW00436	CorTst999
BSW00437	CorTst999
BSW00438	CorTst999
BSW005	CorTst999
BSW006	CorTst999
BSW009	CorTst999
BSW010	CorTst999
BSW101	CorTst040
BSW14105	CorTst002
BSW14112	CorTst064, CorTst067
BSW14113	CorTst064
BSW14114	CorTst067
BSW14115	CorTst057, CorTst060
BSW14116	CorTst057, CorTst060
BSW14117	CorTst016
BSW14118	CorTst053
BSW14119	CorTst076
BSW14124	CorTst999
BSW14125	CorTst999
BSW14126	CorTst048
BSW14130	CorTst026
BSW14131	CorTst055
BSW14133	CorTst137, CorTst139
BSW161	CorTst999
BSW162	CorTst999
BSW164	CorTst002
BSW167	CorTst999
BSW168	CorTst999
BSW170	CorTst999
BSW171	CorTst999
BSW172	CorTst999

Document: AUTOSAR requirements on Basic Software, general

Requirement	***Satisfied by***
Functional Requirements	
[BSW101] Initialization interface	CorTst040
[BSW004] Version check	CorTst112
[BSW159] Tool-based configuration	Both static and runtime configuration parameters are located outside the source code of the module. This is the prerequisite for automatic configuration.
[BSW167] Static configuration checking	Not applicable (requirement on configuration tool)
[BSW168] Diagnostic interface of SW components	Not applicable
[BSW00323] API parameter checking	CorTst161
[BSW00336] Shutdown interface	CorTst046
[BSW00337] Classification of errors	CorTst015: CorTst016:
[BSW00338] Detection and reporting of development errors	CorTst017:
[BSW00339] Reporting of production relevant error status	Not applicable (this module does not need such a function)
[BSW00344] Reference to link-time configuration	Not applicable
[BSW00345] Pre-compile-time configuration	§5.2 Header file structure.
[BSW00369] Do not return development error codes via API	CorTst017: CorTst019:
[BSW00375] Notification of wake-up reason	Not applicable (wakeups are not supported by this module)
[BSW00380] Separate C-files for configuration parameters	CorTst004 CorTst006 CorTst007
[BSW00381] Separate configuration header file for pre-compile time parameters	CorTst004 CorTst006 CorTst007
[BSW00383] List dependencies of configuration files	Not applicable (there are no dependencies to other configuration files)
[BSW00384] List dependencies to other modules	See chapter 5.
[BSW00385] List possible error notifications	CorTst016:
[BSW00386] Configuration for detecting an error	Not applicable (no configuration for error detection)
[BSW00387] Specify the configuration class of callback function	This version supports only pointer at link time.
[BSW00388] Introduce containers	See chapter 10.2
[BSW00389] Containers shall have names	See chapter 10.2
[BSW00390] Parameter content shall be unique within the module	See chapter 10.2
[BSW00391] Parameter shall have unique names	Prefix “CorTst” added to each parameter
[BSW00392] Parameters shall have a type	See chapter 8.2 and 10.2
[BSW00393] Parameters shall have a range	See chapter 8.2 and 10.2
[BSW00394] Specify the scope of the parameters	“Local” marked as Module. See chapter 10.2
[BSW00395] List the required parameters (per parameter)	See chapter 10.2
[BSW00396] Configuration classes	See chapter 10.2
[BSW00397] Pre-compile-time parameters	See chapter 10.2
[BSW00398] Link-time parameters	Not applicable (Module does not support link-time configuration)
[BSW00399] Loadable post-build time parameters	Not applicable (Module does not support post build-time configuration)
[BSW00400] Selectable post-build time parameters	(Module does not support post build-time configuration)

[BSW00402] Published information	Only if delivered in source code and CorTst126
[BSW00404] Reference to post build time configuration	Not applicable (post build time is not supported)
[BSW00405] Reference to multiple configuration sets	Not applicable (post build time is not supported)
[BSW00406] Check module initialization	CorTst040, CorTst018, CorTst170
[BSW00407] Function to read out published parameters	CorTst112
[BSW00409] Header files for production code error IDs	Not applicable (production code error IDs are not supported)
[BSW00412] Separate H-file for configuration parameters	See figure in Header file structure
[BSW00416] Sequence of Initialization	Not applicable (this is a general software integration requirement)
[BSW00417] Reporting of error events by non-basic software	Not applicable (this is a basic software module)
[BSW00419] Separate C-files for pre-compile time configuration parameters	See figure in Header file structure
[BSW00422] Debouncing of production relevant error status	Not applicable (it makes no sense to debounce core error)
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR interfaces	Not applicable (this module has no connection to the RTE)
[BSW00424] BSW main processing function task allocation	Not applicable (the scheduling of a BSW is not part of this SWS)
[BSW00425] Trigger conditions for schedulable objects	Not applicable (requirement for the implementer)
[BSW00426] Exclusive areas in BSW modules	Not applicable (requirement for the implementer)
[BSW00427] ISR description for BSW modules	
[BSW00428] Execution order dependencies of main processing functions	Not applicable (requirement for the implementer and integrator)
[BSW00429] Restricted BSW OS functionality access	Not applicable (this module does not use OS services)
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (this is a special requirement for the BSW scheduler)
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (this module does not have send/receive functionality)
[BSW00433] Calling of main processing functions	CorTst067
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (this is a special requirement for the BSW scheduler)
[BSW00437] No-init area in RAM	Not applicable (this is a requirement on the memory manager)
[BSW00438] Post-build configuration data structure	Not applicable (post build time configuration is not supported)
Non-functional Requirements	
[BSW003] Version identification	CorTst112
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable (this is a requirement on architecture)
[BSW006] Platform independency	Not applicable (Core Test is heavily dependent on underlying hardware to be tested)
[BSW007] HIS MISRA C	Common AUTOSAR non-functional requirement for the implementer.
[BSW009] Module User Documentation	Not applicable (requirement for the implementer)
[BSW010] Memory resource documentation	Not applicable

	(requirement for the implementer)
[BSW158] Separation of configuration from implementation	CorTst001:
[BSW160] Human-readable configuration data	Common AUTOSAR non-functional requirement for the implementer.
[BSW161] Microcontroller abstraction	Not applicable (this is a requirement on architecture)
[BSW162] ECU layout abstraction	Not applicable (this is a requirement on architecture)
[BSW164] Implementation of service routines	CorTst002: (interrupt service routine for testing purposes)
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable (not a SW-Component)
[BSW171] Configurability of optional functionality	Not applicable (no optional functionality available)
[BSW172] Compatibility and documentation of scheduling strategy	Not applicable (requirement for the implementer)
[BSW00300] Module naming convention	Applicable. Common AUTOSAR non-functional requirement for the implementer.
[BSW00301] Limit imported information	Not applicable (requirement for the implementer)
[BSW00302] Limit exported information	Not applicable (requirement for the implementer)
[BSW00304] AUTOSAR integer data types	CorTst027:
[BSW00305] Self-defined data types naming convention	applicable (requirement for the implementer)
[BSW00306] Avoid direct use of compiler and platform specific keywords	Not applicable (requirement for the implementer)
[BSW00307] Global variables naming convention	Common AUTOSAR non functional requirement for the implementer.
[BSW00308] Definition of global data	Not applicable (requirement for the implementer)
[BSW00309] Global data with read-only constraint	Not applicable (requirement for the implementer)
[BSW00310] API naming convention	applicable
[BSW00312] Shared code shall be reentrant	Not applicable (requirement for the implementer)
[BSW00314] Separation of interrupt frames and service routines	Not applicable (interrupt service routine for testing purposes)
[BSW00318] Format of module version numbers	Not applicable (requirement for the implementer)
[BSW00321] Enumeration of module version numbers	Not applicable (requirement for the implementer)
[BSW00325] Runtime of interrupt service routines	Not applicable (requirement for the implementer)
[BSW00326] Transition from ISRs to OS tasks	applicable (requirement for the implementer)
[BSW00327] Error values naming convention	Cortst016
[BSW00328] Avoid duplication of code	Not applicable (requirement for the implementer)
[BSW00329] Avoidance of generic interfaces	Not applicable (no generic interface are available)
[BSW00330] Usage of macros / inline functions instead of functions	Not applicable (requirement for the implementer)
[BSW00331] Separation of error and status values	CorTst037 CorTst038 CorTst039
[BSW00333] Documentation of callback function context	Not applicable (requirement for the implementer)
[BSW00334] Provision of XML file	Not applicable (requirement for the implementer)

[BSW00335] Status values naming convention	Fulfilled for all defined status types see 8.2
[BSW00341] Microcontroller compatibility documentation	Not applicable (requirement for the implementer)
[BSW00342] Usage of source code and object code	Common AUTOSAR non-functional requirement for the implementer.
[BSW00343] Specification and configuration of time	Common AUTOSAR non-functional requirement for the implementer.
[BSW00346] Basic set of module files	Not applicable (requirement for the implementer)
[BSW00347] Naming separation of different instances of BSW drivers	Common AUTOSAR non-functional requirement for the implementer and integrator.
[BSW00348] Standard type header	Fulfilled for all defined status types see 8.2
[BSW00350] Development error detection keyword	CorTst082_Conf
[BSW00353] Platform specific type header	§5.2 Header file structure.
[BSW00355] Do not redefine AUTOSAR integer data types	Not applicable (requirement for the implementer)
[BSW00357] Standard API return type	CorTst064
[BSW00358] Return type of init() functions	CorTst040
[BSW00359] Return type of callback functions	CorTst076
[BSW00360] Parameters of callback functions	CorTst076
[BSW00361] Compiler specific language extension header	§5.2 Header file structure.
[BSW00370] Separation of callback interface from API	Not applicable (the notification functions will be handled via a function pointer in the configuration init structure)
[BSW00371] Do not pass function pointers via API	Not applicable (requirement for the implementer)
[BSW00373] Main processing function naming convention	See section 8.5.1, CorTst_MainFunction
[BSW00374] Module vendor identification	Not applicable (requirement for the implementer)
[BSW00376] Return type and parameters of main processing functions	See section CorTst_MainFunction
[BSW00377] Module specific API return types	See section 8.2
[BSW00378] AUTOSAR Boolean type	Not applicable (requirement for the implementer)
[BSW00379] Module identification	Not applicable (requirement for the implementer)
[BSW00401] Documentation of multiple instances of configuration parameters	Containers and configuration parameters
[BSW00408] Configuration parameter naming convention	See section Containers and configuration parameters
[BSW00410] Compiler switches shall have defined values	See section Containers and configuration parameters
[BSW00411] Get version info keyword	CorTst112:
[BSW00413] Accessing instances of BSW modules	Not applicable (instances makes no sense for this module)
[BSW00414] Parameter of init function	CorTst040
[BSW00415] User dependent include files	See figure in Header file structure
[BSW00435] Module header file structure for the basic software scheduler	See figure in Header file structure
[BSW00436] Module header file structure for the basic software memory mapping	Not applicable (requirement for the implementer)

Document: AUTOSAR requirements on Basic Software, cluster MCAL, Core Test driver module

Requirement	*Satisfied by*
[BSW14101] The Core Test Shall Be Configurable	See section Containers and configuration parameters
[BSW14102] Link Time Configuration Shall Be Supported	See section Containers and configuration parameters
[BSW14104] Core Register Test Shall Be Available	[CorTst029]
[BSW14105] Core Interrupt and Exception Detection Tests Shall Be Available	CorTst002, [CorTst030]
[BSW14106] Core ALU Test Shall Be Available	[CorTst032]
[BSW14107] Core Address Generator Test Shall Be Available	[CorTst033]
[BSW14108] Core Memory Interfaces Test Shall Be Available	[CorTst034]
[BSW14109] Memory Protection Unit (MPU) Test Shall Be Available	[CorTst035]
[BSW14110] Cache Controller Test Shall Be Available	[CorTst036]
[BSW14111] The Core Test Shall Be Divided into Atomic Sequences	Implementation specific
[BSW14112] There Shall Be a Single API for the Core Test Service	[CorTst064], [CorTst067]
[BSW14113] The API Shall Have a Parameter to Select Which Component Shall Be Tested	[CorTst064]
[BSW14114] A Main Function for the Core Test Shall Be Available	[CorTst067]
[BSW14115] Test Metrics Shall Be Available to Caller	[CorTst057], [CorTst060]
[BSW14116] The Test Computes a Checksum/Signature as Test Result	[CorTst057], [CorTst060]
[BSW14131] The Test Computes a Pass/Fail Status Representation as a test result	[CorTst055]
[BSW14117] Faults Shall Be Treated as Production Errors	CorTst173
[BSW14118] Test Status Polling	[CorTst053]
[BSW14119] A Notification of Completion Shall Be Provided	[CorTst076]
[BSW14126] It Shall Be Possible to Cancel a Running Test	[CorTst048]
[BSW14130] Destructive Test Shall Restore Original State of tested Entity	[CorTst026]
[BSW14123] Shared Resources to Be Tested Shall Be Made Exclusively Available to Test	Prerequisite to Core Test Module, shall be handled by upper AUTOSAR layers.
[BSW14125] Diagnostic Coverage	Not applicable for an API
[BSW14124] Compliance to The Automotive Standard	Not applicable for an API
BSW14133 Core Test Interval Id	CorTst137 CorTst139

7 Functional specification

7.1 General Behavior

[CorTst008]
⌈ Core Test shall provide a procedure to test all CPU registers. ⌋()

[CorTst009] ⌈
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. ⌋()

[CorTst010]
⌈ The Core Test shall provide an Arithmetic and Logical Unit (ALU) test. ⌋()

[CorTst011]
⌈ The Core Test shall provide an address generation test. ⌋()

[CorTst012]
⌈ 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. ⌋()

[CorTst013]
⌈ The Core Test shall provide a memory protection unit test (MPU). This is valid even if a Memory Management Unit (MMU) executes MPU functionality. ⌋()

[CorTst014]
⌈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. ⌋()

[CorTst137]
⌈ Each Core Test Interval shall have an Identifier, which shall be incremented by each start of a new test interval in background mode. ⌋(BSW14133)

[CorTst144]
⌈ Core Test module shall provide test execution services in background and foreground mode. ⌋()

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

[CorTst153]「

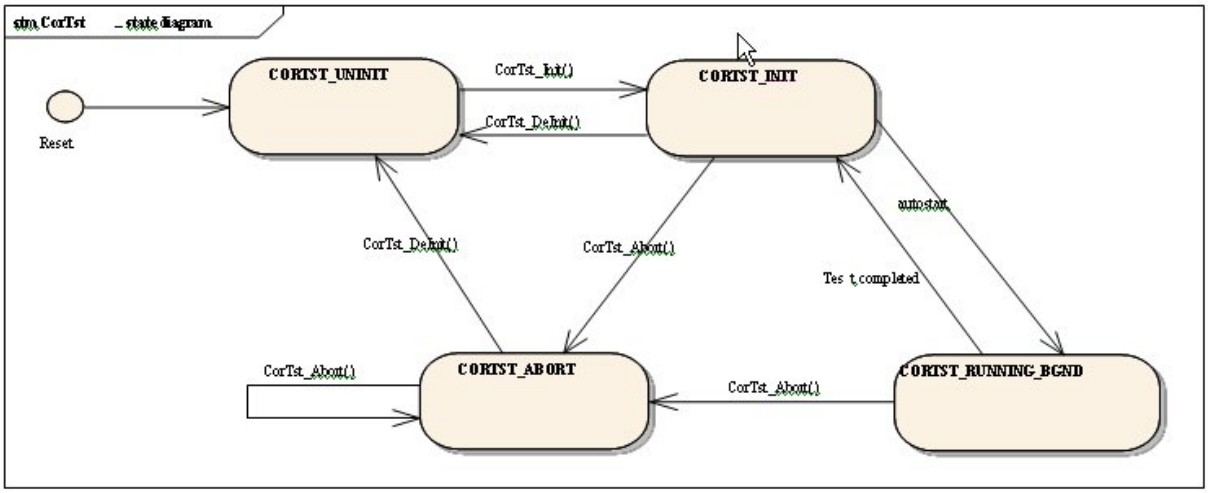


Figure 2 - State Diagram 」()

[CorTst145]「

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

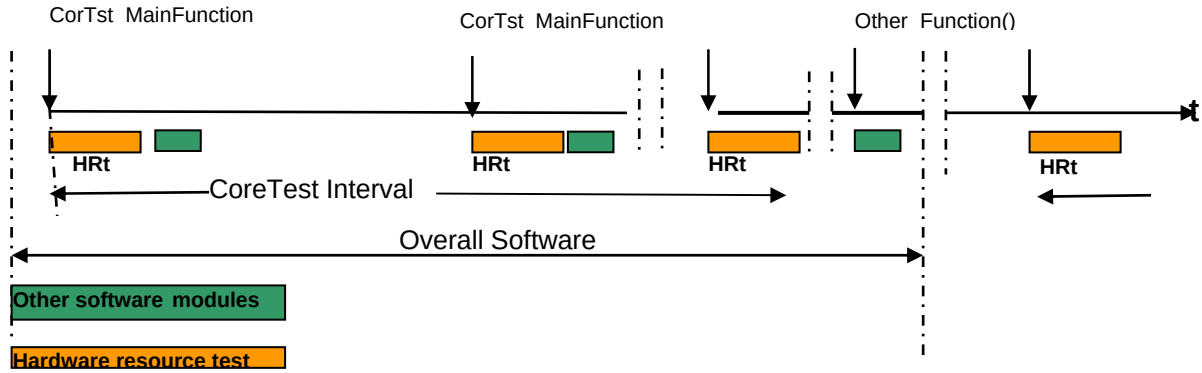
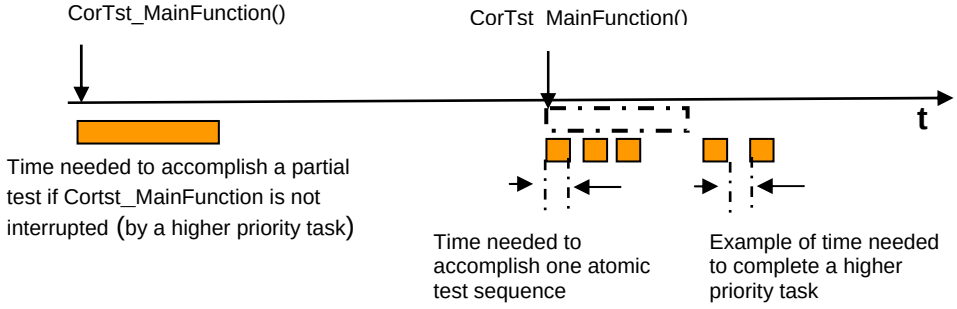


Figure 3 – Backgound Test: Scheduling of Core Test (CorTst)

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

[CorTst015]

「Development error values are of type uint8.」(BSW00337)

[CorTst016]

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

ID:	Type of error	Relevance	Related error code	Value [hex]
CorTst169	API service called with wrong parameter range	Development	CORTST_E_PARAM_INVALID	0x11
CorTst170	API called without Core Test initialization	Development	CORTST_E_UNINIT	0x20
CorTst172	API service CorTst_Init() called again without a CorTst_DeInit() in-between	Development	CORTST_E_ALREADY_INITIALIZED	0x23
CorTst180	API service called with a NULL pointer for CorTst_GetVersionInfo() and CorTst_GetCurrentStatus()	Development	CORTST_E_PARAM_POINTER	0x24
CorTst181	A particular API is called in an unexpected state	Development	CORTST_E_STATUS_FAILURE	0x01
CorTst173	Core failure during tests.	Production	CORTST_E_CORE_FAILURE	Assigned externally by he DEM

」(BSW00337, BSW00385, BSW00327, BSW14117)

7.3 Error detection

[CorTst017]

⌈ If the CORTST_DEV_ERROR_DETECT switch is enabled, development error checking is enabled. Development errors are immediately reported to the calling service during call of an API without executing the intended API functionality. ⌋(BSW00338, BSW00369)

[CorTst018]

⌈If the CORTST_DEV_ERROR_DETECT flag is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter Error classification and chapter API specification. ⌋(BSW00406)

[CorTst019]

⌈ Detection of production errors cannot be switched off. ⌋(BSW00369)

7.4 Error notification

[CorTst020]

⌈ Detected development errors shall be reported to the `Det_ReportError` service of the Development Error Tracer (DET) if the pre-processor switch `CorTstDevErrorDetect` is set (see chapter 10). ⌋()

[CorTst021]

⌈ Production error shall be reported to the Diagnostic Event Manager (DEM) via the Dem_ReportErrorStatus API, 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_ReportErrorStatus API shall be documented by the implementer. ⌋()

7.5 Version Check

[CorTst022]

⌈ The Core Test Module shall avoid the integration of incompatible files by the following pre-processor checks:

For included (external) header files:

- <MODULENAME>_AR_RELEASE_MAJOR_VERSION
- <MODULENAME>_AR_ RELEASE_MINOR_VERSION

shall be verified. ⌋()

Where <MODULENAME> is the module abbreviation of the other (external) modules

which provide header files included by the Core Test module.
If the values are not identical to the values expected by the Core Test Module, an error shall be reported.

7.6 Debugging Support

The following requirements deal with the definition of variables and the description of debug information.

[CorTst0174] ⌈ Each variable that shall be accessible by AUTOSAR Debugging, shall be defined as global variable. ⌋()

[CorTst0175] ⌈ type definitions of variables which shall be debugged, shall be accessible by the header file CorTst.h. ⌋()

[CorTst146] ⌈ The declaration of variables in the header file shall allow to calculate the size of the variables by C-"sizeof". ⌋()

[CorTst147] ⌈ Variables available for debugging shall be described in the respective Basic Software Module Description ⌋()

[CorTst148] ⌈ The state described in CorTst039 shall be available for debugging purposes. ⌋()

7.7 General Requirements

[CorTst023]
⌈ 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. ⌋()

[CorTst024]
⌈ The Core Test implementer shall give an indication on the fault coverage achievements of a Core Test implementation. ⌋()

[CorTst026]

「 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. 」(BSW14130)

8 API specification

8.1 Imported types

This chapter lists all types included from other BSW modules.

[CorTst027]「

Module	*Imported Type*
Dem	Dem_EventIdType
	Dem_EventStatusType
Std_Types	Std_ReturnType
	Std_VersionInfoType

」(BSW00304)

8.2 Type definitions

8.2.1 CorTst_CsumSignatureType

[CorTst037]「

Name:	`CorTst_CsumSignatureType`		
Type:	`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.		

」(BSW00331)

8.2.2 CorTst_CsumSignatureBgndType

[CorTst0176]「

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

」()

8.2.3 CorTst_ErrOkType

[CorTst038]「

Name:	`CorTst_ErrOkType`		
Type:	`Structure`		
Element:	`uint8, uint16, uint32`	`0..<CorTstTestIntervalIdEndValue>`	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)
Description:	This is the type of the Core Test test return if a status is retuned from API to the caller of the API.		

」(BSW00331)

[CorTst138]
「 For the type `CorTst_ErrOkType`, the enumeration value `CORTST_E_NOT_TESTED` shall be the default value after a reset. This enumeration value shall have the numeric value 0. `CorTstTestIntervalId` shall have value zero per default. 」()

8.2.4 CorTst_StateType

[CorTst039]「

Name:	`CorTst_StateType`	
Type:	`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().	

」(BSW00331)

8.2.5 CorTst_TestIdFgndType

[CorTst160]「

Name:	`CorTst_TestIdFgndType`		
Type:	`uint8, uint16, uint32`		
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)).

」()

8.3 Function definitions

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

8.3.1 CorTst_Init

[CorTst040]「

Service name:	CorTst_Init

Syntax:	`void CorTst_Init(`
	`  void`
	`)`

Service ID[hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Service for initialization and change of state of the Core Test

」(BSW101, BSW00406, BSW00358, BSW00414)

[CorTst041]
「 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. 」()

[CorTst0179]
「 The function `CorTst_Init` shall only initialize the configured resources and shall not touch resources that are not configured in the configuration file. 」()

[CorTst042]
「 Execution state will be changed to CORTST_INIT if the driver is called while in state CORTST_UNINIT. 」()

[CorTst0178]
「 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. 」()

[CorTst044]

「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 Development Error Tracer (if development error detection is enabled). 」()

8.3.2 CorTst_DeInit

[CorTst045] 「

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

」()

[CorTst046]
「 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). 」(BSW00336)

[CorTst047]
「 If in state CORTST_INIT: The state shall be changed from CORTST_INIT to CORTST_UNINIT state. 」()

[CorTst136]
「 If in state CORTST_ABORT: The state shall be changed from CORTST_ABORT to CORTST_UNINIT state. 」()

[CorTst149]
「 If the DET is enabled and the status of the CORE Test module is `CORTST_RUNNING_BGND`, the function `CortTst_DeInit` shall report the error value `CORTST_E_STATUS_FAILURE` to the DET, and then immediately return. 」()

8.3.3 CorTst_Abort

[CorTst048] ⌈

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

⌋(BSW14126)

[CorTst049]
⌈ If the current state is CORTST_INIT the state shall be changed from CORTST_INIT to CORTST_ABORT state. ⌋()

[CorTst105]
⌈ If the current state is CORTST_RUNNING_BGND the state shall be changed from CORTST_RUNNING_BGND to CORTST_ABORT state. ⌋()

[CorTst050]
⌈ 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. ⌋()

[CorTst051]
⌈ 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. ⌋()

[CorTst052]
⌈ A call to CorTst_Abort while already being in state CORTST_ABORT does not change the state. ⌋()

[CorTst152]
⌈ A call to `CorTst_Abort` shall set the result of function `CorTst_GetCurrentStatus` to return CORTST_E_NOT_TESTED. ⌋()

8.3.4 CorTstGetState

[CorTst053]「

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.	

」(BSW14118)

[CorTst054]
「 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

[CorTst055]「

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	

」(BSW14131)

[CorTst056]
「 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. 」()

[CorTst120]

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

8.3.6 CorTstGetSignature

[CorTst057] 「

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.	

」(BSW14115, BSW14116)

[CorTst058]
「 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. 」()

[CorTst121]
「 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

[CorTst060] 「

Service name:	CorTst_GetFgndSignature
Syntax:	`CorTst_CsumSignatureType CorTst_GetFgndSignature( void )`
Service ID[hex]:	0x06
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters	None

(inout):		
Parameters (out):	None	
Return value:	CorTst_CsumSignatureType	Implementation specific
Description:	Service to get signature of the last executed Core Test in foreground mode.	

」(BSW14115, BSW14116)

[CorTst061]
「 The function `CorTst_GetFgndSignature` shall return Core Test result signature type as Core Test result of the last completed test run in foreground mode. 」()

[CorTst122]
「 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

[CorTst064] 「

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.	

」(BSW00357, BSW14112, BSW14113)

[CorTst065]
「 The function `CorTst_Start` is only applicable for Foreground mode Core Test operation. 」()

[CorTst109]
「 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. 」()

[CorTst154]
⌈ In case an error occurs during test, the `CorTst_Start` function shall report the production error `CORTST_E_CORE_FAILURE` to the DEM if the core can still report errors reliably by software. ⌋()

[CorTst161]
⌈ 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. ⌋(BSW00323)

8.3.9 CorTst_GetVersionInfo

[CorTst112] 「

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

」(BSW004, BSW00407, BSW003, BSW00411)

[CorTst113]
「 The function `CorTst_GetVersionInfo` shall return the version information of this module. The version information includes:

- Module Id
- Vendor Id
- Vendor specific version numbers 」()

[CorTst115]
「 If source code for caller and callee of `CorTst_GetVersionInfo` is available, the Core Test module should realize `CorTstGet_VersionInfo` as a macro, defined in the module's header file. 」()

[CorTst118]
「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. 」()

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

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.

Terms and definitions:

Fixed cyclic: Fixed cyclic means that one cycle time is defined at configuration and shall not be changed because functionality is requiring a fixed timing (e.g. filters).

Variable cyclic: Variable cyclic means that the cycle times are defined at configuration but might be mode dependent and therefore vary during runtime.

On pre-condition: On pre-condition means that no cycle time can be defined. The function is called when the conditions are fulfilled. Alternatively, the function may be called cyclically, however the cycle time is assigned dynamically during runtime by other modules.

8.5.1 CorTst_MainFunction

[CorTst067] ⌜

Service name:	CorTst_MainFunction

Syntax:	`void CorTst_MainFunction(`
	`  void`
	`)`

Service ID[hex]:	0x0b
Timing:	VARIABLE_CYCLIC_OR_ON_PRECONDITION
Description:	Cyclically called by scheduler to perform processing of Core Test.

⌟(BSW00433, BSW14112, BSW14114)

[CorTst068]
⌜ The function `CorTst_MainFunction` shall set state to `CORTST_INIT`, if all work within a Core Test interval has been finished. ⌟()

[CorTst069]
⌜ The function `CorTst_MainFunction` shall set state to `CORTST_INIT`, if no work within a Core Test needs to be done. ⌟()

[CorTst070]
⌜ 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`. ⌟()

[CorTst071]

⌜ CorTst_MainFunction shall test all selected core hardware entities as configured in CorTst087_Conf. ⌟()

[CorTst072]
⌜ 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. ⌟()

[CorTst073]
⌜ 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. ⌟()

[CorTst074]
⌜ 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. ⌟()

[CorTst139]
⌜ 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. ⌟(BSW14133)

[CorTst155]
⌜ In case an error occurs during test, the CorTest_MainFunction function shall report the production error CORTST_E_CORE_FAILURE to the DEM if the core can still report errors reliably by software. ⌟()

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.

[CorTst0177]

API function	*Description*
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function.

()

8.6.2 Optional Interfaces

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

[CorTst183]

API function	*Description*
Det_ReportError	Service to report development errors.

(BSW00338, BSW00369, BSW00350)

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

[CorTst076]

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.	

(BSW00359, BSW00360, BSW14119)

[CorTst077]

「 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. 」()

[CorTst140]

「 The call of function `CorTst_TestCompletedNotification` shall be pre compile time configurable by the configuration parameter `CorTstNotificationSupported.` 」()

9 Sequence diagrams

9.1 Initialization

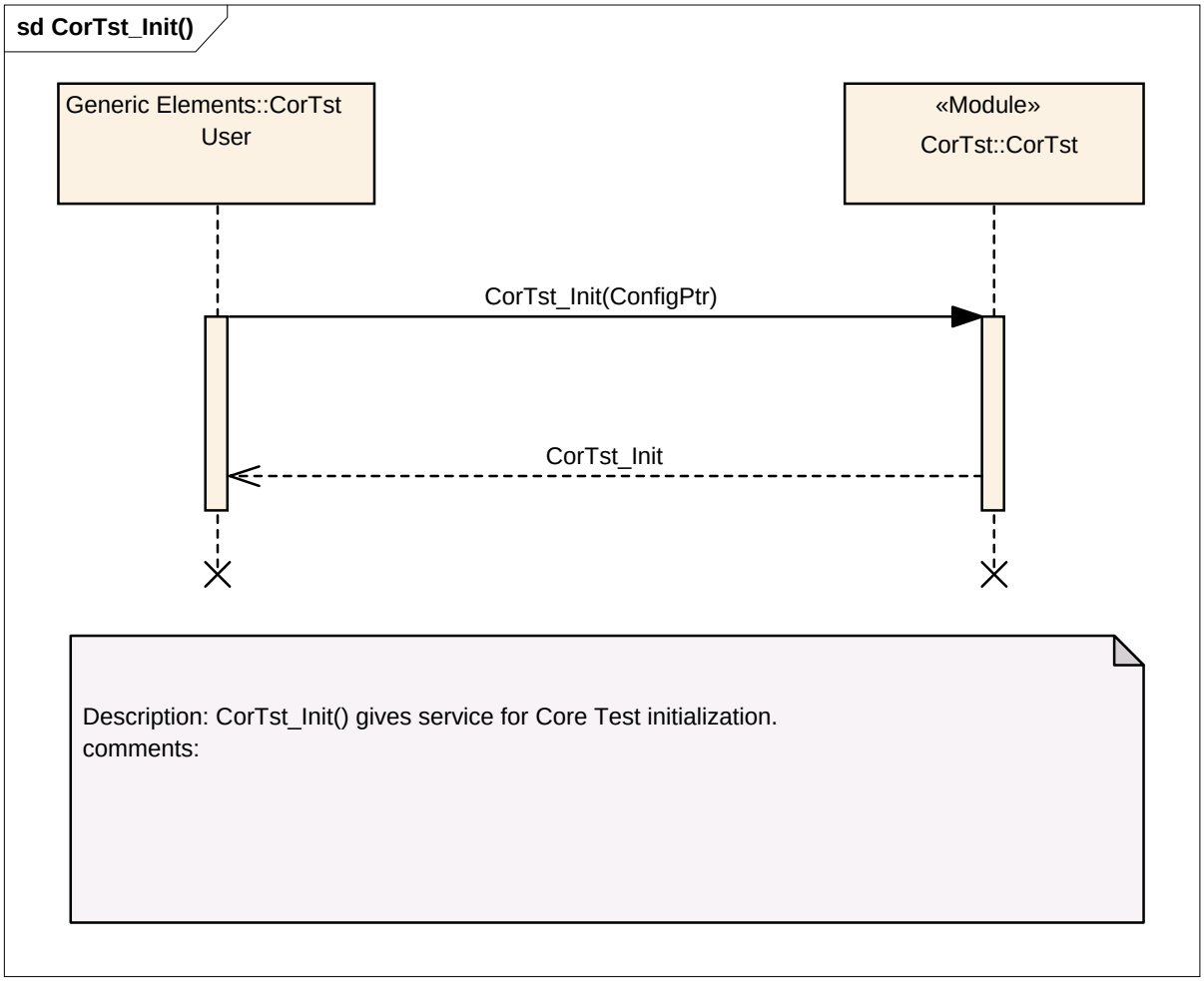


Figure 4 – Core Test Init

9.2 Deinitialization

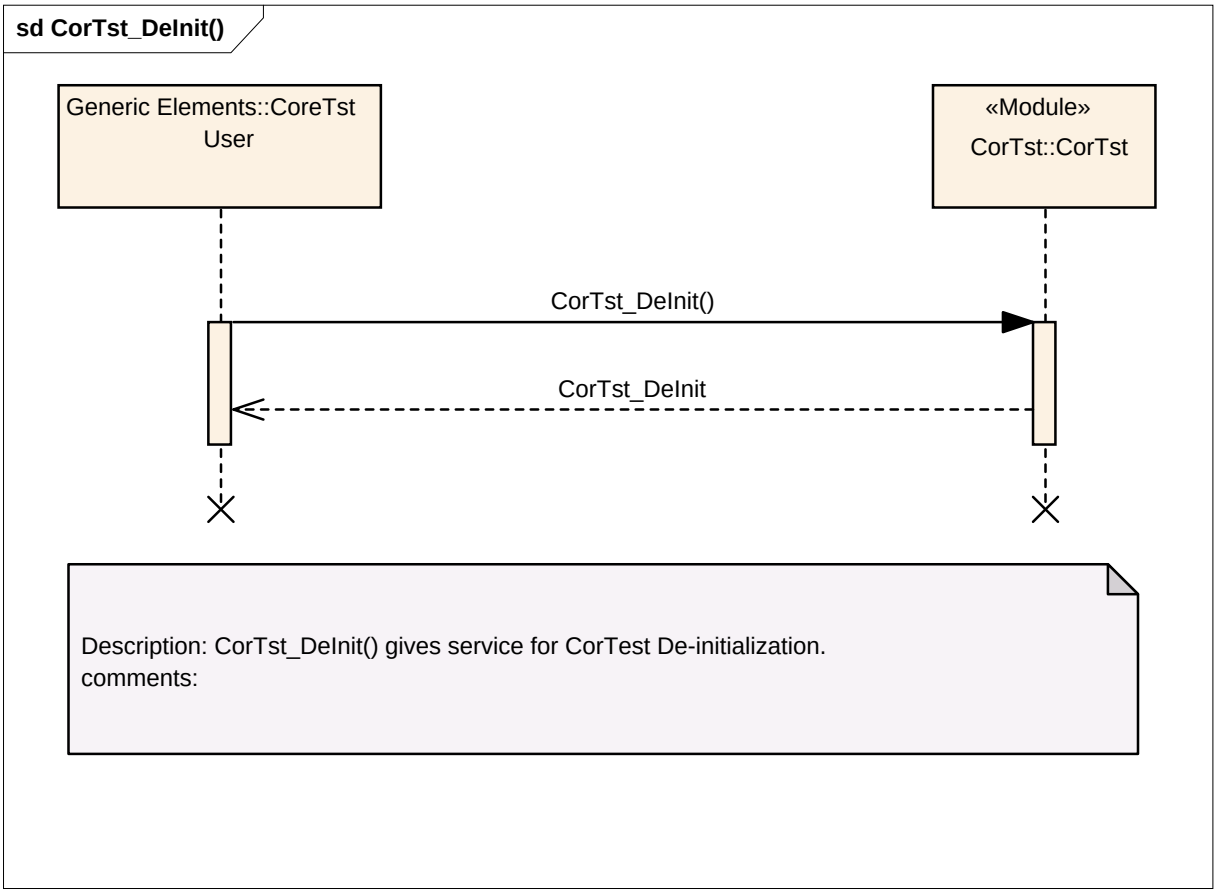


Figure 5 – Core Test De-initialization

9.3 Background Test

9.3.1 Test Result Calculation within Core Test Module

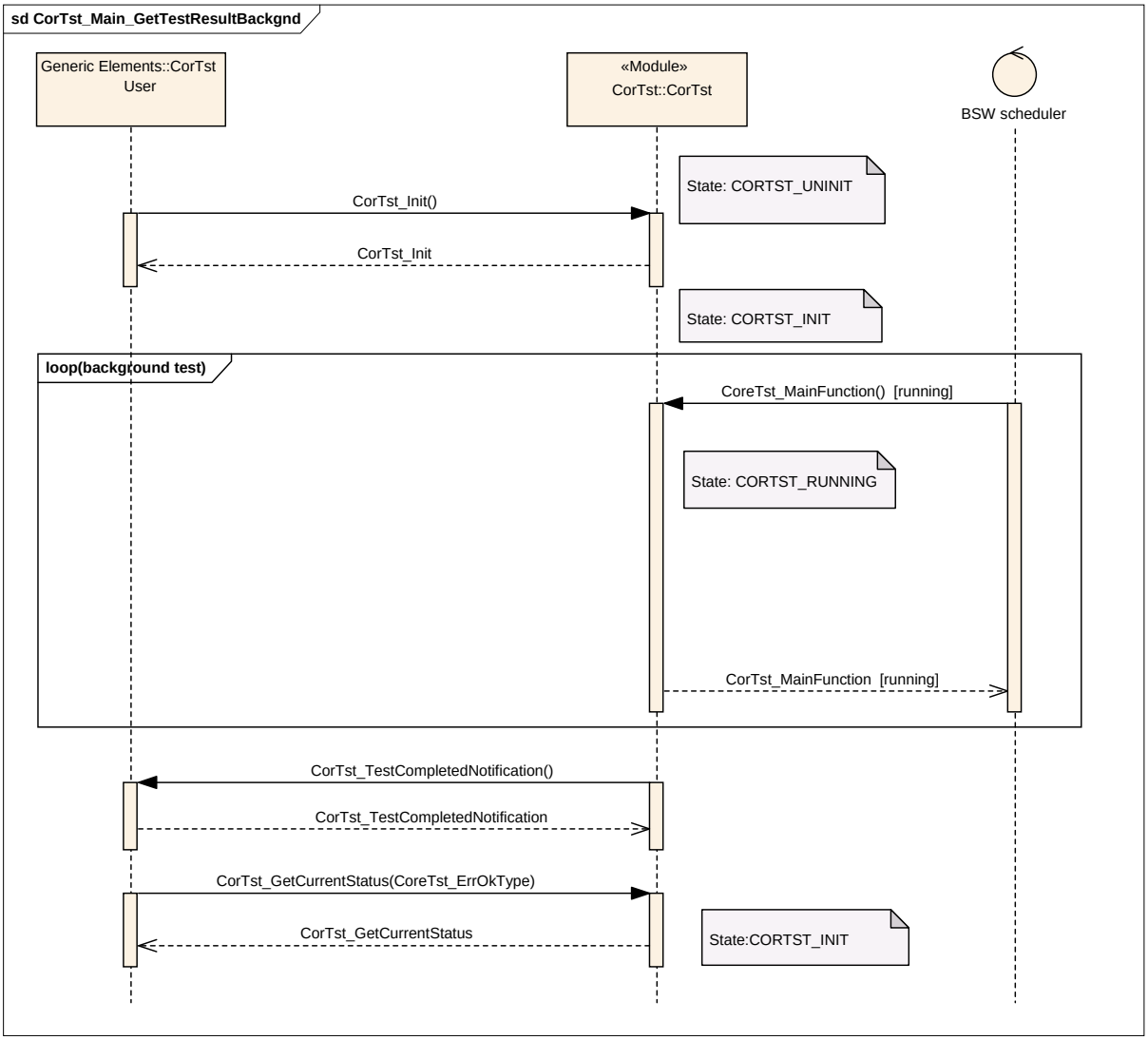


Figure 6 – Result Calculation within Core Test Driver

9.3.2 Core Test Signature provided to Calling Entity

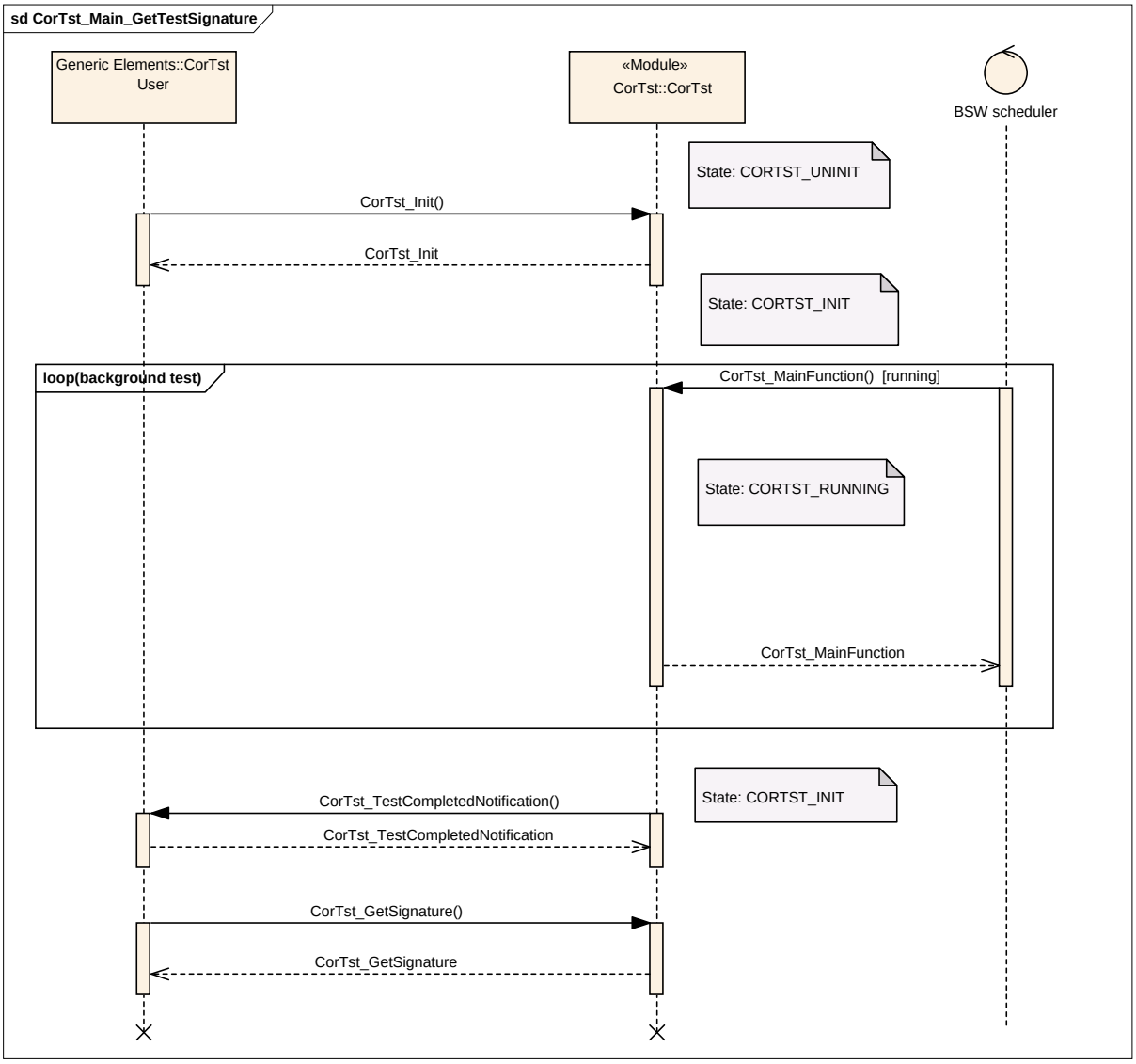


Figure 7 – Result Calculation on Calling Entity

10 Configuration specification

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

10.1 How to read this chapter

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture [1]
- AUTOSAR ECU Configuration Specification [6]
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration Meta model in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Containers

Containers structure the set of configuration parameters. This means:
- *all* configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

10.1.3 Specification template for configuration parameters

The following tables consist of three sections:
- the general section
- the configuration parameter section
- the section of included/referenced containers
-

Pre-compile time	- specifies whether the configuration parameter shall be of configuration class *Pre-compile time* or not

Label	Description
x	The configuration parameter shall be of configuration class *Pre-compile time*.
--	The configuration parameter shall never be of configuration class *Pre-compile time*.

Link time - specifies whether the configuration parameter shall be of configuration class *Link time* or not

Label	Description
x	The configuration parameter shall be of configuration class *Link time*.
--	The configuration parameter shall never be of configuration class *Link time*.

Post Build - specifies whether the configuration parameter shall be of configuration class *Post Build* or not

Label	Description
x	The configuration parameter shall be of configuration class *Post Build* and no specific implementation is required.
L	*Loadable* - the configuration parameter shall be of configuration class *Post Build* and only one configuration parameter set resides in the ECU.
M	*Multiple* - the configuration parameter shall be of configuration class *Post Build* and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class *Post Build*.

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 Variants

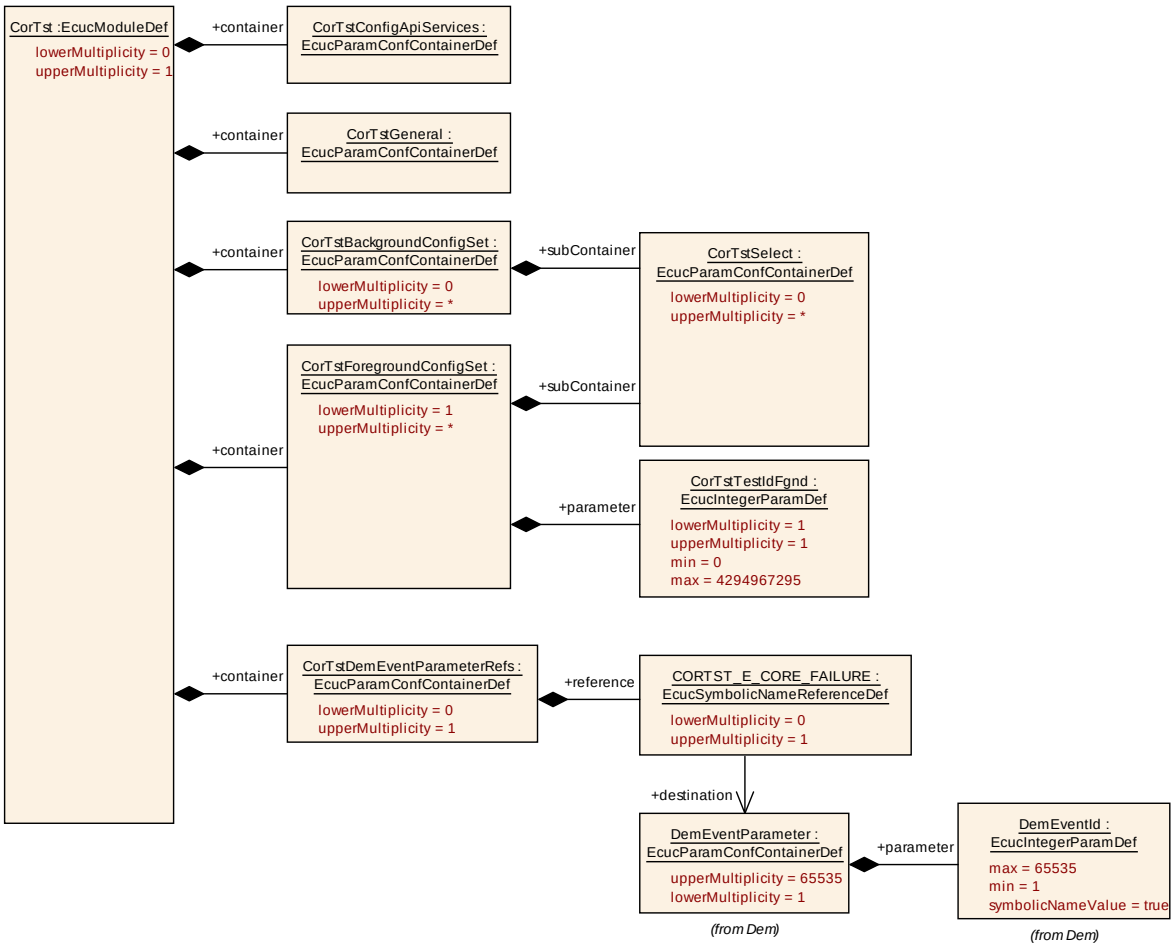
[CorTst078]
⌈ VARIANT-PRE-COMPILE: This variant is limited to pre-compile-configuration parameters only. The intention of this variant is to optimize the parameters configuration for a source code delivery. ⌋()

[CorTst079]
⌈ VARIANT-LINK-TIME: This variant allows a mix of pre-compile time-, link time-configuration parameters. The intention of this variant is to optimize the parameters configuration for an object code delivery. ⌋()

CorTst

SWS Item	**CorTst125_Conf :**
Module Name	*CorTst*
Module Description	Configuration of the CorTst module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CorTstBackgroundConfigSet	0..*	Multiple Configuration Set Container, defines background mode.
CorTstConfigApiServices	1	--
CorTstDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the 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

SWS Item	**CorTst081_Conf :**
Container Name	CorTstGeneral{CORTSTMODULECONFIGURATION}
Description	--
Configuration Parameters	

SWS Item	**CorTst082_Conf :**		
Name	CorTstDevErrorDetect {CORTST_DEV_ERROR_DETECT}		
Description	Switch for enabling the development error detection.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**CorTst159_Conf :**		
Name	CorTstFgndTestNumber {CORTST_FGND_TEST_NUMBER}		
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants

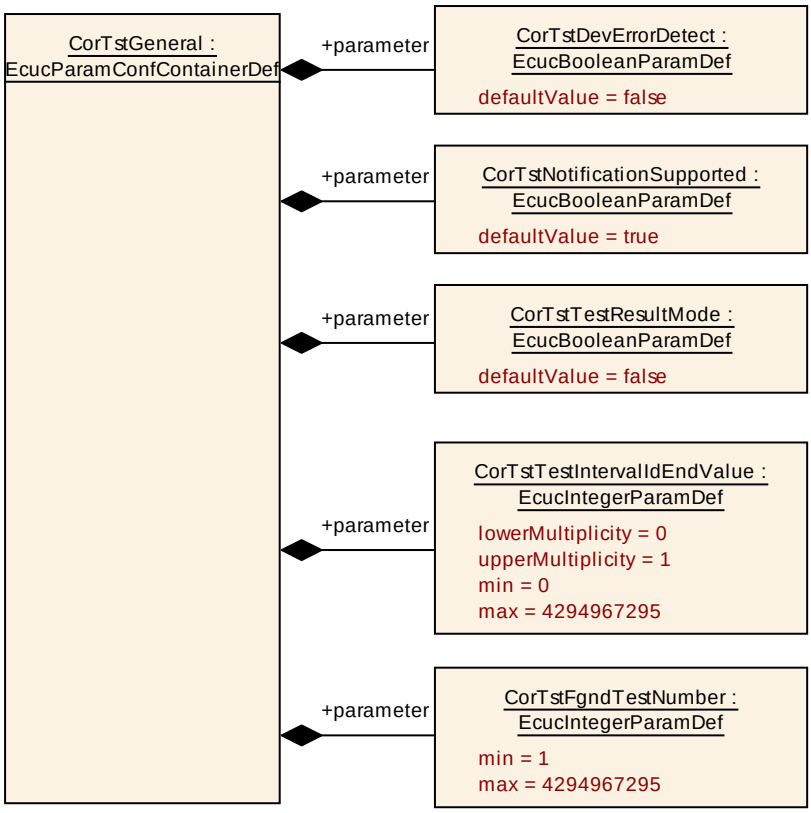
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**CorTst083_Conf :**		
Name	CorTstNotificationSupported {CORTST_NOTIFICATION_SUPPORTED}		
Description	Switch to indicate that the notification is supported.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**CorTst143_Conf :**		
Name	CorTstTestIntervalIdEndValue {CORTST_TEST_INTERVAL_ID_END_VALUE}		
Description	Defines the end value of the Test Interval Id.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**CorTst086_Conf :**		
Name	CorTstTestResultMode {CORTST_TEST_RESULT_MODE}		
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		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

No Included Containers



10.2.3 CorTstSelect

SWS Item	**CorTst089_Conf :**
Container Name	CorTstSelect{CORTST_SELECT}
Description	This container specifies configuration parameters to select individual tests for foreground mode and background mode. The availability is hardware and implementation specific.
Configuration Parameters	

SWS Item	**CorTst130_Conf :**		
Name	CorTstAddress {CORTST_ADDRESS}		
Description	Enable/Disables core address test.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**CorTst129_Conf :**		
Name	CorTstAlu {CORTST_ALU}		
Description	Enable/Disables core ALU test.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	CorTst133_Conf :		
Name	CorTstCache {CORTST_CACHE}		
Description	Enable/Disables core cache test.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: module		

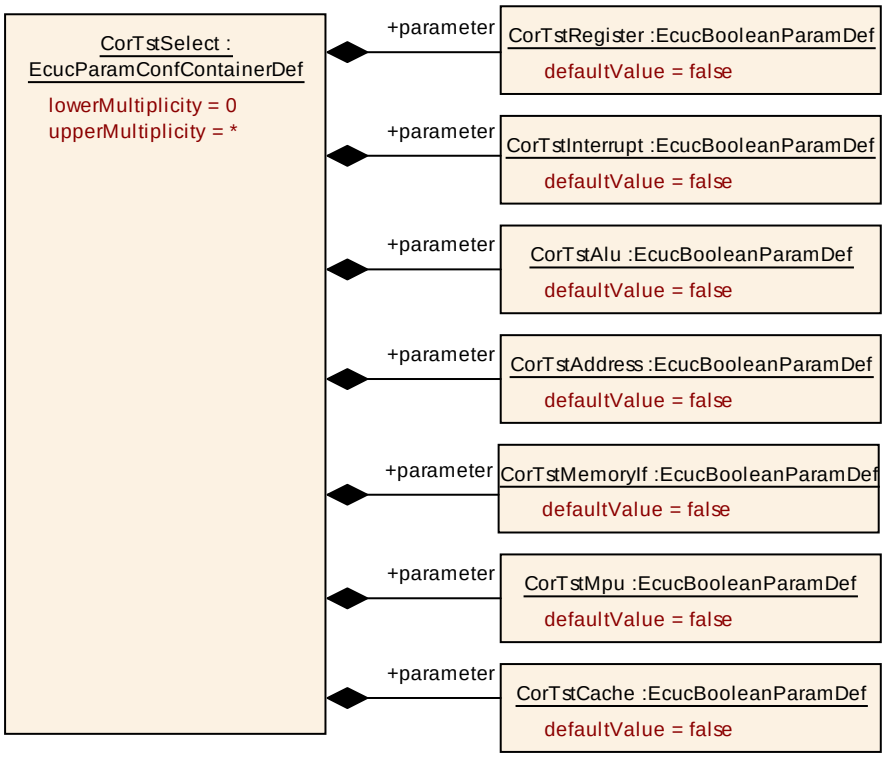
SWS Item	CorTst128_Conf :		
Name	CorTstInterrupt {CORTST_INTERRUPT}		
Description	Enable/Disables core interrupt test		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	CorTst131_Conf :		
Name	CorTstMemoryIf {CORTST_MEMORYIF}		
Description	Enable/Disables core memory interface test		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	CorTst132_Conf :		
Name	CorTstMpu {CORTST_MPU}		
Description	Enable/Disables core MPU test		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	CorTst127_Conf :		
Name	CorTstRegister {CORTST_REGISTER}		
Description	Enable/Disables core register test		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: module		

No Included Containers



10.2.4 CorTstBackgroundConfigSet

SWS Item	**CorTst087_Conf :**
Container Name	CorTstBackgroundConfigSet
Description	Multiple Configuration Set Container, defines background mode.
Configuration Parameters	

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.

CorTstForegroundConfigSet

SWS Item	**CorTst088_Conf :**
Container Name	CorTstForegroundConfigSet
Description	Multiple Configuration Set Container , defines foreground mode.
Configuration Parameters	

SWS Item	**CorTst158_Conf :**
Name	CorTstTestIdFgnd {CORTST_TEST_ID_FGND}
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

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 CorTstConfigApiServices

SWS Item	**CorTst092_Conf :**
Container Name	CorTstConfigApiServices
Description	--
Configuration Parameters	

SWS Item	**CorTst094_Conf :**		
Name	CorTstAbortApi {CORTST_ABORT_API}		
Description	Adds / removes the service CorTst_Abort() from the code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**CorTst104_Conf :**		
Name	CorTstGetCurrentStatus {CORTST_GET_CURRENT_STATUS_API}		
Description	Adds / removes the service CorTst_GetCurrentStatus() from the code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**CorTst103_Conf :**		
Name	CorTstGetFgndSignature {CORTST_GET_FGND_SIGNATURE_API}		
Description	Adds / removes the service CorTst_GetFgndSignature() from the code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**CorTst097_Conf :**

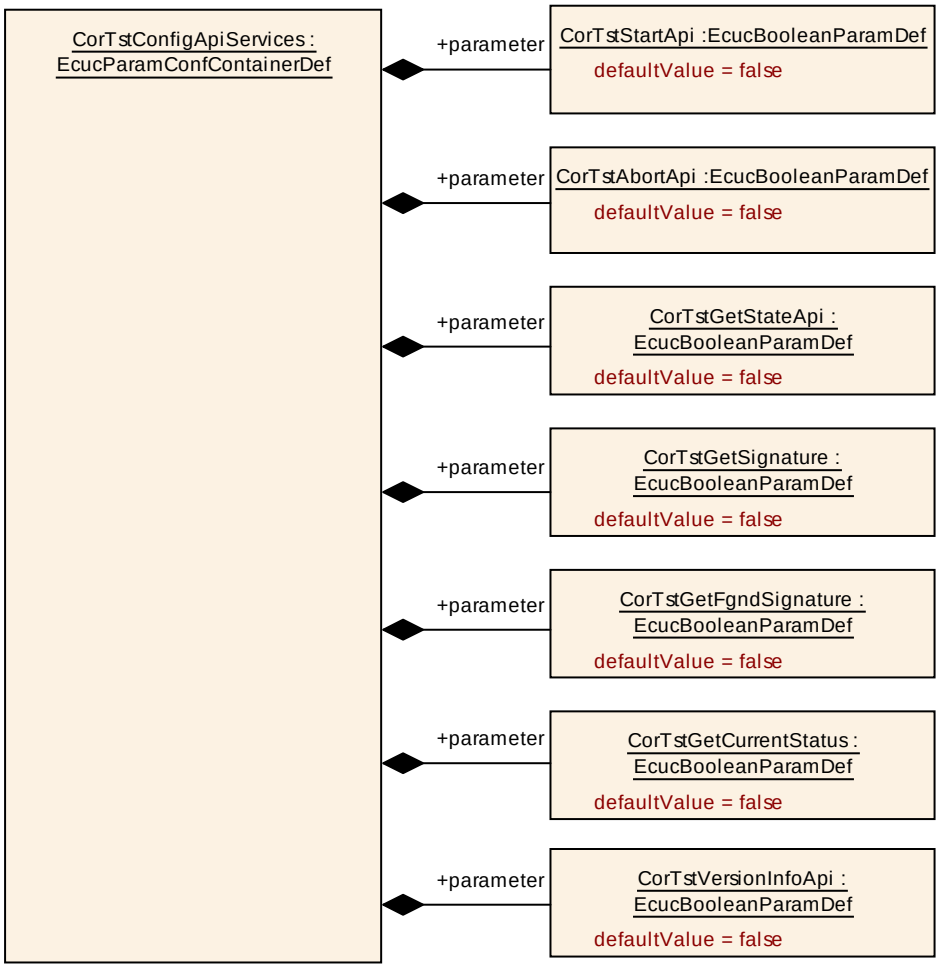
Name	CorTstGetSignature {CORTST_GET_SIGNATURE_API}		
Description	Adds / removes the service CorTst_GetSignature() from the code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**CorTst096_Conf :**		
Name	CorTstGetStateApi {CORTST_GET_STATE_API}		
Description	Adds / removes the service CorTst_GetState() from the code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**CorTst093_Conf :**		
Name	CorTstStartApi {CORTST_START_API}		
Description	Adds / removes the service CorTst_Start() from the code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**CorTst098_Conf :**		
Name	CorTstVersionInfoApi {CORTST_VERSION_INFO_API}		
Description	Adds / removes the service CorTst_GetVersionInfo() from the code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: module		

No Included Containers



10.2.6 CorTstDemEventParameterRefs

SWS Item	**CorTst156_Conf :**
Container Name	CorTstDemEventParameterRefs
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.
Configuration Parameters	

SWS Item	**CorTst157_Conf :**		
Name	CORTST_E_CORE_FAILURE {CORTST_E_CORE_FAILURE}		
Description	Reference to the DemEventParameter which shall be issued when the error "CORE failure" has occured.		
Multiplicity	0..1		
Type	Reference to [DemEventParameter]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: Dem		

No Included Containers

10.3 Published Information

[CorTst182] ⌈ The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules [3] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [1]. ⌋()

Additional module-specific published parameters are listed below if applicable.

11 Not applicable requirements

[CorTst999]「These requirements are not applicable to this specification.」
(BSW167, BSW168, BSW00339, BSW00344, BSW00375, BSW00383, BSW00386, BSW00398, BSW00399, BSW00404, BSW00405, BSW00409, BSW00416, BSW00417, BSW00422, BSW00423, BSW00424, BSW00425, BSW00426, BSW00428, BSW00429, BSW00431, BSW00432, BSW00434, BSW00437, BSW00438, BSW005, BSW006, BSW009, BSW010, BSW161, BSW162, BSW170, BSW171, BSW172, BSW00301, BSW00302, BSW00306, BSW00308, BSW00309, BSW00310, BSW00312, BSW00314, BSW00318, BSW00321, BSW00325, BSW00328, BSW00329, BSW00330, BSW00333, BSW00334, BSW00341, BSW00346, BSW00355, BSW00370, BSW00371, BSW00374, BSW00378, BSW00379, BSW00413, BSW00436, BSW14125, BSW14124)