Document Title	Specification of RAM Test
Document Category	Software specification (SWS)
Document UID	76
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History

Date	Release	Changed by	Description
2026-11-26	R26-11	AUTOSAR Release Management	• ECUC items; restructured • ECUC items; Loadable and selectable post build values introduced • ECUC items; Pre-compile option removed in case of several allowed configuration classes • Extendended production errors changed to production errors • API function RamTst GetVersionInfo removed
2025-11-27	R25-11	AUTOSAR Release Management	• Editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	• Editorial changes
2023-11-23	R23-11	AUTOSAR Release Management	• Editorial changes
2022-11-24	R22-11	AUTOSAR Release Management	• Changed [SWS_RamTst_00999] to [SWS_RamTst_NA_00999]
2021-11-25	R21-11	AUTOSAR Release Management	• Clarificaton on APIs defined as Synchronous / Asynchronous(RamTst_ErrorNotification, RamTst_TestCompletedNotification)
2020-11-30	R20-11	AUTOSAR Release Management	• Updated the structure and tables of the error sections • Editorial changes

△

2019-11-28	R19-11	AUTOSAR Release Management	• MCALMulticoreDistribution(CONC_639) • Production errors updated • Editorial changes • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• MCALMulticoreDistribution(CONC_639) as DRAFT • Header File Cleanup • Minor corrections; For details please refer to the ChangeDocumentation
2017-12-08	4.3.1	AUTOSAR Release Management	• Updated traceability • minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2016-11-30	4.3.0	AUTOSAR Release Management	• Removed subsection 7.5 Debugging • Renamed "RamTstGetVersionInfoApi to "RamTstVersionInfoApi" • Removed [SWS_RamTst_00167] and [SWS_RamTst_00168] • Added line "Supported Config Variants" to the table of the module definition in 10.2.1 • Added sections Runtime Errors and Transient Faults • Renamed "[RS_SPAL_12448]" to "[SRS_SPAL_12448]" • Removed BSW00434, BSW00443, BSW00444, [SRS_BSW_00370], [SRS_BSW_00435], [SRS_BSW_00436]
2015-07-31	4.2.2	AUTOSAR Release Management	• Updated Pass/Fail Criterias for Extended Production Errors • Debugging support marked as obsolete • Diverse corrections • Editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	• Added Pass/Fail Criterias for Extended Production Errors

▽

△

2014-03-31	4.1.3	AUTOSAR Release Management	• Editorial changes • Updated traceability
2013-10-31	4.1.2	AUTOSAR Release Management	• Removed timing attribute of requirement [SWS_RamTst_00110] • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Alignment to the new SWS_BSWGeneral document • Updated the document for Extended Production Errors • Alignment to official naming in other Autosar documents • Adjustment to ISO 26262: major • Clarification of some requirements
2011-12-22	4.0.3	AUTOSAR Administration	• Clarification of some requirements. • Typos correction. • Added a new requirement for DET error reporting
2010-09-30	3.1.5	AUTOSAR Administration	• Clarification on some configuration parameters • Clarification of some types used in API • Improvement of error reporting
2010-02-02	3.1.4	AUTOSAR Administration	• Foreground tests added • Allow more than one configuration per test algorithm • Further maintenance for R4.0 • Legal disclaimer revised
2009-02-04	3.1.2	AUTOSAR CM	• Updated document to new SWS macros
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2008-02-01	3.0.2	AUTOSAR Administration	• Correction of figures in Chapter 1 and Chapter 9.

▽

△

2007-12-21	3.0.1	AUTOSAR Administration	• RAM test concept documented and included; • Requirements tables updated; • Wording/grammar changes; • Sequence diagram changes; • Generated content corrected/modified. • Document meta information extended • Small layout adaptations made
2007-12-21	3.0.1	AUTOSAR Technical Office	• Fixed bug 18934 (changes only in chapter 8, no changes in chapter 9 and chapter 10)
2007-07-24	2.1.16	AUTOSAR Technical Office	• Tables generated from UML-models, UML-diagrams linked to UML-model, general improvements of requirements in preparation of CT-development. No changes in the technical contents of the specification.
2007-01-24	2.1.15	AUTOSAR Administration	• "Advice for users" revised • "Revision Information" added
2006-11-28	2.1.1	AUTOSAR Administration	• File include structure updated • "Modified Hamming code" test removed • RamTst_Stop() and RamTst_Continue() changed to "asynchronous" • Dem API updated • Configuration description corrected • descriptions optimized • Legal disclaimer revised
2006-05-16	2.0	AUTOSAR Administration	• Initial release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 9
2 Acronyms and Abbreviations 18
3 Related documentation 19
3.1 Input documents & related standards and norms . . . 19
3.2 Related specification . . . 19
4 Constraints and assumptions 20
4.1 Limitations . . . 20
4.2 Full RAM Test . . . 20
4.3 Partial RAM Test . . . 21
4.4 Applicability to car domains . . . 21
5 Dependencies to other modules 22
6 Requirements Tracing 23
7 Functional specification 25
7.1 Requirements . . . 25
7.2 Error Classification . . . 26
7.2.1 Development Errors . . . 27
7.2.2 Runtime Errors . . . 27
7.2.3 Production Errors . . . 27
7.3 General Test Behavior . . . 29
8 API specification 32
8.1 Imported types . . . 32
8.2 Type definitions . . . 32
8.2.1 RamTst_ConfigType . . . 32
8.2.2 RamTst_ExecutionStatusType . . . 32
8.2.3 RamTst_TestResultType . . . 33
8.2.4 RamTst_AlgParamsIdType . . . 33
8.2.5 RamTst_AlgorithmType . . . 34
8.2.6 RamTst_NumberOfTestedCellsType . . . 35
8.2.7 RamTst_NumberOfBlocksType . . . 35
8.3 Function definitions . . . 35
8.3.1 RamTst_Init . . . 35
8.3.2 RamTst_DeInit . . . 36
8.3.3 RamTst_Stop . . . 37
8.3.4 RamTst_Allow . . . 38
8.3.5 RamTst_Suspend . . . 39
8.3.6 RamTst_Resume . . . 39
8.3.7 RamTst_GetExecutionStatus . . . 40
8.3.8 RamTst_GetTestResult . . . 41

8.3.9 RamTst_GetTestResultPerBlock . . . 42
8.3.10 RamTst_GetAlgParams . . . 43
8.3.11 RamTst_GetTestAlgorithm . . . 43
8.3.12 RamTst_GetNumberOfTestedCells . . . 44
8.3.13 RamTst _SelectAlgParams . . . 44
8.3.14 RamTst_ChangeNumberOfTestedCells . . . 46
8.3.15 RamTst_RunFullTest . . . 47
8.3.16 RamTst_RunPartialTest . . . 48
8.4 Scheduled functions . . . 50
8.4.1 RamTst_MainFunction . . . 50
8.5 Expected interfaces . . . 52
8.5.1 Mandatory interfaces . . . 52
8.5.2 Optional interfaces . . . 52
8.5.3 Callout Interfaces . . . 53
8.5.4 Configurable interfaces . . . 53
8.5.4.1 RamTst_TestCompletedNotification . . . 53
8.5.4.2 RamTst_ErrorNotification . . . 54
9 Sequence diagrams 56
9.1 RamTst_MainFunction (Examples) . . . 56
9.2 RamTst_ChangeNumberOfTestedCells . . . 59
9.3 RamTst_SelectAlgParams . . . 59
9.4 RamTst_GetAlgParams . . . 59
9.5 RamTst_GetExecutionStatus . . . 60
9.6 RamTst_GetTestResult . . . 60
9.7 RamTst_GetTestResultPerBlock . . . 60
9.8 RamTst_GetTestAlgorithm . . . 61
9.9 RamTst_GetNumberOfTestedCells . . . 61
10 Configuration specification 62
10.1 How to read this chapter . . . 62
10.2 Containers and configuration parameters . . . 62
10.2.1 Variants . . . 63
10.2.2 RamTst . . . 63
10.2.3 RamTstDemEventParameterRefs . . . 64
10.2.4 RamTstCommon . . . 66
10.2.5 RamTstAlgorithms . . . 72
10.2.6 RamTstConfigParams . . . 75
10.2.7 RamTstAlgParams . . . 78
10.2.8 RamTstBlockParams . . . 82
10.3 Published Information . . . 85
10.3.1 RamTstPublishedInformation . . . 85
10.4 Implementation Specific Information and Parameters . . . 86
A Not applicable requirements 88

B Change history of AUTOSAR traceable items 89
B.1 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 89
B.1.1 Added Specification Items in R26-11 . . . 89
B.1.2 Changed Specification Items in R26-11 . . . 89
B.1.3 Deleted Specification Items in R26-11 . . . 92
B.1.4 Added Constraints in R26-11 . . . 92
B.1.5 Changed Constraints in R26-11 . . . 92
B.1.6 Deleted Constraints in R26-11 . . . 92
B.2 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 92
B.2.1 Added Specification Items in R25-11 . . . 92
B.2.2 Changed Specification Items in R25-11 . . . 92
B.2.3 Deleted Specification Items in R25-11 . . . 93
B.2.4 Added Constraints in R25-11 . . . 93
B.2.5 Changed Constraints in R25-11 . . . 93
B.2.6 Deleted Constraints in R25-11 . . . 93

1 Introduction and functional overview

This document specifies the functionality, API and configuration of the AUTOSAR Basic Software module "RAM Test".

The RAM Test is a test of the physical health of the RAM cells. It is not intended to test the contents of the RAM. RAM used for registers is also tested.

Within this document, a RAM cell is understood as the unit of memory, which can be individually addressed by the processor. Thus the cell size in bits is for example 16 for a 16-bit processor.

Different algorithms exist to test RAM. They target different sets of fault models, achieve different coverages, result in different runtimes and are either destructive or non-destructive. Coverage also depends on the underlying physical RAM architecture. ISO 26262 only establishes a distinction between three basic coverage levels Low (60%), Medium (90%) and High (99%) [1]. This basic distinction is also used in the AUTOSAR specification.

An ECU safety analysis must be performed to determine which RAM Test diagnostic coverage rate (Low, Medium or High) is required. Appropriate RAM Test algorithms and further configuration parameters are then selected at compile time. At run time, the application software may choose between the compiled algorithms (and between further parameters).

A RAM Test may be called synchronously by the test environment (hereafter called "foreground test") or may be called in a cyclic manner by an OS task or other cyclic calling method (hereafter called "background test"). The test environment may select test parameters, start and stop the test, and get status reports. Development errors are reported to the Default Error Tracer (DET) and production errors are reported to the Diagnostic Event Manager (DEM).

The RamTst module consists of a `RamTst_MainFunction` for background testing, the APIs for foreground testing, several configuration and status APIs (Application Programming Interface), and several configuration containers.

TEST FUNCTION APIs	*DEFINITION*
`RamTst_Init`	Prepare resources for testing as necessary. Initialize the test execution state as necessary. Proceed to "test stopped" state after initialization is complete.
`RamTst_DeInit`	Reset all used registers to reset values, and release all used resources.
`RamTst_Allow`	Permit the `RamTst_MainFunction` to perform testing at its next scheduled call.
`RamTst_Stop`	Prohibit the `RamTst_MainFunction` from performing tests at its next scheduled call. When `RamTst_Stop` is called, testing stops after the current atomic sequence. Test status is retained, but test parameters (block number, loop count, etc.) are discarded.
`RamTst_Suspend`	Temporarily prohibit the `RamTst_MainFunction` from performing tests at its next scheduled call. When `RamTst_Suspend` is called, testing stops after the current atomic sequence. Test status and test parameters are retained.
`RamTst_Resume`	Permits the `RamTst_MainFunction` to continue testing at the point where it was suspended, at its next scheduled call. Testing continues according to the saved test parameters.

△

TEST FUNCTION APIs	DEFINITION
RamTst_RunFull-Test	Test the entire RAM space without interruption. RamTst_Stop must be called prior to calling this API.
RamTst_Run-PartialTest	Test the portion of the RAM defined by the API. RamTst_Stop or RamTst_Suspend must be called prior to calling this API.

TEST PARAMETER AND FEEDBACK APIs
RamTst_GetExecutionStatus
RamTst_GetTestResult
RamTst_GetTestResultPerBlock
RamTst_GetAlgParams
RamTst_GetTestAlgorithm
RamTst_GetNumberOfTestedCells
RamTst_SelectAlgParams
RamTst_ChangeNumberOfTestedCells

RamTst_MainFunction is the scheduled function for background testing.

- For background testing, RamTst_MainFunction is called periodically by a scheduler, and is interruptible. One complete test consists of testing with one algorithm over the memory space defined by the currently selected configuration. This complete test is split up over many scheduled calls.
- For foreground testing, RamTst_RunFullTest() or RamTst_RunPartial-Test() is called once, and is not interruptible by routines which access the tested memory area (this has to be controlled by the test environment). It tests with one algorithm over the memory space (or a subset in case of partial test) defined by the selected configuration.

The state chart below shows the various states of the test execution.

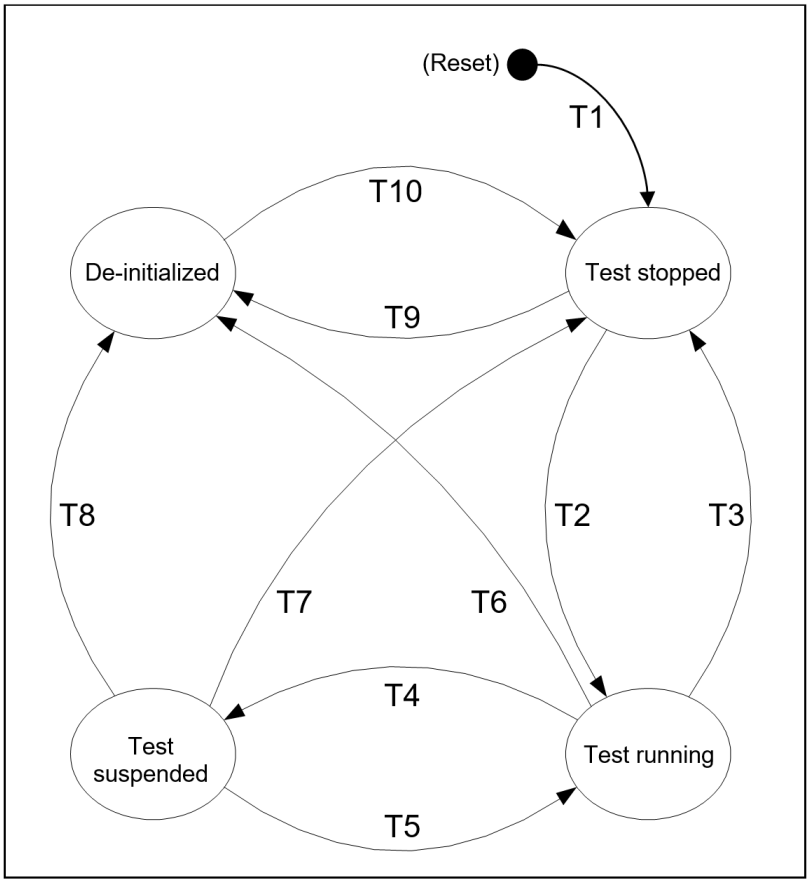


Figure 1.1: Phases of RAM Test module

Event	Event Trigger
T1	API: RamTst_Init
T2	API: RamTst_RunFullTest API: RamTst_RunPartialTest API: RamTst_Allow
T3	API: RamTst_Stop (or end of RamTst_RunFullTest) (or end of RamTst_RunPartialTest)
T4	API: RamTst_Suspend (or end of RamTst_RunPartialTest)
T5	API: RamTst_Resume API: RamTst_RunPartialTest
T6	API: RamTst_DeInit
T7	API: RamTst_Stop
T8	API: RamTst_DeInit
T9	API: RamTst_DeInit
T10	API: RamTst_Init

Note: The state "test running" does not necessarily mean that testing is continuously being performed. For foreground testing, it does mean that the test is directly performed by an API call and RamTst_MainFunction is not scheduled. For background

testing, it only means that RamTst_MainFunction is permitted to test a small portion of the RAM when it is called periodically by the scheduler.

In the actual specification, this state is further divided into "test allowed" and "test running". The state "test allowed" is only used in the initial phase of a background test; for the big picture given in this overview this difference has been neglected.

All APIs and configuration variables are fully defined elsewhere within this document.

The following table shows, which APIs are allowed to be called in each state. For any cell in the table where there is an "N", there should be a corresponding DET error assigned.

API: Application Programming Interface

	API allowable in this State?			
APIs which cause a change of state in the state chart	**Test Stopped**	**Test Running (or Allowed)**	**Test Suspended**	**Test De-initialized**
RamTst_Init	N	N	N	Y
RamTst_RunFullTest	Y	N	N	N
RamTst_RunPartialTest	Y	N	Y	N
RamTst_Suspend [1]	N	Y	N	N
RamTst_Resume	N	N	Y	N
RamTst_Stop	N	Y	Y	N
RamTst_Allow [2]	Y	N	N	N
RamTst_DeInit	Y	Y	Y	N
RamTst_GetExecutionStatus	Y	Y	Y	N
RamTst_GetTestResult	Y	Y	Y	N
RamTst_GetTestResultPerBlock	Y	Y	Y	N
RamTst_GetAlgParams	Y	Y	Y	N
RamTst_GetTestAlgorithm	Y	Y	Y	N
RamTst_GetNumberOfTestedCells	Y	Y	Y	N
RamTst_SelectAlgParams [3]	Y	N	N	N
RamTst_ChangeNumberOfTestedCells [4]	Y	N	N	N

The following figure shows how blocks are configured for an algorithm, and how RamTst_MainFunction then tests the memory cells for each block in a background test.

[1] RamTst_Suspend causes a state change to "test suspended" at the end of the current RamTst_MainFunction atomic sequence if RamTst_MainFunction is actively testing.

[2] RamTst_Allow is called to permit the RamTst_MainFunction to test when called, it does not initiate any test itself.

[3] RamTst_Stop must first be called before selecting another configuration parameter set by RamTst_SelectAlgParams.

[4] RamTst_ChangeNumberOfTestedCells operates at the end of the current RamTst_MainFunction atomic sequence if RamTst_MainFunction is actively testing. For a foreground test, RamTst_ChangeNumberOfTestedCells is not relevant.

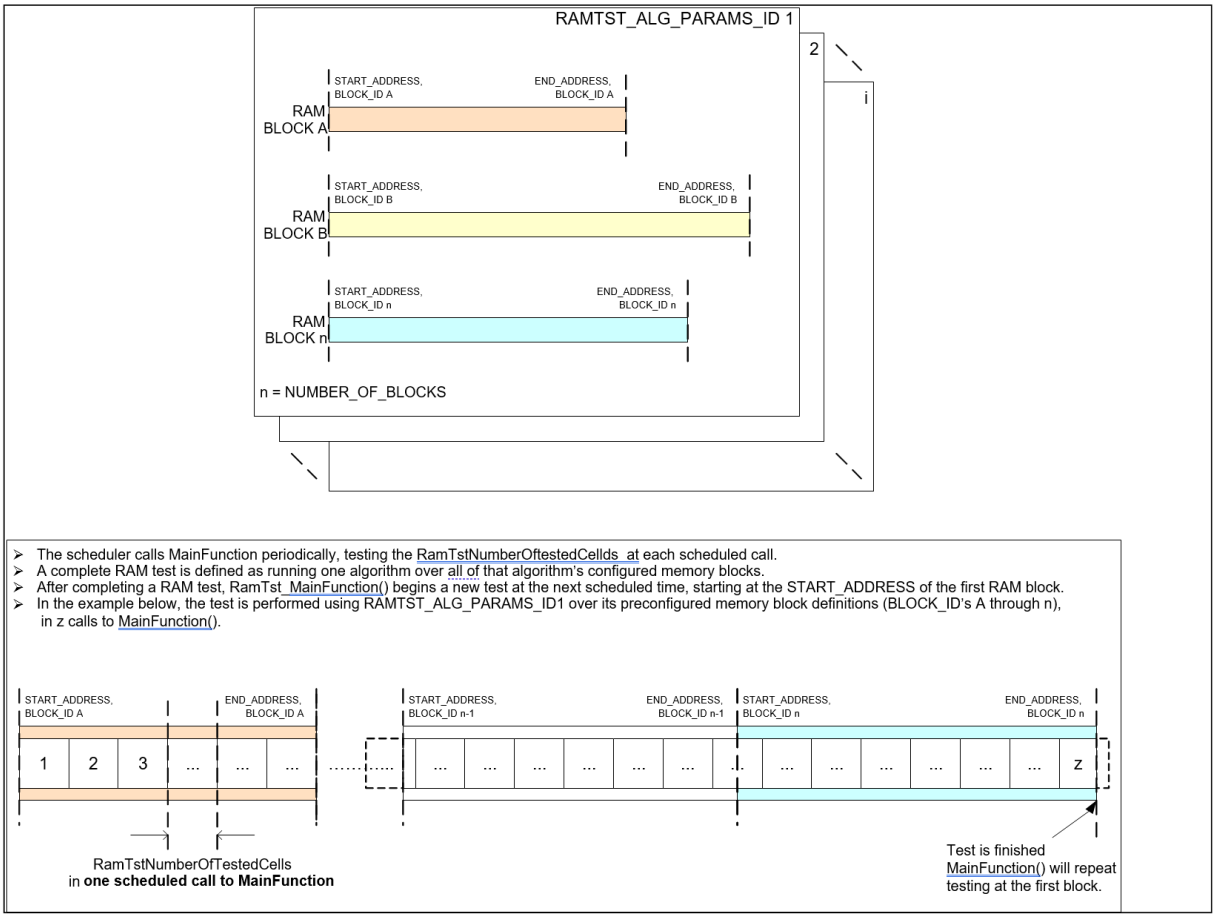


Figure 1.2: Configured memory blocks for background memory test

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

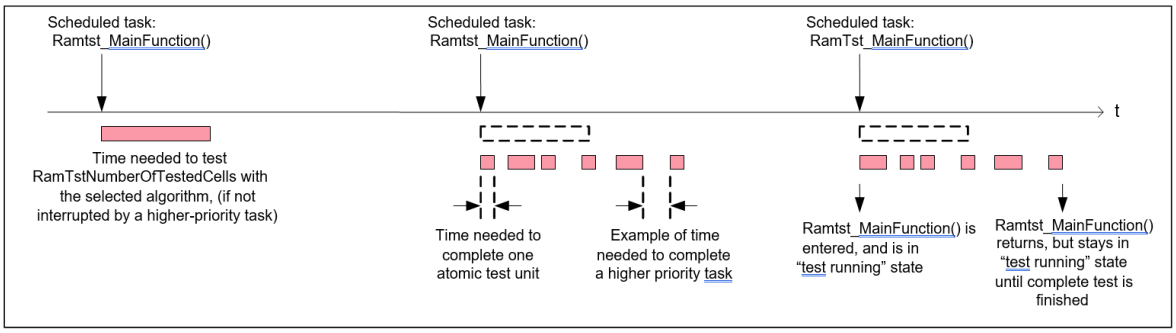


Figure 1.3: Main Function Schedule

`RamTstNumberOfTestedCells`

The `RamTstNumberOfTestedCells` default is set by configuration (pre-compile or link) in the `RamTstAlgParams` container and applies to every block defined within an algorithm, but can be different for each `RamTstAlgParams`, thus can be different

for different algorithms or for different parameter sets for the same algorithm. RamTstNumberOfTestedCells can be changed during runtime using the API RamTst_ChangeNumberOfTestedCells. This capability, for example, could be used to reduce the duration of the RAM test task before running some other high-bandwidth task in order to prevent task overruns. Such a situation could occur when unusual conditions in a vehicle cause a normally dormant special algorithm to become active.

RamTstNumberOfTestedCells is only applicable to background testing.

RamTstNumberOfTestedCells may not exceed RamTstMaxNumberOfTestedCells.

The absolute maximum size of RamTstNumberOfTestedCells for a given RamTstAlgParams container is defined and documented by the implementer. This maximum should be equal to the sum of the block sizes as defined by the block descriptions. The integrator sets RamTstExtNumberOfTestedCells to this absolute maximum value (pre-compile or link) in the RamTstAlgParams container. RamTstExtNumberOfTestedCells is not changeable during run time.

The integrator also configures (pre-compile or link) the RamTstMaxNumberOfTestedCells for each RamTstAlgParams container. The integrator must carefully select RamTstMaxNumberOfTestedCells such that it puts an upper limit on the run time of RamTst_MainFunction in a background task according to the system needs for throughput. In no case should RamTstMaxNumberOfTestedCells be set to a value greater than RamTstExtNumberOfTestedCells. RamTstMaxNumberOfTestedCells is not changeable during run time.

The minimum value of RamTstNumberOfTestedCells is defined and documented by the implementer. The minimum should be defined as one cell unless there is some physical reason for a larger minimum. The integrator configures (pre-compile or link) the RamTstMinNumberOfTestedCells to be greater than or equal to the minimum defined by the implementer. RamTstMinNumberOfTestedCells applies to the entire RAM test module, and not to individual algorithms or parameter sets. It is configured in the RamTstConfigParams container. RamTstMinNumberOfTestedCells is not changeable during run time.

The cell size (in terms of bits) is also defined by the implementer and cannot be changed at integration time, as it should be a fixed value for a given processor. Therefore the corresponding parameter is specified as a published parameter (see chapter 10.3).

No matter how many blocks or partial blocks are tested in one RamTst_MainFunction scheduled call, test status information must be maintained for each block separately.

RamTst_MainFunction

A **background** test is performed by the scheduler periodically calling the RamTst_MainFunction to test a RamTstNumberOfTestedCells of memory using the selected algorithm, until the entire defined area of RAM is tested. This RamTst_MainFunction can be interrupted at the end of each atomic sequence during a scheduled call.

RamTst_MainFunction:

- Is made up of one or more atomic (i.e. uninterruptable) pieces of code. The number of cells that can be tested in one atomic sequence is considered as implementation specific, thus it is not determined by any (standardized) configuration parameter. However, it is expected that at least RamTstMinNumberOfTestedCells are completely tested during one atomic sequence. It should be noted, that in general the detection of coupling faults between cells is limited to those cells which are tested together in the same atomic sequence.
- At the end of each atomic piece, internal flags are checked to see if an OS task has changed any parameter of the state chart, and to respond to question-type APIs.
- Knows inherently:
  - which algorithm it is using;
  - which memory blocks must be tested for this algorithm,
  - start and end addresses of each block;
  - number of cells to test at each call
  - further parameters for the test (see chapter 7.3)
- Remembers:
  - which block it is in;
  - which address to start at in the next call;
  - status of the test;
  - overall test results;
  - test results for each block.
- When RamTstNumberOfTestedCells is reached, RamTst_MainFunction ends testing for that scheduled call, and starts testing in the next scheduled call at the next (saved) address.
- When the end of a block is reached during a scheduled call, RamTst_MainFunction continues testing at the beginning of the next block, and continues until RamTstNumberOfTestedCells is reached. (Note: The atomic test se-

quence should be careful to take into account any issues regarding crossing into the next block.)

- When all blocks are fully tested, RamTst_MainFunction issues a notification and repeats testing at the first block.
- If there is an error during testing, RamTst_MainFunction issues a notification (if configured) and continues testing.

RamTst_RunFullTest, RamTst_RunPartialTest

"Full" and "Partial" refers to full or partial memory, and **not** the full or partial set of algorithms over the memory space. The test is performed over the specified memory area using only one algorithm. The desired parameter set (which includes the algorithm) is selected by calling the API RamTst_SelectAlgParams before calling the foreground test API.

Note that due to the possibility of testing larger memory areas without interruption the fault coverage of foreground tests is in general better than of background test for the same algorithm.

RamTst_RunFullTest API:

The user calls RamTst_RunFullTest with no arguments (the test parameter set is selected before). This test is normally used for a full RAM check at system startup or shutdown.

Sequence:

- RamTst_Stop
- RamTst_SelectAlgParams to chose the desired parameter set
- RamTst_RunFullTest

RamTst_RunPartialTest API:

The user calls RamTst_RunPartialTest with one argument specifying the desired block to be tested. This test is used for example to check a specified memory section immediately before using that memory. This capability is to enable a system safety concept.

Sequence:

- RamTst_Stop
- RamTst_SelectAlgParams to chose the desired parameter set

- RamTst_RunPartialTest (ChosenBlock)

or if background test shall continue afterwards:

- RamTst_Suspend
- RamTst_RunPartialTest (ChosenBlock)

2 Acronyms and Abbreviations

Abbreviation / Acronym:	Description:
API	Application Programming Interface
CRC	Cyclic Redundancy Check
DEM	Diagnostic Event Manager
DET	Default Error Tracer
DMA	Direct Memory Access
ECC	Error Correction Code
NMI	Non Maskable Interrupt
RAM	Random Access Memory

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

Definitions

Note: These definition are copied from the AUTOSAR_FO_TR_Glossary.pdf [2]

Synchronous: A communication is synchronous when the calling software entity is blocked until the called operation is evaluated. The calling software entity continues its operation by getting the result. Synchronous communication between distributed functional units has to be implemented as remote procedure call.

Asynchronous: Asynchronous communication does not block the sending software entity. The sending software entity continues its operation without getting a response from the communication partner(s). There could be an acknowledgement by the communication system about the sending of the information. A later response to the sending software entity is possible.

3 Related documentation

3.1 Input documents & related standards and norms

[1] ISO 26262-5:2018 Part 5: Product development at the hardware level
https://www.iso.org

[2] Glossary
AUTOSAR_FO_TR_Glossary

[3] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[4] Requirements on RAM Test
AUTOSAR_CP_RS_RAMTest

[5] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[6] General Requirements on SPAL
AUTOSAR_CP_RS_SPALGeneral

3.2 Related specification

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

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

4 Constraints and assumptions

Note: To achieve ISO 26262 compliance, the software implementation must be according to the requirements of ISO 26262 for the required safety integrity level (ASIL A, ASIL B, ASIL C or ASIL D) of the safety goals of the system.

4.1 Limitations

[SWS_RamTst_00002] ⌈During the execution of a RAM test algorithm, no other software shall be allowed to modify the RAM area under test.⌋

In case of background test, the testing code shall be implemented in small atomic pieces in order to accomplish this.

In case of foreground test, it is assumed that the test environment provides the conditions for exclusive access to the tested RAM area.

The rationale behind this requirement is the incapability of the RAM test module to ensure data consistency (e.g. during an NMI, or during a DMA transfer).

[SWS_RamTst_00082] ⌈The implementer shall provide integration hints for each algorithm, e.g. "do not use in parallel with a DMA".⌋

When testing shared memory in a multi-core system it might not be possible to get exclusive access to more than one memory cell via interrupt locking. In this case, the usage of a test configuration for shared memory blocks must be restricted to foreground tests and to specific ECU states, see 3 Related Documentation and [SWS_RamTst_00203] for additional information.

In a multi-core system, disabling the interrupts does not guarantee atomicity for more than a single memory access. Since a RAM test operation consists of more than a single memory access, a more sophisticated mechanism is needed to realize atomicity. Therefore, different solutions for shared and non-shared RAM are required.

4.2 Full RAM Test

A full test shall be executed when only a single core is running. In a Master-Slave system, this is possible during the initial boot phase while only the master core is active. Additionally full tests can be performed during ECU sleep mode. This allows the EcuM to delay the sleep state of one of the cores to perform a RAM tests on that core.

Full RAM tests shall be allowed whenever atomicity across cores can be guaranteed, known moments are,

1. Before the master core has activated any other core.

2. When all cores except one have entered sleep mode.

4.3 Partial RAM Test

During normal operation, the memory is split into non-shared and shared parts. The integrator has to specify for each `ALGORITHM_ID` the memory areas on which the algorithm works. A non-shared area is owned by a specific core and can be tested by the code running on that core as in the single core case. Lack of atomicity in MC causes problems for shared memory.

4.4 Applicability to car domains

No restrictions.

5 Dependencies to other modules

An actual selected parameter set for a RAM test basically consists of a set of RAM bocks and a test algorithm.

The available parameter sets for the RAM blocks and test algorithms must be configured at pre-compile time. The software responsible for monitoring the RAM state of health must then select an appropriate parameter set, (it can also switch between several ones at runtime), according to the results of the ECU safety analysis.

Within each parameter set, the detailed definition of the blocks to be tested, e.g. their start/end address, must be configured at pre-compile or link time. Further parameters controlling the details of the test are explained later in the document (see 7.3).

If the test environment calls a RAM Test API to test all or part of the RAM immediately (in the foreground), then the test environment is responsible to mask interrupts as desired or to call the test in a particular situation, where the tested blocks are not accessed by other modules.

For background testing, the ECU State Manager or the BSW Scheduler must schedule the RAM Test main function. The number of cells tested in one cycle is set as a default at pre-compile or link time based upon the needs of the scheduler. This size may be changed during runtime to accommodate a change in the schedule. In addition, the parameter set used for the background test may be switched during runtime, so that e.g. certain critical blocks can be tested in certain ECU states with higher coverage than in other ECU states or uncritical blocks can be excluded from tests in certain ECU states.

In development mode the error-hook function of module DET will be called.

6 Requirements Tracing

The following tables reference the requirements specified in [4], [5], [6] and links to the fulfillment of these.

Requirement	Description	Satisfied by
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_RamTst_00007] [SWS_RamTst_00099]
[SRS_BSW_00314]	All internal driver modules shall separate the interrupt frame definition from the service routine	[SWS_RamTst_00221]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_RamTst_00033] [SWS_RamTst_00037] [SWS_RamTst_00039] [SWS_RamTst_00040] [SWS_RamTst_00095] [SWS_RamTst_00097] [SWS_RamTst_00170] [SWS_RamTst_00172] [SWS_RamTst_00210] [SWS_RamTst_00214]
[SRS_BSW_00325]	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	[SWS_RamTst_00221]
[SRS_BSW_00331]	All Basic Software Modules shall strictly separate error and status information	[SWS_RamTst_00102] [SWS_RamTst_00103]
[SRS_BSW_00337]	Classification of development errors	[SWS_RamTst_00067]
[SRS_BSW_00339]	Reporting of production relevant error status	[SWS_RamTst_00011] [SWS_RamTst_00067] [SWS_RamTst_00071] [SWS_RamTst_00111] [SWS_RamTst_00213] [SWS_RamTst_00216] [SWS_RamTst_01002] [SWS_RamTst_01005] [SWS_RamTst_01008]
[SRS_BSW_00344]	BSW Modules shall support link-time configuration	[SWS_RamTst_00026] [SWS_RamTst_00027]
[SRS_BSW_00345]	BSW Modules shall support pre-compile configuration	[SWS_RamTst_00058]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_RamTst_00099]
[SRS_BSW_00373]	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	[SWS_RamTst_00110]
[SRS_BSW_00385]	List possible error notifications	[SWS_RamTst_00067] [SWS_RamTst_01002] [SWS_RamTst_01005] [SWS_RamTst_01008]
[SRS_BSW_00402]	Each module shall provide version information	[SWS_RamTst_00081]
[SRS_BSW_00406]	API handling in uninitialized state	[SWS_RamTst_00006]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_RamTst_00093] [SWS_RamTst_01011] [SWS_RamTst_01012]
[SRS_BSW_00427]	ISR functions shall be defined and documented in the BSW module description template	[SWS_RamTst_00221]
[SRS_BSW_00439]	Enable BSW modules to handle interrupts	[SWS_RamTst_00221]
[SRS_BSW_00450]	A Main function of a un-initialized module shall return immediately	[SWS_RamTst_00175]

▽

△

Requirement	Description	Satisfied by
[SRS_RamTst_13800]	The number of tested cells shall be changeable at runtime	[SWS_RamTst_00036] [SWS_RamTst_00107]
[SRS_RamTst_13802]	Multiple RAM areas shall be configurable at post build/ link time	[SWS_RamTst_00026]
[SRS_RamTst_13803]	A subset of available RAM Test algorithms shall be selectable at pre-compile time	[SWS_RamTst_00026] [SWS_RamTst_00027] [SWS_RamTst_00063] [SWS_RamTst_00224] [SWS_RamTst_00225] [SWS_RamTst_00226]
[SRS_RamTst_13804]	A subset of the pre-compile time selected RAM Check test algorithms shall be selectable at runtime	[SWS_RamTst_00083] [SWS_RamTst_00105]
[SRS_RamTst_13809]	It shall be possible to divide the RAM test execution into smaller pieces	[SWS_RamTst_00008] [SWS_RamTst_00026] [SWS_RamTst_00059] [SWS_RamTst_00107] [SWS_RamTst_00108]
[SRS_RamTst_13810]	Current status of RAM test execution per block shall be available through a get status interface	[SWS_RamTst_00010] [SWS_RamTst_00011] [SWS_RamTst_00019] [SWS_RamTst_00024] [SWS_RamTst_00038] [SWS_RamTst_00104]
[SRS_RamTst_13811]	The RAM test module shall be able to perform its tests in a non-destructive manner	[SWS_RamTst_00060] [SWS_RamTst_00061] [SWS_RamTst_00200]
[SRS_RamTst_13812]	The RAM test module shall be able to perform its tests in a destructive manner	[SWS_RamTst_00061] [SWS_RamTst_00201]
[SRS_RamTst_13816]	Effects of Instruction / Data queue shall be taken into account	[SWS_RamTst_00062]
[SRS_RamTst_13820]	RAM test execution status shall be provided by a notification mechanism	[SWS_RamTst_00043] [SWS_RamTst_00044] [SWS_RamTst_00045] [SWS_RamTst_00046] [SWS_RamTst_00113] [SWS_RamTst_00114]
[SRS_RamTst_13822]	A safety mechanism with low coverage shall be available	[SWS_RamTst_00224]
[SRS_RamTst_13823]	A Test algorithm with medium coverage shall be available	[SWS_RamTst_00225]
[SRS_RamTst_13824]	A Test algorithm with high coverage shall be available	[SWS_RamTst_00226]
[SRS_SPAL_00157]	All drivers and handlers of the AUTOSAR Basic Software shall implement notification mechanisms of drivers and handlers	[SWS_RamTst_00043] [SWS_RamTst_00044] [SWS_RamTst_00045] [SWS_RamTst_00046]
[SRS_SPAL_12056]	All driver modules shall allow the static configuration of notification mechanism	[SWS_RamTst_00043] [SWS_RamTst_00044]
[SRS_SPAL_12057]	All driver modules shall implement an interface for initialization	[SWS_RamTst_00007] [SWS_RamTst_00026] [SWS_RamTst_00027]
[SRS_SPAL_12129]	The ISRs shall be responsible for resetting the interrupt flags and calling the according notification function	[SWS_RamTst_00221]
[SRS_SPAL_12163]	All driver modules shall implement an interface for de-initialization	[SWS_RamTst_00146]
[SRS_SPAL_12263]	The implementation of all driver modules shall allow the configuration of specific module parameter types at link time	[SWS_RamTst_00026] [SWS_RamTst_00027]
[SRS_SPAL_12448]	All driver modules shall have a specific behavior after a development error detection	[SWS_RamTst_00033] [SWS_RamTst_00037] [SWS_RamTst_00039] [SWS_RamTst_00040] [SWS_RamTst_00084] [SWS_RamTst_00095]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Requirements

[SWS_RamTst_00005] ⌈The RAM Test module shall provide the background RAM test as an asynchronous service.⌋

[SWS_RamTst_00206] ⌈The RAM Test module shall provide the foreground RAM test as an synchronous service.⌋

[SWS_RamTst_00063]

Upstream requirements: SRS_RamTst_13803

⌈The configuration process for the RAM Test module shall allow the selection of a subset of different RAM Test algorithms during pre-compile time.⌋

This subset is to be chosen from the different RAM Test algorithms as specified in [SWS_RamTst_00224], [SWS_RamTst_00225], [SWS_RamTst_00226], [SWS_RamTst_00204].

[SWS_RamTst_00060]

Upstream requirements: SRS_RamTst_13811

⌈If non-destructive RAM Test is chosen, the RAM Test module shall save the RAM area to be tested before the module modifies it. The RAM Test module shall execute the complete procedure (saving, changing, restoring) without interruption.⌋

Note: "Saving" and "restoring" does not necessarily mean explicit copying actions. If the test algorithm is "transparent" it restores the original content in the tested cells after the test without needing additional memory for saving.

[SWS_RamTst_00061]

Upstream requirements: SRS_RamTst_13811, SRS_RamTst_13812

⌈For both the destructive and non-destructive options, the RAM Test module shall ensure that the test algorithm does not overwrite the RAM Test internal variables.⌋

[SWS_RamTst_00062]

Upstream requirements: SRS_RamTst_13816

⌈After writing to a cell and before reading back, the RAM Test module shall provide the possibility to inject instruction(s) forcing the controller to clear its CPU internal cache.⌋

[SWS_RamTst_00224]

Upstream requirements: SRS_RamTst_13803, SRS_RamTst_13822

⌈Unless covered by some hardware mechanism the RAM Test module shall provide a test algorithm with low coverage as stated in [1], Table D.1.⌋

[SWS_RamTst_00225]

Upstream requirements: SRS_RamTst_13803, SRS_RamTst_13823

⌈Unless covered by some hardware mechanism the RAM Test module shall provide a test algorithm with medium coverage as stated in [1], Table D.1.⌋

[SWS_RamTst_00226]

Upstream requirements: SRS_RamTst_13803, SRS_RamTst_13824

⌈Unless covered by some hardware mechanism the RAM Test module shall provide a test algorithm with high coverage as stated in [1], Table D.1.⌋

[SWS_RamTst_00204] ⌈Unless covered by some hardware mechanism the RAM Test module shall provide a test algorithm, the RAM Test module may provide additional vendor or hardware specific test algorithms or different variants of the algorithms listed above. These algorithms must be clearly documented by the implementer, especially their fault coverage (for vendor specific configuration parameters see [SWS_RamTst_00205].⌋

[SWS_RamTst_00221]

Upstream requirements: SRS_BSW_00427, SRS_BSW_00314, SRS_BSW_00325, SRS_BSW_00439, SRS_SPAL_12129

⌈A processor specific test algorithm is allowed to make use of hardware macros and/or interrupts supporting the detection of data loss (like CRC, ECC) if appropriate. The implementer must describe any interrupt routine in the Basic Software Module Description and the implementation must follow the general requirements for interrupt handling.⌋

7.2 Error Classification

Chapter [3, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of the following error types which may occur in BSW modules.

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

7.2.1 Development Errors

[SWS_RamTst_00067] Definition of development errors in module RamTst

Upstream requirements: SRS_BSW_00337, SRS_BSW_00339, SRS_BSW_00385

⌈

Type of error	*Related error code*	*Error value*
A particular API is called in an unexpected state (see also: [SWS_RamTst_00033], [SWS_RamTst_00037], [SWS_RamTst_00095], [SWS_RamTst_00097], [SWS_RamTst_00170], [SWS_RamTst_00172], [SWS_RamTst_00210], [SWS_RamTst_00214])	RAMTST_E_STATUS_FAILURE	0x01
API parameter out of specified range (see also: [SWS_RamTst_00039], [SWS_RamTst_00040], [SWS_RamTst_00084], [SWS_RamTst_00223])	RAMTST_E_OUT_OF_RANGE	0x02
API service used without module initialization (see also: [SWS_RamTst_00089])	RAMTST_E_UNINIT	0x03
API service called with a NULL pointer	RAMTST_E_PARAM_POINTER	0x04

⌋

[SWS_RamTst_00089] ⌈The function `RamTst_Init` shall be called first before calling any other RAM test functions. If this sequence is not respected, the error code `RAMTST_E_UNINIT` shall be reported to the Default Error Tracer (if development error detection is enabled).⌋

[SWS_RamTst_00069] ⌈Additional errors that are detected because of specific implementation and/or specific hardware properties shall be added in the RAM Test device specific implementation specification. The classification and enumeration shall be compatible to the errors listed above in [SWS_RamTst_00067].⌋

7.2.2 Runtime Errors

There are no runtime errors.

7.2.3 Production Errors

[SWS_RamTst_00071]

Upstream requirements: SRS_BSW_00339

⌈Production errors shall be reported to Diagnostic Event Manager (DEM) via the `Dem_SetEventStatus` API.⌋

[SWS_RamTst_01002] RAM failure during test.

Upstream requirements: SRS_BSW_00339, SRS_BSW_00385

⌈

Diagnostic Event (Error Name)	RAMTST_MAIN_RAM_FAILURE
Description	The function RamTst_MainFunction shall update the overall test result status and error is updated based on the current status of RamTstAlgParams.See: SWS_RamTst_00011.
Failed condition	If test result of at least one block is RAMTST_RESULT_NOT_OK. See SWS_RamTst_01003.
Passed condition	If test result of all the blocks is RAMTST_RESULT_OK. See SWS_RamTst_01004.

⌋

[SWS_RamTst_01003] ⌈RAM test for `RamTst_MainFunction` function shall Fail if test result of at least one block is `RAMTST_RESULT_NOT_OK`.⌋

[SWS_RamTst_01004] ⌈RAM test for `RamTst_MainFunction` function shall Pass if test result of all the blocks is `RAMTST_RESULT_OK`.⌋

[SWS_RamTst_01005] RAM failure during test.

Upstream requirements: SRS_BSW_00339, SRS_BSW_00385

⌈

Diagnostic Event (Error Name)	RAMTST_RUNFL_RAM_FAILURE
Description	The function RamTst_RunFullTest shall update the overall test result status and error is updated based on the current status of RamTstAlgParams. See: SWS_RamTst_00213.
Failed condition	If test result of at least one block is RAMTST_RESULT_NOT_OK. See SWS_RamTst_01006.
Passed condition	If test result of all the blocks is RAMTST_RESULT_OK. See SWS_RamTst_01007.

⌋

[SWS_RamTst_01006] ⌈RAM Test for `RamTst_RunFullTest` function shall Fail if test result of at least one block is `RAMTST_RESULT_NOT_OK`.⌋

[SWS_RamTst_01007] ⌈RAM Test for `RamTst_RunFullTest` function shall Pass If test result of all the blocks is `RAMTST_RESULT_OK`.⌋

[SWS_RamTst_01008] RAM failure during test.

Upstream requirements: SRS_BSW_00339, SRS_BSW_00385

⌈

Diagnostic Event (Error Name)	RAMTST_PART_RAM_FAILURE
Description	The function RamTst_RunPartialTest shall update the test result status of the tested block and error is updated based on the current status of RamTstAlgParams. See: SWS_RamTst_00216.
Failed condition	If test result of at least one block is RAMTST_RESULT_NOT_OK. See SWS_RamTst_01009.

▽

△

Passed condition	If test result of all the blocks is RAMTST_RESULT_OK. See SWS_RamTst_01010.

⌋

[SWS_RamTst_01009] ⌈RAM Test for `RamTst_RunPartialTest` function function shall Fail if test result of at least one block is `RAMTST_RESULT_NOT_OK`.⌋

[SWS_RamTst_01010] ⌈RAM Test for `RamTst_RunPartialTest` function function shall Pass If test result of all the blocks is `RAMTST_RESULT_OK`.⌋

7.3 General Test Behavior

This sections contains detailed specifications items which hold for the foreground test and the background test as well.

Both foreground of background tests are controlled by the currently selected parameter set `RamTstAlgParams` which defines the test algorithm, the set of memory blocks to be tested and several attributes controlling the behavior of the test.

Note that the same test algorithm can be used as part of several different `RamTstAlgParams`, that none of the `RamTstAlgParams` must necessarily contain all memory blocks and that (in general) for foreground and background tests different `RamTstAlgParams` can be selected. This allows for a flexible approach of usings the tests differently in specific ECU modes or for different types of memory.

[SWS_RamTst_00200]

Upstream requirements: SRS_RamTst_13811

⌈If the configuration parameter `RamTstTestPolicy` for a block is set to `RAMTEST_NON_DESTRUCTIVE`, the test algorithm shall restore the original memory content of the tested cells of this block after the test (given that no error is detected).⌋

Hint: For a transparent test algorithm, this behavior is automatically fulfilled without additional overhead. For a non-transparent test algorithm, this option means overhead in runtime/memory in order to save and restore the content.

[SWS_RamTst_00201]

Upstream requirements: SRS_RamTst_13812

⌈If the configuration parameter `RamTstTestPolicy` for a block is set to `RAMTEST_DESTRUCTIVE`, the test algorithm shall fill the tested cells after the test with the bit pattern defined for this block by parameter `RamTstFillPattern` (given that no error is detected).⌋

This requirement shall ensure reproducible behavior.

Hint: For a transparent test algorithm, specifying this option would mean runtime overhead. For a non-transparent algorithm, the runtime overhead can be minimized, if the fill pattern corresponds to a constant value left behind by the algorithm anyhow.

[SWS_RamTst_00202] ⌈The overall test result - for the set of blocks in the current `RamTstAlgParams` - shall be

- `RAMTST_RESULT_NOT_TESTED` if no test was started yet (after reset or de-init).
- `RAMTST_RESULT_UNDEFINED` if a test was started, not all blocks have yet been tested and no block result is `RAMTST_RESULT_NOT_OK`.
- `RAMTST_RESULT_OK` if all blocks have been tested with result status `RAMTST_RESULT_OK`.
- `RAMTST_RESULT_NOT_OK` if at least one block test result is `RAMTST_RESULT_NOT_OK` regardless whether all blocks have been already tested or not.

⌋

[SWS_RamTst_00207] ⌈The test result for a specific block (identified in the given `RamTstAlgParams`) - shall be

- `RAMTST_RESULT_NOT_TESTED` if this block is considered as not yet tested.
- `RAMTST_RESULT_UNDEFINED` if a test on this block is running.
- `RAMTST_RESULT_OK` if all memory cells in this block have been tested sucessfully.
- `RAMTST_RESULT_NOT_OK` if a failure has been detected for at least one memory cell in this block.

⌋

For a given processor type, memory layout and fault model, not all possible combinations of test algorithms, block configurations and their attributes make sense. For example:

- The implementer might want to exclude a certain combination of test algorithm and `RamTstTestPolicy`.
- A certain test algorithm might have to be excluded from background tests due to performance reason.
- Some memory blocks might have to be excluded from background tests due to performance reasons or because an exclusive access cannot be guaranteed under normal operation (e.g. for shared memory).

This leads to the following requirement:

[SWS_RamTst_00203] ⌈The implementer shall document possible restrictions for the combination of configuration parameters and for their usage in background/foreground

tests. Where applicable, he shall support this by the definition of predefined or recommended configuration parameter values attached to the BSW Module Description.⌋

8 API specification

8.1 Imported types

This chapter lists data type definitions for the included variables and constants.

[SWS_RamTst_00098] Definition of imported datatypes of module RamTst ⌈

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

⌋

8.2 Type definitions

8.2.1 RamTst_ConfigType

[SWS_RamTst_01011] Definition of datatype RamTst_ConfigType

Upstream requirements: SRS_BSW_00414

⌈

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

⌋

8.2.2 RamTst_ExecutionStatusType

[SWS_RamTst_00189] Definition of datatype RamTst_ExecutionStatusType ⌈

Name	RamTst_ExecutionStatusType		
Kind	Enumeration		
Range	RAMTST_EXECUTION_UNINIT	0x00	The RAM Test is not initialized or not usable.
	RAMTST_EXECUTION_STOPPED	0x01	The RAM Test is stopped and ready to be started in foreground or to be allowed in background.

▽

△

	RAMTST_EXECUTION_RUNNING	0x02	The RAM Test is currently running.
	RAMTST_EXECUTION_SUSPENDED	0x03	The background RAM Test is waiting to be resumed.
Description	This is a status value returned by the API service RamTst_GetExecutionStatus().		
Available via	RamTst.h		

⌋

[SWS_RamTst_00006]

Upstream requirements: SRS_BSW_00406

⌈For the type `RamTst_ExecutionStatusType`, the enumeration value `RAMTST_EXECUTION_UNINIT` shall be the default value after a reset.⌋

8.2.3 RamTst_TestResultType

[SWS_RamTst_00190] Definition of datatype RamTst_TestResultType ⌈

Name	RamTst_TestResultType		
Kind	Enumeration		
Range	RAMTST_RESULT_NOT_TESTED	0x00	The RAM Test is not executed.
	RAMTST_RESULT_OK	0x01	The RAM Test has been tested with OK result
	RAMTST_RESULT_NOT_OK	0x02	The RAM Test has been tested with NOT-OK result.
	RAMTST_RESULT_UNDEFINED	0x03	The RAM Test is currently running.
Description	This is a status value returned by the API service RamTst_GetTestResult().		
Available via	RamTst.h		

⌋

[SWS_RamTst_00012] ⌈For the type `RamTst_TestResultType` (of the overall test result), the enumeration value `RAMTST_RESULT_NOT_TESTED` shall be the default value after a reset.⌋

For more details on the usage of this status see chapter 7.3.

8.2.4 RamTst_AlgParamsIdType

[SWS_RamTst_00191] Definition of datatype RamTst_AlgParamsIdType ⌈

Name	RamTst_AlgParamsIdType
Kind	Type
Derived from	uint8

▽

△

Range	0...255	–	–
Description	Data type used to identify a set of configuration parameters for a test algorithm.		
Available via	RamTst.h		

⌋

[SWS_RamTst_00188] ⌈For the type `RamTst_AlgParamsIdType`, the value 0 shall indicate, that no test parameters (and thus no test algorithm) is selected. This shall be the default value of the corresponding variable after reset.⌋

8.2.5 RamTst_AlgorithmType

[SWS_RamTst_00227] Definition of datatype RamTst_AlgorithmType ⌈

Name	RamTst_AlgorithmType		
Kind	Enumeration		
Range	RAMTST_ALGORITHM_UNDEFINED	0x00	Undefined algorithm (uninitialized value)
	RAMTST_CHECKERBOARD_TEST	0x01	Checkerboard test algorithm
	RAMTST_MARCH_TEST	0x02	March test algorithm
	RAMTST_WALK_PATH_TEST	0x03	Walk path test algorithm
	RAMTST_GALPAT_TEST	0x04	Galpat test algorithm
	RAMTST_TRANSP_GALPAT_TEST	0x05	Transparent Galpat test algorithm
	RAMTST_ABRAHAM_TEST	0x06	Abraham test algorithm
Description	This is a value returned by the API service RamTst_GetTestAlgorithm().		
Available via	RamTst.h		

⌋

[SWS_RamTst_00013] ⌈For the type `RamTst_AlgorithmType`, the enumeration value `RAMTST_ALGORITHM_UNDEFINED` shall be the default value after reset.⌋

[SWS_RamTst_00058]

Upstream requirements: SRS_BSW_00345

⌈The type `RamTst_AlgorithmType` shall contain only the enumerations of the algorithms selected at pre-compile time.⌋

Note that if vendor specific algorithms were defined (see [SWS_RamTst_00205]), the enumeration fields of `RamTst_AlgorithmType` should be extended accordingly by the implementer (or by a code generator).

8.2.6 RamTst_NumberOfTestedCellsType

[SWS_RamTst_00173] Definition of datatype RamTst_NumberOfTestedCellsType ⌈

Name	RamTst_NumberOfTestedCellsType		
Kind	Type		
Derived from	uint32		
Range	1...(2^32-1)	–	–
Description	Data type of number of tested RAM cells		
Available via	RamTst.h		

⌋

8.2.7 RamTst_NumberOfBlocksType

[SWS_RamTst_00174] Definition of datatype RamTst_NumberOfBlocksType ⌈

Name	RamTst_NumberOfBlocksType		
Kind	Type		
Derived from	uint16		
Range	1...65535	–	–
Description	Data type used to identify or count RAM blocks given in the test configuration parameters.		
Available via	RamTst.h		

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 RamTst_Init

[SWS_RamTst_00099] Definition of API function RamTst_Init

Upstream requirements: SRS_BSW_00101, SRS_BSW_00358

⌈

Service Name	RamTst_Init

Syntax	`void RamTst_Init (`
	`  const RamTst_ConfigType* ConfigPtr`
	`)`

Service ID [hex]	0x00
Sync/Async	Synchronous
Reentrancy	Non Reentrant

▽

△

Parameters (in)	ConfigPtr	Pointer to the selected configuration set.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service for RAM Test initialization.	
Available via	RamTst.h	

⌋

[SWS_RamTst_01012]

Upstream requirements: SRS_BSW_00414

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

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

Note: See also [SWS_RamTst_00093] in 10.2.1.

[SWS_RamTst_00007]

Upstream requirements: SRS_BSW_00101, SRS_SPAL_12057

⌈The function `RamTst_Init` shall initialize all RAM Test relevant registers and global variables and change the execution status to `RAMTST_EXECUTION_STOPPED`. The test is initialized to use the default test parameter set (`RamTstDefaultAlgParamsId`) as configured by its `RamTstAlgParams` container.⌋

[SWS_RamTst_00096] ⌈If the DET is enabled and the execution status of the RAM Test is not `RAMTST_EXECUTION_UNINIT`, the function `RamTst_Init` shall report the error value `RAMTST_E_STATUS_FAILURE` to the DET, and then immediately return.⌋

8.3.2 RamTst_DeInit

[SWS_RamTst_00146] Definition of API function RamTst_DeInit

Upstream requirements: SRS_SPAL_12163

⌈

Service Name	RamTst_DeInit
Syntax	`void RamTst_DeInit ( void )`
Service ID [hex]	0x0c
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None

▽

△

Parameters (out)	None
Return value	None
Description	Service for RAM Test deinitialization.
Available via	RamTst.h

⌋

[SWS_RamTst_00147] ⌈The function `RamTst_DeInit` shall deinitialize all RAM Test relevant registers and global variables that were initialized by `RamTst_Init` and change the execution status to `RAMTST_EXECUTION_UNINIT`.⌋

If the RAM Test is in the `RAMTST_EXECUTION_UNINIT` state after a `RamTst_DeInit` call, a call to any RamTst Module function (except `RamTst_Init`) may result in unknown software behavior.

8.3.3 RamTst_Stop

[SWS_RamTst_00100] Definition of API function RamTst_Stop ⌈

Service Name	RamTst_Stop
Syntax	`void RamTst_Stop ( void )`
Service ID [hex]	0x02
Sync/Async	Asynchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Service for stopping the RAM Test.
Available via	RamTst.h

⌋

[SWS_RamTst_00014] ⌈When the `RamTst_Stop` function is called, `RamTst_MainFunction` shall still finish the current atomic sequence (if it is executing), and afterward the status shall be set to `RAMTST_EXECUTION_STOPPED`. The test result is retained, but test parameters and loop data are not.⌋

[SWS_RamTst_00148] ⌈After a `RamTst_Stop` call, `RamTst_MainFunction` shall not begin testing again when called by the scheduler until after a `RamTst_Allow` call.⌋

[SWS_RamTst_00033]

Upstream requirements: SRS_BSW_00323, SRS_SPAL_12448

⌈If the DET is enabled and the execution status of the RAM Test is not `RAMTST_EXECUTION_RUNNING` or `RAMTST_EXECUTION_SUSPENDED`, the function `RamTst_`

Stop shall report the error value RAMTST_E_STATUS_FAILURE to the DET, and then immediately return.⌋

The RamTst_Stop API can be enabled or disabled by the configuration parameter RamTstStopApi within the container RamTstCommon.

8.3.4 RamTst_Allow

[SWS_RamTst_00149] Definition of API function RamTst_Allow ⌈

Service Name	RamTst_Allow
Syntax	`void RamTst_Allow ( void )`
Service ID [hex]	0x03
Sync/Async	Asynchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Service for continuing the RAM Test after calling 'RamTst_Stop.
Available via	RamTst.h

⌋

[SWS_RamTst_00169] ⌈The function RamTst_Allow shall permit the RamTst_MainFunction to perform testing at its next scheduled call, if it had been stopped. Therefore, it shall change the execution status to RAMTST_EXECUTION_RUNNING, if it has been RAMTST_EXECUTION_STOPPED. The test shall be continued with the default test parameter set (RamTstDefaultAlgParamsId) as configured by its RamTstAlgParams container.⌋

[SWS_RamTst_00170]

Upstream requirements: SRS_BSW_00323

⌈If DET is enabled and the execution status is not RAMTST_EXECUTION_STOPPED, the function RamTst_Allow shall report the error value RAMTST_E_STATUS_FAILURE to the DET, and then immediately return.⌋

The RamTst_Allow API can be enabled or disabled by the configuration parameter RamTstAllowApi within the container RamTstCommon.

8.3.5 RamTst_Suspend

[SWS_RamTst_00150] Definition of API function RamTst_Suspend ⌈

Service Name	RamTst_Suspend

Syntax	`void RamTst_Suspend (`
	`  void`
	`)`

Service ID [hex]	0x0d
Sync/Async	Asynchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Service for suspending current operation of background RAM Test, until RESUME is called.
Available via	RamTst.h

⌋

[SWS_RamTst_00171] ⌈The function RamTst_Suspend shall temporarily prohibit the RamTst_MainFunction from performing tests at its next scheduled call. When RamTst_Suspend is called and the execution status is RAMTST_EXECUTION_RUNNING, testing stops after the current atomic sequence, test result and current test states are retained and the execution status is changed to RAMTST_EXECUTION_SUSPENDED.⌋

[SWS_RamTst_00172]

Upstream requirements: SRS_BSW_00323

⌈If DET is enabled and the execution status is not RAMTST_EXECUTION_RUNNING, the function RamTst_Suspend shall report the error value RAMTST_E_STATUS_FAILURE to the DET, and then immediately return.⌋

The RamTst_Suspend API can be enabled or disabled by the configuration parameter RamTstSuspendApi within the container RamTstCommon.

8.3.6 RamTst_Resume

[SWS_RamTst_00101] Definition of API function RamTst_Resume ⌈

Service Name	RamTst_Resume

Syntax	`void RamTst_Resume (`
	`  void`
	`)`

Service ID [hex]	0x0e
Sync/Async	Asynchronous
Reentrancy	Non Reentrant
Parameters (in)	None

▽

△

Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Service for allowing to continue the background RAM Test at the point is was suspended.
Available via	RamTst.h

⌋

[SWS_RamTst_00018] ⌈The function RamTst_Resume shall permit the RamTst_MainFunction to continue testing at the point where it was suspended, at its next scheduled call. Testing continues according to the saved test states. The function RamTst_Resume shall change the execution status to RAMTST_EXECUTION_RUNNING if it has been RAMTST_EXECUTION_SUSPENDED.⌋

[SWS_RamTst_00037]

Upstream requirements: SRS_BSW_00323, SRS_SPAL_12448

⌈If DET is enabled and the execution status of the RAM Test module is not RAMTST_EXECUTION_SUSPENDED, the function RamTst_Resume shall report the error value RAMTST_E_STATUS_FAILURE to the DET, and then immediately return.⌋

The RamTst_Resume API can be enabled or disabled by the configuration parameter RamTstResumeApi within the container RamTstCommon.

8.3.7 RamTst_GetExecutionStatus

[SWS_RamTst_00102] Definition of API function RamTst_GetExecutionStatus

Upstream requirements: SRS_BSW_00331

⌈

Service Name	RamTst_GetExecutionStatus	
Syntax	`RamTst_ExecutionStatusType RamTst_GetExecutionStatus ( void )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	RamTst_ExecutionStatusType	See type definition
Description	Service returns the current RAM Test execution status.	
Available via	RamTst.h	

⌋

[SWS_RamTst_00019]

Upstream requirements: SRS_RamTst_13810

⌈The function `RamTst_GetExecutionStatus` shall return the current RAM Test execution status.⌋

The `RamTst_GetExecutionStatus` API can be enabled or disabled by the configuration parameter `RamTstGetExecutionStatusApi` within the container `RamTstCommon`.

8.3.8 RamTst_GetTestResult

[SWS_RamTst_00103] Definition of API function RamTst_GetTestResult

Upstream requirements: SRS_BSW_00331

⌈

Service Name	RamTst_GetTestResult	
Syntax	`RamTst_TestResultType RamTst_GetTestResult ( void )`	
Service ID [hex]	0x05	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	RamTst_TestResultType	See type definition
Description	Service returns the current RAM Test result.	
Available via	RamTst.h	

⌋

[SWS_RamTst_00024]

Upstream requirements: SRS_RamTst_13810

⌈The function `RamTst_GetTestResult` shall return the current RAM test result.⌋

The test result is determined according to [SWS_RamTst_00202].

The `RamTst_GetTestResult` API can be enabled or disabled by the configuration parameter `RamTstGetTestResultApi` within the container `RamTstCommon`.

8.3.9 RamTst_GetTestResultPerBlock

[SWS_RamTst_00104] Definition of API function RamTst_GetTestResultPerBlock

Upstream requirements: SRS_RamTst_13810

⌈

Service Name	RamTst_GetTestResultPerBlock	
Syntax	`RamTst_TestResultType RamTst_GetTestResultPerBlock ( RamTst_NumberOfBlocksType BlockID )`	
Service ID [hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockID	Identifies the block
Parameters (inout)	None	
Parameters (out)	None	
Return value	RamTst_TestResultType	See type definition
Description	Service returns the current RAM Test result for the specified block.	
Available via	RamTst.h	

⌋

[SWS_RamTst_00038]

Upstream requirements: SRS_RamTst_13810

⌈The function `RamTst_GetTestResultPerBlock` shall return the current RAM test result for the specified block.⌋

The test result per block is determined according to [SWS_RamTst_00207].

[SWS_RamTst_00039]

Upstream requirements: SRS_BSW_00323, SRS_SPAL_12448

⌈If DET is enabled and the `BlockID` is out of range, the function `RamTst_GetTestResultPerBlock` shall report the error value `RAMTST_E_OUT_OF_RANGE` to the DET and return the test result value `RAMTST_RESULT_UNDEFINED`.⌋

Hint: "Out of range" means here, that the `BlockID` does not match to one of the values configured for the currently selected `RamTstAlgParams`/ `RamTstBlockParams`/ `RamTstBlockId`, see [ECUC_RamTst_00143].

The `RamTst_GetTestResultPerBlock` API can be enabled or disabled by the configuration parameter `RamTstGetTestResultPerBlockApi` within the container `RamTstCommon`.

8.3.10 RamTst_GetAlgParams

[SWS_RamTst_00193] Definition of API function RamTst_GetAlgParams ⌈

Service Name	RamTst_GetAlgParams	
Syntax	`RamTst_AlgParamsIdType RamTst_GetAlgParams ( void )`	
Service ID [hex]	0x12	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	RamTst_AlgParamsId Type	See type definition.
Description	Service returns the ID of the current RAM Test algorithm parameter set.	
Available via	RamTst.h	

⌋

[SWS_RamTst_00194] ⌈The function `RamTst_GetAlgParams` shall return the ID of the currently selected test algorithm parameter set (i.e. the ID of the currently selected `RamTstAlgParams` in the container `RamTstConfigParams`).⌋

The `RamTst_GetAlgParams` API can be enabled or disabled by the configuration parameter `RamTstGetAlgParamsApi` within the container `RamTstCommon`.

8.3.11 RamTst_GetTestAlgorithm

[SWS_RamTst_00106] Definition of API function RamTst_GetTestAlgorithm ⌈

Service Name	RamTst_GetTestAlgorithm	
Syntax	`RamTst_AlgorithmType RamTst_GetTestAlgorithm ( void )`	
Service ID [hex]	0x07	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	RamTst_AlgorithmType	See type definition
Description	Service returns the current RAM Test algorithm.	
Available via	RamTst.h	

⌋

[SWS_RamTst_00021] ⌈The function `RamTst_GetTestAlgorithm` shall return the current RAM Test algorithm.⌋

The RamTst_GetTestAlgorithm API can be enabled or disabled by the configuration parameter RamTstGetTestAlgorithmApi within the container RamTstCommon.

8.3.12 RamTst_GetNumberOfTestedCells

[SWS_RamTst_00108] Definition of API function RamTst_GetNumberOfTestedCells

Upstream requirements: SRS_RamTst_13809

⌈

Service Name	RamTst_GetNumberOfTestedCells	
Syntax	`RamTst_NumberOfTestedCellsType RamTst_GetNumberOfTestedCells ( void )`	
Service ID [hex]	0x09	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	RamTst_NumberOfTestedCellsType	Number of currently tested cells per cycle.
Description	Service returns the current number of tested cells per main-function cycle.	
Available via	RamTst.h	

⌋

[SWS_RamTst_00034] ⌈The function RamTst_GetNumberOfTestedCells shall read the current number of tested cells per cycle.⌋

The RamTst_GetNumberOfTestedCells API can be enabled or disabled by the configuration parameter RamTstGetNumberOfTestedCellsApi within the container RamTstCommon.

8.3.13 RamTst _SelectAlgParams

[SWS_RamTst_00105] Definition of API function RamTst_SelectAlgParams

Upstream requirements: SRS_RamTst_13804

⌈

Service Name	RamTst_SelectAlgParams
Syntax	`void RamTst_SelectAlgParams ( RamTst_AlgParamsIdType NewAlgParamsId )`

▽

△

Service ID [hex]	0x0b	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	NewAlgParamsId	Identifies the parameter set to be used.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service used to set the test algorithm and its parameter set.	
Available via	RamTst.h	

⌋

[SWS_RamTst_00083]

Upstream requirements: SRS_RamTst_13804

⌈The function `RamTst_SelectAlgParams` shall select the test parameter set (i.e. one of the `RamTstAlgParams` in the container `RamTstConfigParams`) to be used by the RAM Test module.⌋

Note: Depending on the configured content of `RamTstAlgParams`, this function may be used to select a different test algorithm. But the function may also be used to select a different set of blocks (e.g. for foreground testing) for the same test algorithm.

[SWS_RamTst_00085] ⌈The function `RamTst_SelectAlgParams` shall re-initialize all RAM Test relevant registers and global variables with the values for the "NewAlg ParamsId".⌋

[SWS_RamTst_00084]

Upstream requirements: SRS_SPAL_12448

⌈If DET is enabled and the parameter "NewAlgParamsId" is out of range, the function `RamTst_SelectAlgParams` shall report the error value `RAMTST_E_OUT_OF_RANGE` to the DET, leaving the current `RamTstAlgParams` unchanged.⌋

Hint: "Out of range" means, that the "NewAlgParamsId" does not match to one of the configured values for `RamTstAlgParams`/ `RamTstAlgParamsId`, see [ECUC_RamTst_00179].

[SWS_RamTst_00097]

Upstream requirements: SRS_BSW_00323

⌈If DET is enabled and the execution status of the RAM Test module is not `RAMTST_EXECUTION_STOPPED`, the function `RamTst_SelectAlgParams` shall report the error value `RAMTST_E_STATUS_FAILURE` to the DET, then immediately return from the function.⌋

[SWS_RamTst_00094] ⌈The function `RamTst_SelectAlgParams` shall initialize the test result status (according to [SWS_RamTst_00207] and [SWS_RamTst_00202]).⌋

Hint: It makes no sense to keep the previous test results at this point (as it was specified in former version of this document), since the block structure might change due to the selected parameter set, so the previous result could no longer be interpreted. If the test environment wants to save the previous results, it can easily retrieve them via RamTst_GetTestResult or RamTst_GetTestResultPerBlock before calling RamTst_SelectAlgParams.

The RamTst_SelectAlgParams API can be enabled or disabled by the configuration parameter RamTstSelectAlgParamsApi within the container RamTstCommon.

8.3.14 RamTst_ChangeNumberOfTestedCells

[SWS_RamTst_00107] Definition of API function RamTst_ChangeNumberOfTestedCells

Upstream requirements: SRS_RamTst_13800, SRS_RamTst_13809

⌈

Service Name	RamTst_ChangeNumberOfTestedCells	
Syntax	`void RamTst_ChangeNumberOfTestedCells ( RamTst_NumberOfTestedCellsType NewNumberOfTestedCells )`	
Service ID [hex]	0x08	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	NewNumberOfTested Cells	See type definition
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service changes the current number of tested cells.	
Available via	RamTst.h	

⌋

[SWS_RamTst_00036]

Upstream requirements: SRS_RamTst_13800

⌈The function RamTst_ChangeNumberOfTestedCells shall change the current number of tested cells (for background tests).⌋

[SWS_RamTst_00040]

Upstream requirements: SRS_BSW_00323, SRS_SPAL_12448

⌈If DET is enabled and the parameter NewNumberOfTestedCells is out of range (min= MinNumberOfTestedCells / max = MaxNumberOfTestedCells), the function RamTst_ChangeNumberOfTestedCells shall report the error value RAMTST_E_OUT_OF_RANGE to the DET. The function shall leave the number of tested cells unchanged.⌋

[SWS_RamTst_00095]

Upstream requirements: SRS_BSW_00323, SRS_SPAL_12448

⌈If the execution status of the RAM Test module is not in the status RAMTST_EXECUTION_STOPPED, the function RamTst_ChangeNumberOfTestedCells shall not change the current number of tested cells and (if DET is enabled) shall report the error value RAMTST_E_STATUS_FAILURE to the DET.⌋

The RamTst_ChangeNumberOfTestedCells API can be enabled or disabled by the configuration parameter RamTstChangeNumOfTestedCellsApi within the container RamTstCommon.

8.3.15 RamTst_RunFullTest

[SWS_RamTst_00195] Definition of API function RamTst_RunFullTest ⌈

Service Name	RamTst_RunFullTest

Syntax	`void RamTst_RunFullTest (`
	`  void`
	`)`

Service ID [hex]	0x10
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Service for executing the full RAM Test in the foreground.
Available via	RamTst.h

⌋

[SWS_RamTst_00196] ⌈If the RAM Test execution status is RAMTST_EXECUTION_STOPPED, the function RamTst_RunFullTest shall test all RAM blocks defined in the selected RamTstAlgParams.⌋

[SWS_RamTst_00210]

Upstream requirements: SRS_BSW_00323

⌈If DET is enabled and the execution status of the RAM Test module is not RAMTST_EXECUTION_STOPPED, the function RamTst_RunFullTest shall report the error value RAMTST_E_STATUS_FAILURE to the DET, and then immediately return.⌋

[SWS_RamTst_00211] ⌈If the RAM Test execution status is RAMTST_EXECUTION_STOPPED, the function RamTst_RunFullTest shall set the execution status to RAMTST_EXECUTION_RUNNING during the test and set it back to RAMTST_EXECUTION_STOPPED before returning.⌋

[SWS_RamTst_00212] ⌈The function RamTst_RunFullTest shall update the test result status of single blocks according to [SWS_RamTst_00207].⌋

[SWS_RamTst_00213]

Upstream requirements: SRS_BSW_00339

⌈The function RamTst_RunFullTest shall update the overall test result status according to [SWS_RamTst_00202]. If at least one block test result is RAMTST_RESULT_NOT_OK, then the function shall report the production error RAMTST_RUNFL_RAM_FAILURE to the DEM.⌋

The RamTst_RunFullTest API can be enabled or disabled by the configuration parameter RamTstRunFullTestApi within the container RamTstCommon.

Destruction or restoration of the memory content are handled according to the requirements [SWS_RamTst_00200] and [SWS_RamTst_00201].

For pre-conditions on the function RamTst_RunFullTest, see requirement [SWS_RamTst_00002].

Implementation Hints:

For reasons of effiency and optimum fault coverage, the implementation of RamTst_RunFullTest shall assume, that it has exclusive access to all memory blocks contained in its RamTstAlgParams during the call. This allows to apply the test algorithm on a wider range of memory than in background test, which increases the fault coverage especially for coupling faults.

Thus it is the responsibility of the test environment, to either provide appropriate resource locking, or to call the function in an ECU mode, where the memory blocks of the selected RamTstAlgParams are not in use. The test environment must also ensure, that RamTst_MainFunction is not scheduled during the foreground test.

A test algorithm usually requires various write and read cycles over a given range of memory. Some algorithms also require multiple walks through this range. It is up to the implementation, whether such a tested range corresponds to one block (which means, that the full test is split into several ranges) or even includes several or all blocks. This depends on performance issues and the assumed fault model. In any case, the test behavior must be clearly documented.

8.3.16 RamTst_RunPartialTest

[SWS_RamTst_00197] Definition of API function RamTst_RunPartialTest ⌈

Service Name	RamTst_RunPartialTest

Syntax	`void RamTst_RunPartialTest (`
	`  RamTst_NumberOfBlocksType BlockId`
	`)`

Service ID [hex]	0x11
Sync/Async	Synchronous
Reentrancy	Non Reentrant

▽

△

Parameters (in)	BlockId	Identifies the single RAM block to be tested in the selected set of RamTstAlgParams.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service for testing one RAM block in the foreground.	
Available via	RamTst.h	

⌋

[SWS_RamTst_00198] ⌈If the RAM Test execution status is RAMTST_EXECUTION_STOPPED or RAMTST_EXECUTION_SUSPENDED, the function RamTst_RunPartialTest shall test the specified RAM block.⌋

[SWS_RamTst_00214]

Upstream requirements: SRS_BSW_00323

⌈If DET is enabled and the RAM test execution status is neither RAMTST_EXECUTION_STOPPED nor RAMTST_EXECUTION_SUSPENDED, the function RamTst_RunPartialTest shall report the error value RAMTST_E_STATUS_FAILURE to the DET, and then immediately return.⌋

[SWS_RamTst_00215] ⌈If the RAM Test execution status is RAMTST_EXECUTION_STOPPED or RAMTST_EXECUTION_SUSPENDED, the function RamTst_RunPartialTest shall set the execution status to RAMTST_EXECUTION_RUNNING during the test, and after the test shall set it back to the previous state (the state when the function was called).⌋

[SWS_RamTst_00216]

Upstream requirements: SRS_BSW_00339

⌈The function RamTst_RunPartialTest shall update the test result status of the tested block according to [SWS_RamTst_00207]. If this block test result is RAMTST_RESULT_NOT_OK, then the function shall report the production error RAMTST_PART_RAM_FAILURE to the DEM.⌋

[SWS_RamTst_00217] ⌈A successful partial foreground test shall set the block specific result to RAMTST_RESULT_OK. It shall not modify the overall test result. A failing partial foreground test shall set both, the block specific as well as the overall test result to RAMTST_RESULT_NOT_OK.⌋

[SWS_RamTst_00223] ⌈If DET is enabled and the BlockId is out of range, the function RamTst_RunPartialTest shall report the error value RAMTST_E_OUT_OF_RANGE to the DET and then immediately return.⌋

Notes:

'Out of range' in [SWS_RamTst_00223] means that the BlockId does not match to one of the configured values for the currently selected Block Identifier (RamTstAlgParams/ RamTstBlockParams/ RamTstBlockId).

Updating the test results will overwrite the result from a previous test of this block and the overall test result in case of failure, including the result from a suspended background test.

The RamTst_RunPartialTest API can be enabled or disabled by the configuration parameter RamTstRunPartialTestApi within the container RamTstCommon.

Destruction or restoration of the memory content is handled according to the requirements [SWS_RamTst_00200] and [SWS_RamTst_00201].

For pre-conditions on the function RamTst_RunPartialTest, see requirement [SWS_RamTst_00002].

Implementation Hints:

The implementation hints given for RamTst_RunFullTest also apply here, as far as applicable to one single block.

8.4 Scheduled functions

For details refer to [3] Chapter 8.4 *"Scheduled functions"*.

8.4.1 RamTst_MainFunction

[SWS_RamTst_00110] Definition of scheduled function RamTst_MainFunction

Upstream requirements: SRS_BSW_00373

⌈

Service Name	RamTst_MainFunction
Syntax	`void RamTst_MainFunction ( void )`
Service ID [hex]	0x01
Description	Scheduled function for executing the RAM Test in the background.
Available via	SchM_RamTst.h

⌋

[SWS_RamTst_00008]

Upstream requirements: SRS_RamTst_13809

⌈If the RAM Test execution status is RAMTST_EXECUTION_RUNNING, the function RamTst_MainFunction shall continue to test the RAM blocks defined in the selected RamTstAlgParams.⌋

[SWS_RamTst_00009] ⌈If the RAM Test execution status is RAMTST_EXECUTION_RUNNING and if no blocks have yet been tested (first call of the function), then the function RamTst_MainFunction shall start testing with the first configured RAM block in the selected RamTstAlgParams.⌋

[SWS_RamTst_00175]

Upstream requirements: SRS_BSW_00450

⌈If the RAM Test execution status is not RAMTST_EXECUTION_RUNNING when this API is called, the function RamTst_MainFunction shall return immediately without any actions.⌋

[SWS_RamTst_00010]

Upstream requirements: SRS_RamTst_13810

⌈The function RamTst_MainFunction shall update the test result status of single blocks according to [SWS_RamTst_00207].⌋

[SWS_RamTst_00011]

Upstream requirements: SRS_BSW_00339, SRS_RamTst_13810

⌈The function RamTst_MainFunction shall update the overall test result status according to [SWS_RamTst_00202]. If at least one block test result is RAMTST_RESULT_NOT_OK, then the function shall report the production error RAMTST_MAIN_RAM_FAILURE to the DEM.⌋

[SWS_RamTst_00047] ⌈After the function RamTst_MainFunction has completed testing all RAM blocks configured in the selected RamTstAlgParams, the next call of the function RamTst_MainFunction shall restart the test from the beginning.⌋

[SWS_RamTst_00059]

Upstream requirements: SRS_RamTst_13809

⌈The function RamTst_MainFunction shall test the defined number of RAM cells within one call. The defined number is specified by the function RamTst_ChangeNumberOfTestedCells or by initialization.⌋

Notes:

Updating the test results will overwrite the result from a previous test of the current block and the overall test result, including the case that the background test was resumed after a partial foreground test of the current block.

Destruction or restoration of the memory content are handled according to the requirements [SWS_RamTst_00200] and [SWS_RamTst_00201].

For pre-conditions on the function RamTst_MainFunction, see requirement [SWS_RamTst_00002].

Implementation Hints:

In general, the actual test algorithm within one call of `RamTst_MainFunction` must be performed within one or more atomic sequences. Only within one atomic sequence, the memory written by the algorithm is allowed to be corrupted during the test. This means, that the algorithm can be applied only to those cells accessed within one atomic sequence, so that the detection of coupling faults between cells (by background test) is restricted to those cells which are included in one atomic sequence.

An atomic sequence can either be declared as exclusive area via the BSW module description (see [5, RS BSW General], [SRS_BSW_00426]), leaving the actual locking method to the BSW Scheduler, or be directly implemented via interrupt locking (see [5, RS BSW General], [SRS_BSW_00429]). The latter is allowed, because the RAM test module belongs to the MCAL layer.

8.5 Expected interfaces

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

8.5.1 Mandatory interfaces

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

[SWS_RamTst_00111] Definition of mandatory interfaces required by module RamTst

Upstream requirements: SRS_BSW_00339

⌈

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

⌋

8.5.2 Optional interfaces

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

[SWS_RamTst_00112] Definition of optional interfaces requested by module Ram Tst ⌈

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

⌋

8.5.3 Callout Interfaces

There is no callout interface required for this module.

8.5.4 Configurable interfaces

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

Terms and definitions:

Reentrant: interface is expected to be reentrant

Don't care: reentrancy of interface not relevant for this module (in general, it is in this case not reentrant).

[SWS_RamTst_00043]

Upstream requirements: SRS_SPAL_00157, SRS_SPAL_12056, SRS_RamTst_13820

⌈The callback notifications shall have no parameters and no return value.⌋

[SWS_RamTst_00044]

Upstream requirements: SRS_SPAL_00157, SRS_SPAL_12056, SRS_RamTst_13820

⌈If a callback notification is configured as null pointer, the RAM Test module shall not execute the callback.⌋

8.5.4.1 RamTst_TestCompletedNotification

[SWS_RamTst_00113] Definition of configurable interface RamTst_TestCompletedNotification

Upstream requirements: SRS_RamTst_13820

⌈

Service Name	RamTst_TestCompletedNotification
Syntax	`void RamTst_TestCompletedNotification ( void )`
Sync/Async	Synchronous
Reentrancy	Don't care
Parameters (in)	None
Parameters (inout)	None

▽

△

Parameters (out)	None
Return value	None
Description	The function RamTst_TestCompleted shall be called every time when all RAM blocks of the current test configuration have been tested in the background.
Available via	RamTst.h

⌋

[SWS_RamTst_00045]

Upstream requirements: SRS_SPAL_00157, SRS_RamTst_13820

⌈The RAM Test module shall call the callback `RamTst_TestCompletedNotification` every time when it has tested all RAM blocks of the selected parameter set in a background test.⌋

The callback notification `RamTst_TestCompletedNotification` requires configuration of the parameter `RamTstTestCompletedNotification` within the container `RamTstConfigParams`.

8.5.4.2 RamTst_ErrorNotification

[SWS_RamTst_00114] Definition of configurable interface RamTst_ErrorNotification

Upstream requirements: SRS_RamTst_13820

⌈

Service Name	RamTst_ErrorNotification
Syntax	`void RamTst_ErrorNotification ( void )`
Sync/Async	Synchronous
Reentrancy	Don't care
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	The function RamTst_Error shall be called every time when a RAM failure has been detected by the selected test algorithm in the background.
Available via	RamTst.h

⌋

[SWS_RamTst_00046]

Upstream requirements: SRS_SPAL_00157, SRS_RamTst_13820

⌈The RAM test module shall call the callback `RamTst_ErrorNotification` every time when the test algorithm of the selected parameter set has detected a RAM failure in a background test.⌋

The callback notification RamTst_ErrorNotification requires configuration of the parameter RamTstTestErrorNotification within the container RamTstConfig-Params.

9 Sequence diagrams

9.1 RamTst_MainFunction (Examples)

The first example sequence shows the initialization of the RAM Test module, a foreground Run Full Test request, error notification, and the cyclic call background testing.

A cyclic background task called by a scheduler consists of several small atomic sequences in succession. At the end of each atomic sequence, the command variables are checked to see if any command has been received, and corresponding actions are taken.

The stop request is handled following the currently running atomic sequence of the main routine, or at the next cyclic call of the main routine if it is not currently running. The allow request is handled at the next cyclic call of the main routine.

The second example shows the Suspend and Resume commands, a foreground Run Partial Test request, a Test Completed notification, and the DeInit command.

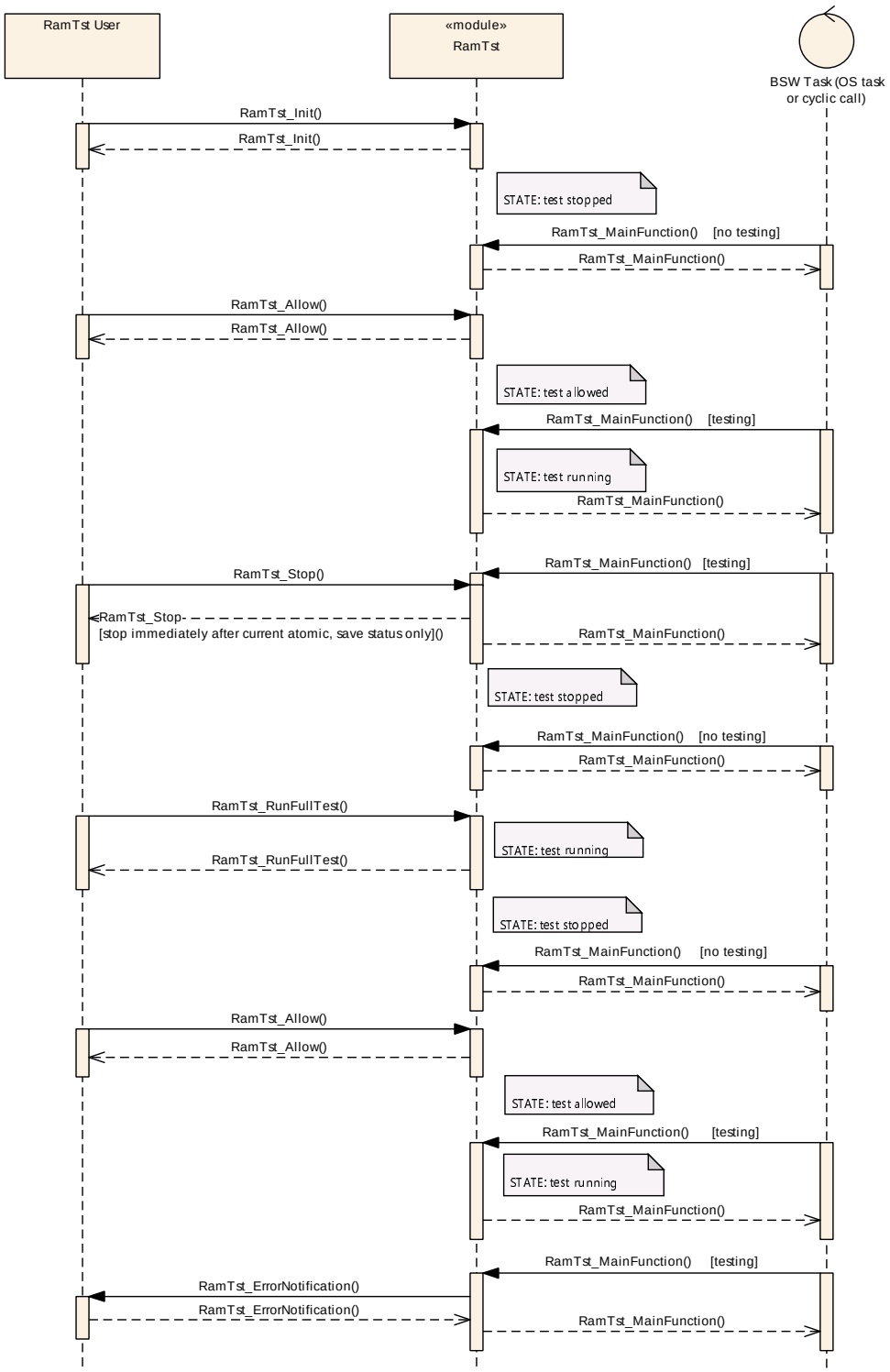


Figure 9.1: Initialization and foreground Run Full Test of the RAM Test module

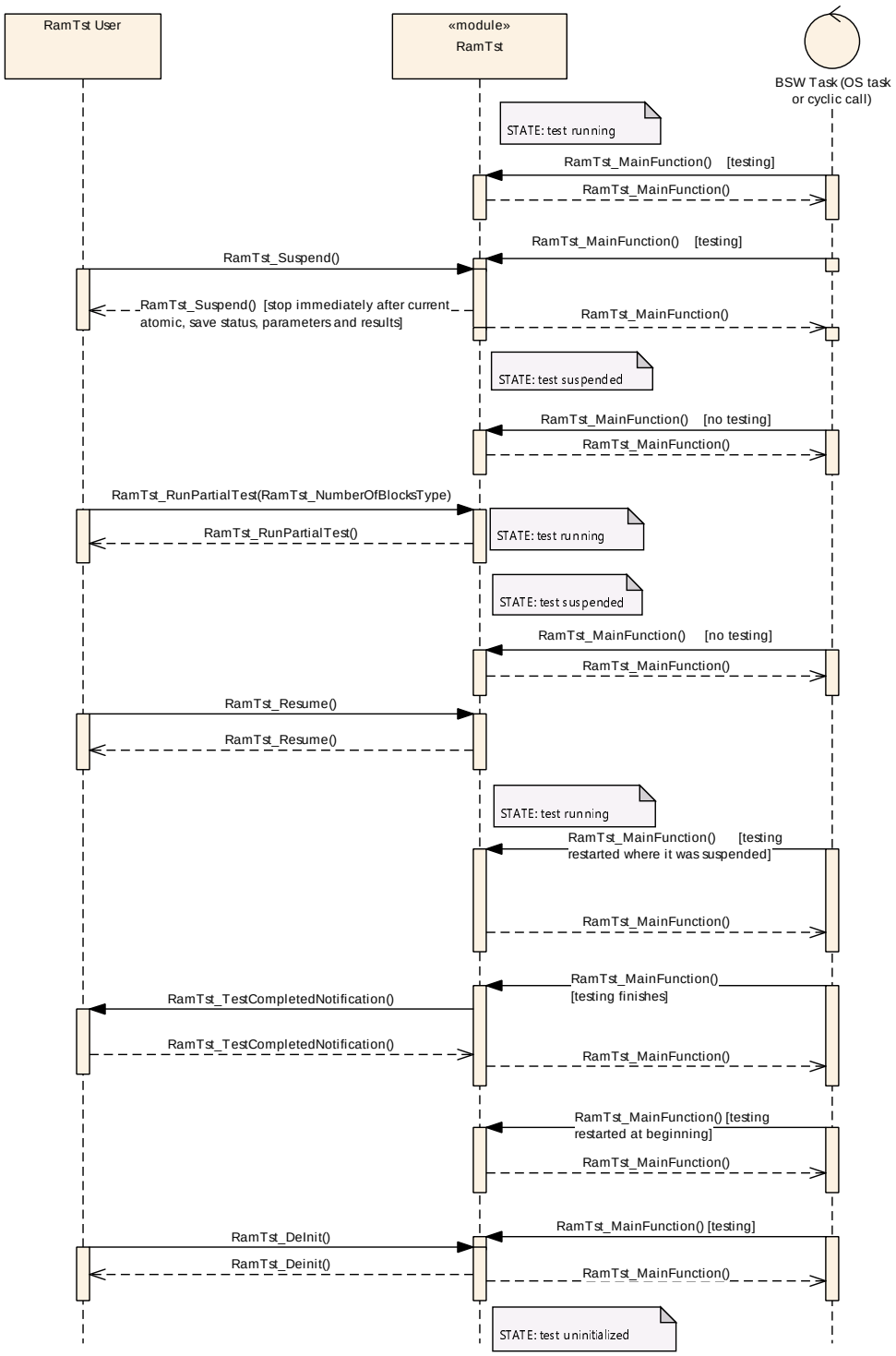


Figure 9.2: Deinitialization and foreground Run Partial Test of the RAM Test module

9.2 RamTst_ChangeNumberOfTestedCells

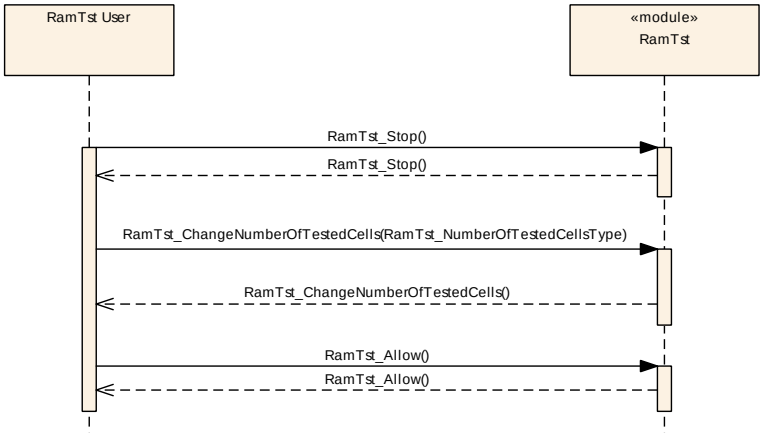


Figure 9.3: Change Number of Tested Cells

9.3 RamTst_SelectAlgParams

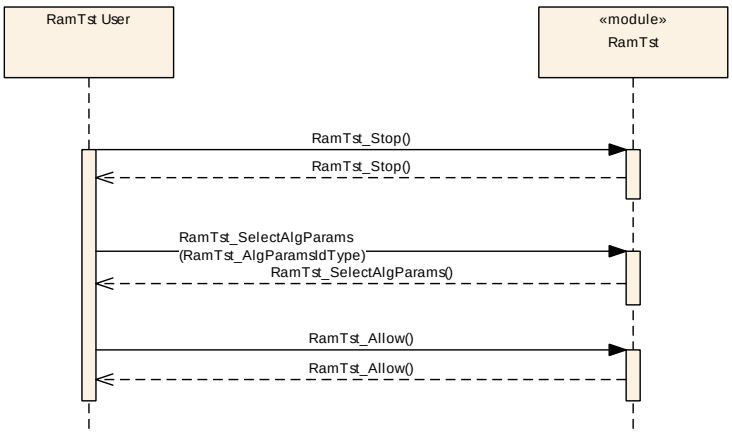


Figure 9.4: Set and Select Algorithm Parameters

9.4 RamTst_GetAlgParams

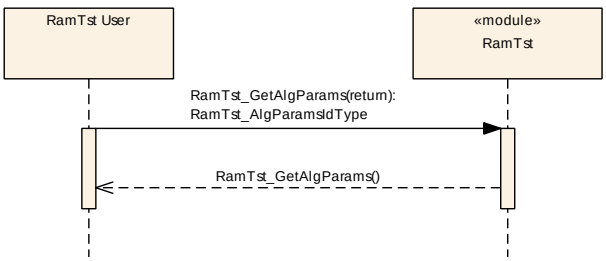


Figure 9.5: Get Algorithm Parameters

9.5 RamTst_GetExecutionStatus

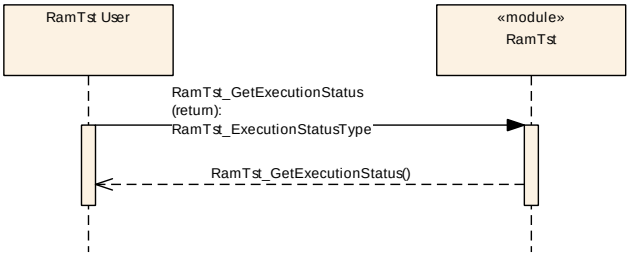


Figure 9.6: Get test execution status

9.6 RamTst_GetTestResult

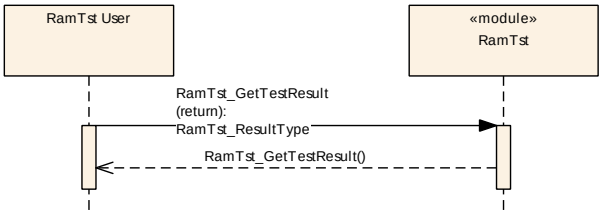


Figure 9.7: Get Test Result

9.7 RamTst_GetTestResultPerBlock

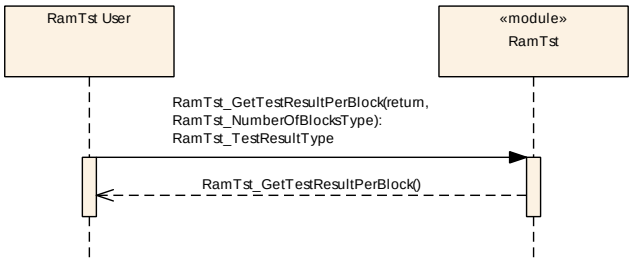


Figure 9.8: Get Test Result per Block

9.8 RamTst_GetTestAlgorithm

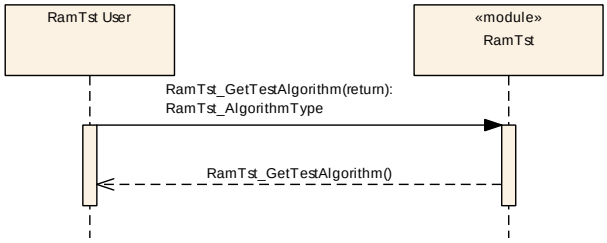


Figure 9.9: Get Test Algorithm

9.9 RamTst_GetNumberOfTestedCells

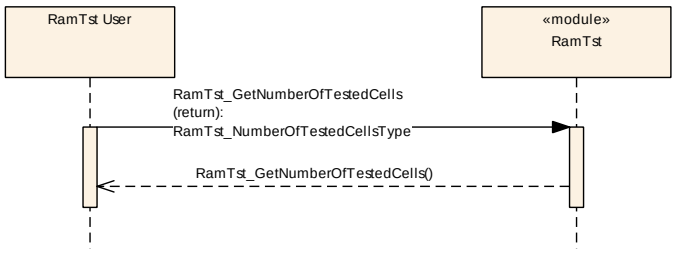


Figure 9.10: Get Number of Tested Cells

10 Configuration specification

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

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

Chapter 10.3 specifies published information of the module RAM Test.

Chapter 10.4 contains additional information for the module RAM Test.

10.1 How to read this chapter

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

[SWS_RamTst_01013] ⌈The RAM Test module shall reject configurations with partition mappings which are not supported by the implementation.⌋

10.2 Containers and configuration parameters

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

[SWS_RamTst_00026]

Upstream requirements: SRS_BSW_00344, SRS_SPAL_12057, SRS_SPAL_12263, SRS_RamTst_13802, SRS_RamTst_13803, SRS_RamTst_13809

⌈Within the configuration data for the RAM Test module, there shall be a set of configuration containers named `RamTstAlgParams`. Each one defines a possible test parameter set to be selected at runtime, which includes an algorithm. The algorithms included in `RamTstAlgParams` are restricted to those that were pre-compile selected to be available to the user via the container `RamTstAlgorithms`.⌋

[SWS_RamTst_00027]

Upstream requirements: SRS_BSW_00344, SRS_SPAL_12057, SRS_SPAL_12263, SRS_RamTst_13803

⌈Within the configuration data for the RAM Test module, each parameter set of type `RamTstAlgParams` (as required in [SWS_RamTst_00026]) shall also have memory block configuration containers. The memory block configuration container is defined in `RamTstBlockParams`. The number of memory block configuration containers is defined by the integrator according to the RAM test strategy.⌋

10.2.1 Variants

[SWS_RamTst_00093]

Upstream requirements: SRS_BSW_00414

⌈Init functions shall have a pointer to a configuration structure as single parameter. This means that, in accordance with [SRS_BSW_00414] only one interface for initialization shall be implemented and it shall not depend on the modules configuration which interface the calling software module shall use.⌋

10.2.2 RamTst

[ECUC_RamTst_00150] Definition of EcucModuleDef RamTst ⌈

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

Included Containers		
Container Name	**Multiplicity**	**Description**
RamTstCommon	1	This container holds a list of all available functions in the RamTst module. Each function is turned ON or OFF before compiling so that only the desired functions and test algorithms are in the compiled code.
RamTstDemEventParameterRefs	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.
RamTstPublishedInformation	1	Container holding all RamTst specific published information parameter.

⌋

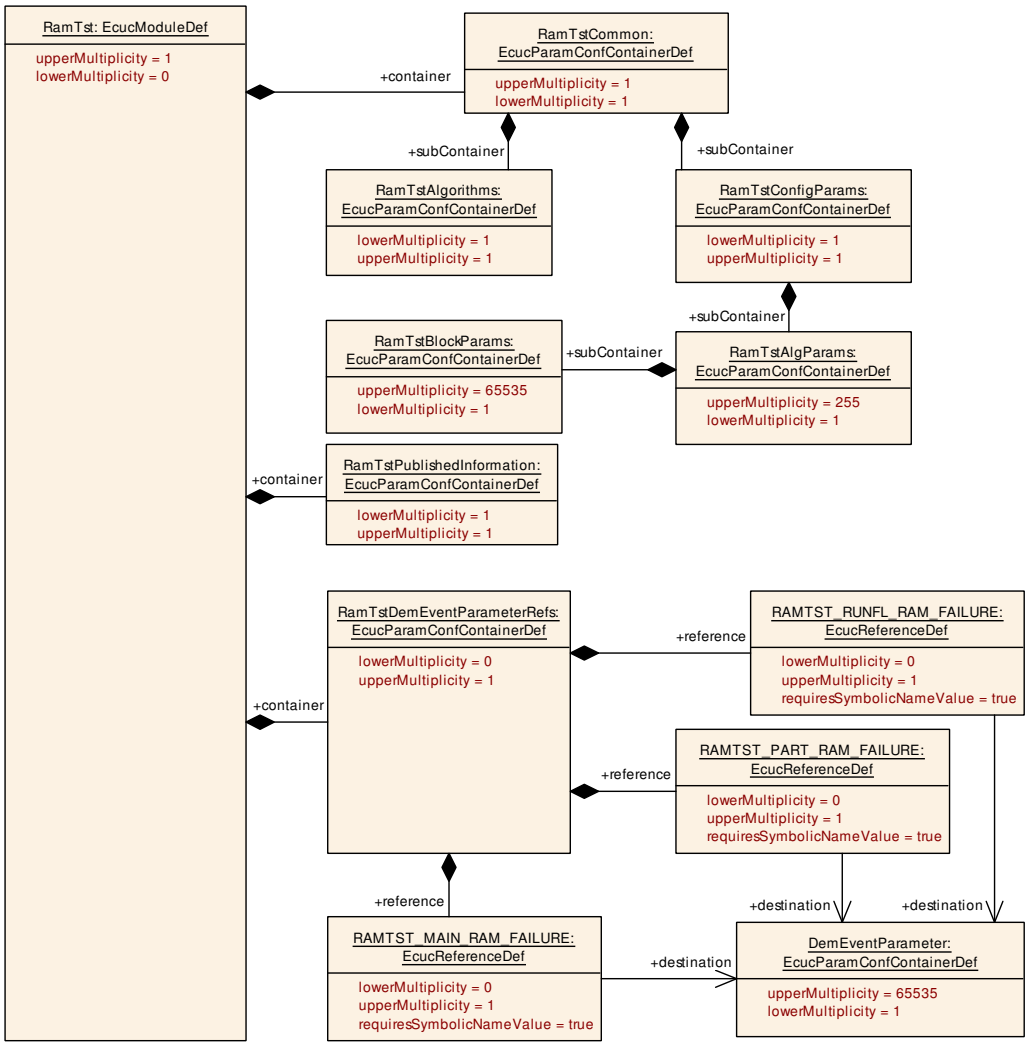


Figure 10.1: RAM Test Configuration Overview

10.2.3 RamTstDemEventParameterRefs

[ECUC_RamTst_00188] Definition of EcucParamConfContainerDef RamTstDemEventParameterRefs ⌈

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

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
RAMTST_MAIN_RAM_FAILURE	0..1	[ECUC_RamTst_00194]
RAMTST_PART_RAM_FAILURE	0..1	[ECUC_RamTst_00193]
RAMTST_RUNFL_RAM_FAILURE	0..1	[ECUC_RamTst_00192]

No Included Containers

⌋

[ECUC_RamTst_00194] Definition of EcucReferenceDef RAMTST_MAIN_RAM_FAILURE ⌈

Parameter Name	RAMTST_MAIN_RAM_FAILURE
Parent Container	RamTstDemEventParameterRefs
Upstream mapping defined	–
Description	Reference to the DemEventParameter which shall be issued when the error "RAM main test failure" has occurred.
Multiplicity	0..1
Type	Symbolic name reference to DemEventParameter
Pre-Compile only	true

⌋

[ECUC_RamTst_00193] Definition of EcucReferenceDef RAMTST_PART_RAM_FAILURE ⌈

Parameter Name	RAMTST_PART_RAM_FAILURE
Parent Container	RamTstDemEventParameterRefs
Upstream mapping defined	–
Description	Reference to the DemEventParameter which shall be issued when the error "RAM partial test failure" has occurred.
Multiplicity	0..1
Type	Symbolic name reference to DemEventParameter
Pre-Compile only	true

⌋

[ECUC_RamTst_00192] Definition of EcucReferenceDef RAMTST_RUNFL_RAM_FAILURE ⌈

Parameter Name	RAMTST_RUNFL_RAM_FAILURE
Parent Container	RamTstDemEventParameterRefs
Upstream mapping defined	–
Description	Reference to the DemEventParameter which shall be issued when the error "RAM full test failure" has occurred.
Multiplicity	0..1
Type	Symbolic name reference to DemEventParameter
Pre-Compile only	true

⌋

10.2.4 RamTstCommon

[ECUC_RamTst_00070] Definition of EcucParamConfContainerDef RamTstCommon ⌈

Container Name	RamTstCommon
Parent Container	RamTst
Upstream mapping defined	–
Description	This container holds a list of all available functions in the RamTst module. Each function is turned ON or OFF before compiling so that only the desired functions and test algorithms are in the compiled code.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
RamTstAllowApi	1	[ECUC_RamTst_00120]
RamTstChangeNumOfTestedCellsApi	1	[ECUC_RamTst_00118]
RamTstDevErrorDetect	1	[ECUC_RamTst_00121]
RamTstGetAlgParamsApi	1	[ECUC_RamTst_00183]
RamTstGetExecutionStatusApi	1	[ECUC_RamTst_00122]
RamTstGetNumberOfTestedCellsApi	1	[ECUC_RamTst_00123]
RamTstGetTestAlgorithmApi	1	[ECUC_RamTst_00124]
RamTstGetTestResultApi	1	[ECUC_RamTst_00125]
RamTstGetTestResultPerBlockApi	1	[ECUC_RamTst_00126]
RamTstResumeApi	1	[ECUC_RamTst_00155]
RamTstRunFullTestApi	1	[ECUC_RamTst_00184]
RamTstRunPartialTestApi	1	[ECUC_RamTst_00185]
RamTstSelectAlgParamsApi	1	[ECUC_RamTst_00182]
RamTstStopApi	1	[ECUC_RamTst_00127]
RamTstSuspendApi	1	[ECUC_RamTst_00156]
RamTstEcucPartitionRef	0..*	[ECUC_RamTst_00190]

Included Containers		
Container Name	**Multiplicity**	**Description**
RamTstAlgorithms	1	This container holds all of the available test algorithms for the specific microcontroller. Each test algorithm is selected ON or OFF before compiling so that only the desired test algorithms are in the compiled code.
RamTstConfigParams	1	This container specifies configuration parameters which are set at pre-compile or link time.

⌋

[ECUC_RamTst_00120] Definition of EcucBooleanParamDef RamTstAllowApi ⌈

Parameter Name	RamTstAllowApi
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	Preprocessor switch to disable / enable the function "RamTst_Allow".
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00118] Definition of EcucBooleanParamDef RamTstChangeNumOfTestedCellsApi ⌈

Parameter Name	RamTstChangeNumOfTestedCellsApi
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	Preprocessor switch to disable / enable the function "RamTst_ChangeNumberOfTested Cells".
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00121] Definition of EcucBooleanParamDef RamTstDevErrorDetect ⌈

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

⌋

[ECUC_RamTst_00183] Definition of EcucBooleanParamDef RamTstGetAlgParamsApi ⌈

Parameter Name	RamTstGetAlgParamsApi
Parent Container	RamTstCommon
Upstream mapping defined	–

▽

△

Description	Preprocessor switch to disable / enable the function "RamTst_GetAlgParams".
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00122] Definition of EcucBooleanParamDef RamTstGetExecutionStatusApi ⌈

Parameter Name	RamTstGetExecutionStatusApi
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	Preprocessor switch to disable / enable the function "RamTst_GetExecutionStatus"
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00123] Definition of EcucBooleanParamDef RamTstGetNumberOfTestedCellsApi ⌈

Parameter Name	RamTstGetNumberOfTestedCellsApi
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	Preprocessor switch to disable / enable the function "RamTst_GetNumberOfTested Cells".
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00124] Definition of EcucBooleanParamDef RamTstGetTestAlgorithmApi ⌈

Parameter Name	RamTstGetTestAlgorithmApi
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	Preprocessor switch to disable / enable the function "RamTst_GetTestAlgorithm"
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00125] Definition of EcucBooleanParamDef RamTstGetTestResultApi ⌈

Parameter Name	RamTstGetTestResultApi
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	Preprocessor switch to disable / enable the function "RamTst_GetTestResult"
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00126] Definition of EcucBooleanParamDef RamTstGetTestResultPerBlockApi ⌈

Parameter Name	RamTstGetTestResultPerBlockApi
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	Preprocessor switch to disable / enable the function "RamTst_GetTestResultPerBlock"
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00155] Definition of EcucBooleanParamDef RamTstResumeApi ⌈

Parameter Name	RamTstResumeApi
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	Preprocessor switch to disable / enable the function "RamTst_Resume".
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00184] Definition of EcucBooleanParamDef RamTstRunFullTestApi ⌈

Parameter Name	RamTstRunFullTestApi
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	Preprocessor switch to disable / enable the function "RamTst_RunFullTest"
Multiplicity	1

▽

△

Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00185] Definition of EcucBooleanParamDef RamTstRunPartialTestApi ⌈

Parameter Name	RamTstRunPartialTestApi
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	Preprocessor switch to disable / enable the function "RamTst_RunPartialTest"
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00182] Definition of EcucBooleanParamDef RamTstSelectAlgParamsApi ⌈

Parameter Name	RamTstSelectAlgParamsApi
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	Preprocessor switch to disable / enable the function "RamTst_SelectAlgParams".
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00127] Definition of EcucBooleanParamDef RamTstStopApi ⌈

Parameter Name	RamTstStopApi
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	Preprocessor switch to disable / enable the function "RamTst_Stop"
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00156] Definition of EcucBooleanParamDef RamTstSuspendApi

⌈

Parameter Name	RamTstSuspendApi
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	Preprocessor switch to disable / enable the function "RamTst_Suspend".
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00190] Definition of EcucReferenceDef RamTstEcucPartitionRef

⌈

Parameter Name	RamTstEcucPartitionRef
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	Maps the RAM test to zero or multiple ECUC partitions to make the modules API available in this partition. The RAM test will operate as an independent instance in each of the partitions.
Multiplicity	0..*
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

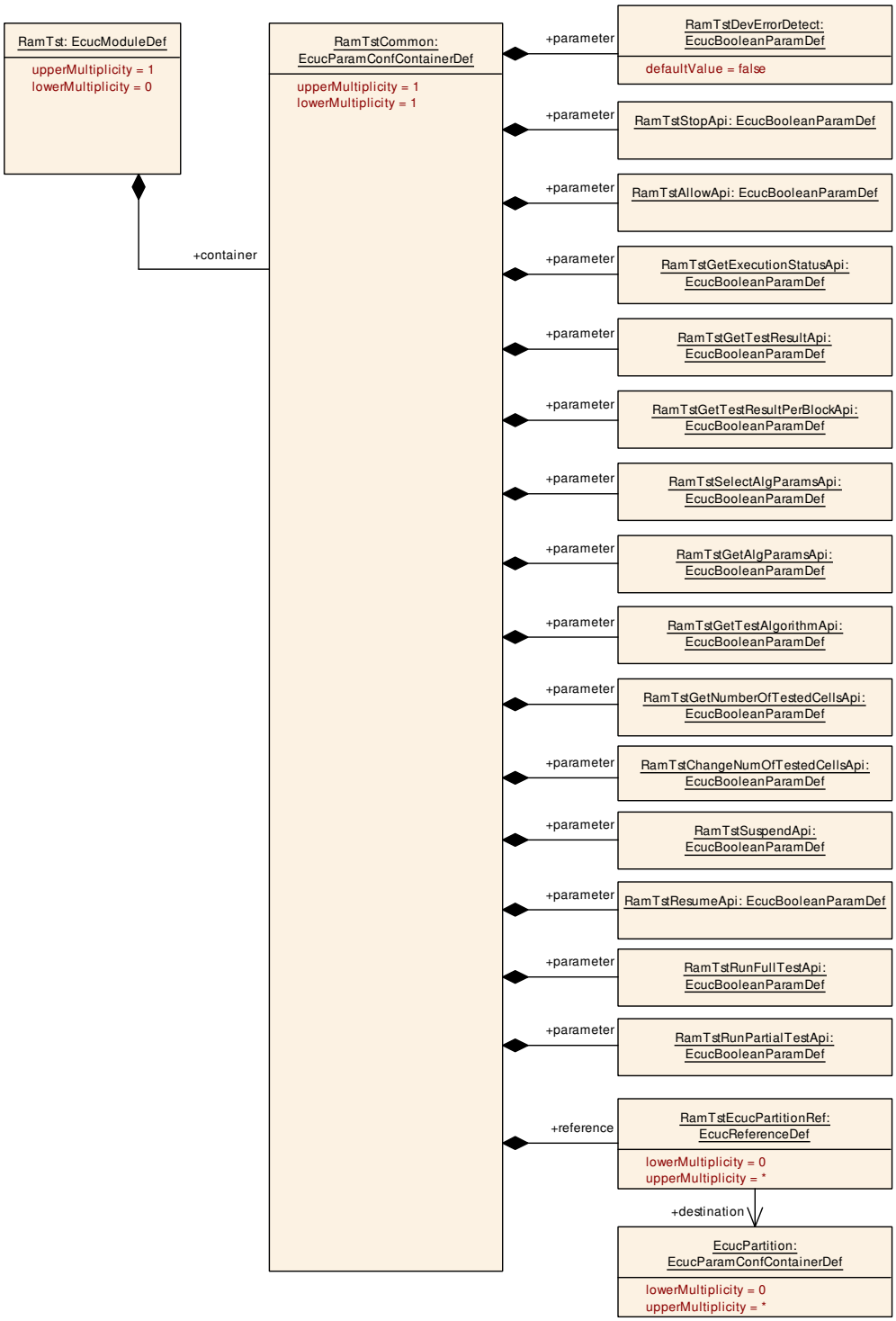


Figure 10.2: RamTstCommon

10.2.5 RamTstAlgorithms

[ECUC_RamTst_00065] Definition of EcucParamConfContainerDef RamTstAlgorithms ⌈

Container Name	RamTstAlgorithms
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	This container holds all of the available test algorithms for the specific microcontroller. Each test algorithm is selected ON or OFF before compiling so that only the desired test algorithms are in the compiled code.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
RamTstAbrahamTestSelected	1	[ECUC_RamTst_00129]
RamTstCheckerboardTestSelected	1	[ECUC_RamTst_00130]
RamTstGalpatTestSelected	1	[ECUC_RamTst_00132]
RamTstMarchTestSelected	1	[ECUC_RamTst_00133]
RamTstTranspGalpatTestSelected	1	[ECUC_RamTst_00134]
RamTstWalkPathTestSelected	1	[ECUC_RamTst_00135]

No Included Containers

⌋

[ECUC_RamTst_00129] Definition of EcucBooleanParamDef RamTstAbrahamTestSelected ⌈

Parameter Name	RamTstAbrahamTestSelected
Parent Container	RamTstAlgorithms
Upstream mapping defined	–
Description	Preprocessor switch to select the Abraham Test ON or OFF
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00130] Definition of EcucBooleanParamDef RamTstCheckerboardTestSelected ⌈

Parameter Name	RamTstCheckerboardTestSelected
Parent Container	RamTstAlgorithms
Upstream mapping defined	–
Description	Preprocessor switch to select the Checkerboard Test ON or OFF
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00132] Definition of EcucBooleanParamDef RamTstGalpatTest Selected ⌈

Parameter Name	RamTstGalpatTestSelected
Parent Container	RamTstAlgorithms
Upstream mapping defined	–
Description	Preprocessor switch to select the Galpat Test ON or OFF
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00133] Definition of EcucBooleanParamDef RamTstMarchTest Selected ⌈

Parameter Name	RamTstMarchTestSelected
Parent Container	RamTstAlgorithms
Upstream mapping defined	–
Description	Preprocessor switch to select the March Test ON or OFF
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00134] Definition of EcucBooleanParamDef RamTstTranspGalpatTestSelected ⌈

Parameter Name	RamTstTranspGalpatTestSelected
Parent Container	RamTstAlgorithms
Upstream mapping defined	–
Description	Preprocessor switch to select the Transparent Galpat Test ON or OFF
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00135] Definition of EcucBooleanParamDef RamTstWalkPathTestSelected ⌈

Parameter Name	RamTstWalkPathTestSelected
Parent Container	RamTstAlgorithms
Upstream mapping defined	–
Description	Preprocessor switch to select the Walking Path Test ON or OFF
Multiplicity	1

▽

△

Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

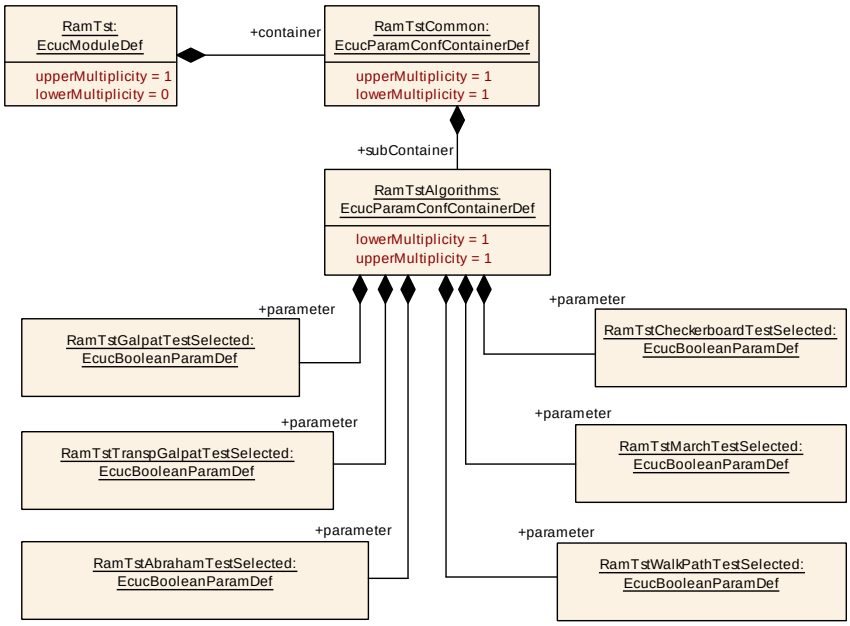


Figure 10.3: RamTstAlgorithms

10.2.6 RamTstConfigParams

[ECUC_RamTst_00066] Definition of EcucParamConfContainerDef RamTstConfigParams ⌈

Container Name	RamTstConfigParams
Parent Container	RamTstCommon
Upstream mapping defined	–
Description	This container specifies configuration parameters which are set at pre-compile or link time.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
RamTstDefaultAlgParamsId	1	[ECUC_RamTst_00181]
RamTstMinNumberOfTestedCells	1	[ECUC_RamTst_00154]
RamTstNumberOfAlgParamSets	1	[ECUC_RamTst_00180]

▽

△

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
RamTstTestCompletedNotification	1	[ECUC_RamTst_00138]
RamTstTestErrorNotification	1	[ECUC_RamTst_00139]

Included Containers		
Container Name	**Multiplicity**	**Description**
RamTstAlgParams	1..255	This container holds parameters for configuring an algorithm. For each algorithm selected in the RamTst_Algorithms container there can be one or more RamTstAlgParams containers. The multiplicity of the included container RamTstBlockParams depends on the number of separate blocks of RAM which are defined for the particular test configuration.

⌋

[ECUC_RamTst_00181] Definition of EcucIntegerParamDef RamTstDefaultAlgParamsId ⌈

Parameter Name	RamTstDefaultAlgParamsId
Parent Container	RamTstConfigParams
Upstream mapping defined	–
Description	This is the identifier for the default "RamTstAlgParams" valid after the "RamTst_Init(..)" function. It is the initial value for a RAM variable which could be changed by the function "RamTst_SelectAlgParams".
Multiplicity	1
Type	EcucIntegerParamDef
Range	1 .. 255
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00154] Definition of EcucIntegerParamDef RamTstMinNumberOfTestedCells ⌈

Parameter Name	RamTstMinNumberOfTestedCells
Parent Container	RamTstConfigParams
Upstream mapping defined	–
Description	Minimum number of tested cells for one cycle of a background test, as defined by implementer.
Multiplicity	1
Type	EcucIntegerParamDef
Range	1 .. 4294967295
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00180] Definition of EcucIntegerParamDef RamTstNumberOfAlgParamSets ⌈

Parameter Name	RamTstNumberOfAlgParamSets
Parent Container	RamTstConfigParams
Upstream mapping defined	–
Description	Number of configured parameter sets for the available test algorithms. calculation Formula = count of the container RamTst_AlgParams Dependency: "RamTstNumberOfAlgParamSets" is derived by the count of "RamTstAlg Params" which is part of the same subContainer and has a multiplicity of 1 to 255.
Multiplicity	1
Type	EcucIntegerParamDef
Range	1 .. 255
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00138] Definition of EcucFunctionNameDef RamTstTestCompletedNotification ⌈

Parameter Name	RamTstTestCompletedNotification
Parent Container	RamTstConfigParams
Upstream mapping defined	–
Description	This function will be called from a background test after finishing the RAM test without an error.
Multiplicity	1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00139] Definition of EcucFunctionNameDef RamTstTestErrorNotification ⌈

Parameter Name	RamTstTestErrorNotification
Parent Container	RamTstConfigParams
Upstream mapping defined	–
Description	This function will be called from a background test if an error occurs during the RAM test.
Multiplicity	1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

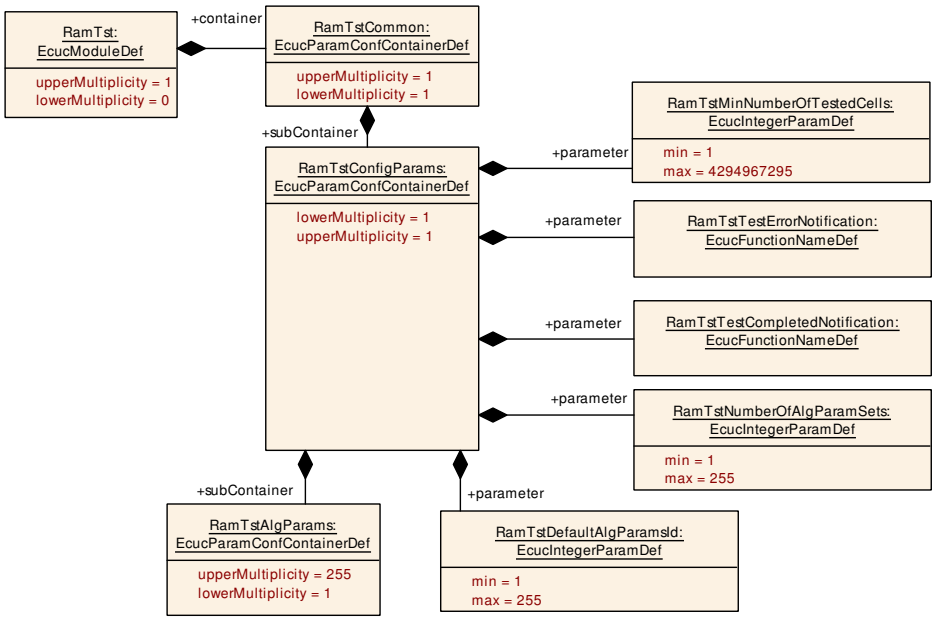


Figure 10.4: RamTstConfigParams

10.2.7 RamTstAlgParams

[ECUC_RamTst_00090] Definition of EcucParamConfContainerDef RamTstAlg Params ⌈

Container Name	RamTstAlgParams
Parent Container	RamTstConfigParams
Upstream mapping defined	–
Description	This container holds parameters for configuring an algorithm. For each algorithm selected in the RamTst_Algorithms container there can be one or more RamTstAlg Params containers. The multiplicity of the included container RamTstBlockParams depends on the number of separate blocks of RAM which are defined for the particular test configuration.
Multiplicity	1..255
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
RamTstAlgorithm	1	[ECUC_RamTst_00178]
RamTstAlgParamsId	1	[ECUC_RamTst_00179]
RamTstExtNumberOfTestedCells	1	[ECUC_RamTst_00152]
RamTstMaxNumberOfTestedCells	1	[ECUC_RamTst_00153]
RamTstNumberOfBlocks	1	[ECUC_RamTst_00141]
RamTstNumberOfTestedCells	1	[ECUC_RamTst_00142]
RamTstBlockParamsEcucPartitionRef	0..1	[ECUC_RamTst_00191]

Included Containers		
Container Name	**Multiplicity**	**Description**
RamTstBlockParams	1..65535	This container holds the description for one block of RAM. For each RAM block to be tested by a given algorithm, there is one container which describes the block. Multiple instances of this container are included in each container RamTst_AlgParams.

⌋

[ECUC_RamTst_00178] Definition of EcucEnumerationParamDef RamTstAlgorithm ⌈

Parameter Name	RamTstAlgorithm	
Parent Container	RamTstAlgParams	
Upstream mapping defined	–	
Description	This is the algorithm for which this RamTstAlgParams set is defined. Note that the same algorithm can be used in more than one RamTstAlgParams. Constraint: Only the algorithms selected by RamTstCommon/ RamTstAlgorithms can be used.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	RAMTST_ABRAHAM_TEST	Abraham test algorithm
	RAMTST_CHECKERBOARD_TEST	Checkerboard test algorithm
	RAMTST_GALPAT_TEST	Galpat test algorithm
	RAMTST_MARCH_TEST	March test algorithm
	RAMTST_TRANSP_GALPAT_TEST	Transparent Galpat test algorithm
	RAMTST_WALK_PATH_TEST	Walk path test algorithm
Pre-Compile only	true	

⌋

[ECUC_RamTst_00179] Definition of EcucIntegerParamDef RamTstAlgParamsId ⌈

Parameter Name	RamTstAlgParamsId	
Parent Container	RamTstAlgParams	
Upstream mapping defined	–	
Description	This is the identifier by which this RamTstAlgParams set can be selected.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_RamTst_00152] Definition of EcucIntegerParamDef RamTstExtNumberOfTestedCells ⌈

Parameter Name	RamTstExtNumberOfTestedCells	
Parent Container	RamTstAlgParams	
Upstream mapping defined	–	
Description	This is the absolute maximum value for the number of cells that NUMBER_OF_TESTED_CELLS and MAX_NUMBER_OF_TESTED_CELLS can be.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_RamTst_00153] Definition of EcucIntegerParamDef RamTstMaxNumberOfTestedCells ⌈

Parameter Name	RamTstMaxNumberOfTestedCells	
Parent Container	RamTstAlgParams	
Upstream mapping defined	–	
Description	This is the maximum value for the number of cells that can be tested in one cycle of a background test.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_RamTst_00141] Definition of EcucIntegerParamDef RamTstNumberOfBlocks ⌈

Parameter Name	RamTstNumberOfBlocks	
Parent Container	RamTstAlgParams	
Upstream mapping defined	–	
Description	Number of RAM blocks configured using the container "RamTst_BlockParams" calculationFormula = Count of RamTstBlockParams contained in this RamTstAlg Params. Dependency: "RamTstNumberOfBlocks" is derived by the count of the number of "Ram TstBlockParams" containers which are part of the same subcontainer and have a multiplicity of 0..65535.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_RamTst_00142] Definition of EcucIntegerParamDef RamTstNumberOfTestedCells ⌈

Parameter Name	RamTstNumberOfTestedCells
Parent Container	RamTstAlgParams
Upstream mapping defined	–
Description	This is the initial value for a RAM variable, which can be changed by the function "RamTst_ChangeNumberOfTestedCells"
Multiplicity	1
Type	EcucIntegerParamDef
Range	1 .. 4294967295
Default value	–
Pre-Compile only	true

⌋

[ECUC_RamTst_00191] Definition of EcucReferenceDef RamTstBlockParams EcucPartitionRef ⌈

Parameter Name	RamTstBlockParamsEcucPartitionRef
Parent Container	RamTstAlgParams
Upstream mapping defined	–
Description	Maps the RAM test block parameter configuration of an individual RAM block to zero or one ECUC partitions. The ECUC partition referenced is a subset of the ECUC partitions where the RamTst driver is mapped to. The according test shall run in the referenced ECUC partition.
Multiplicity	0..1
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

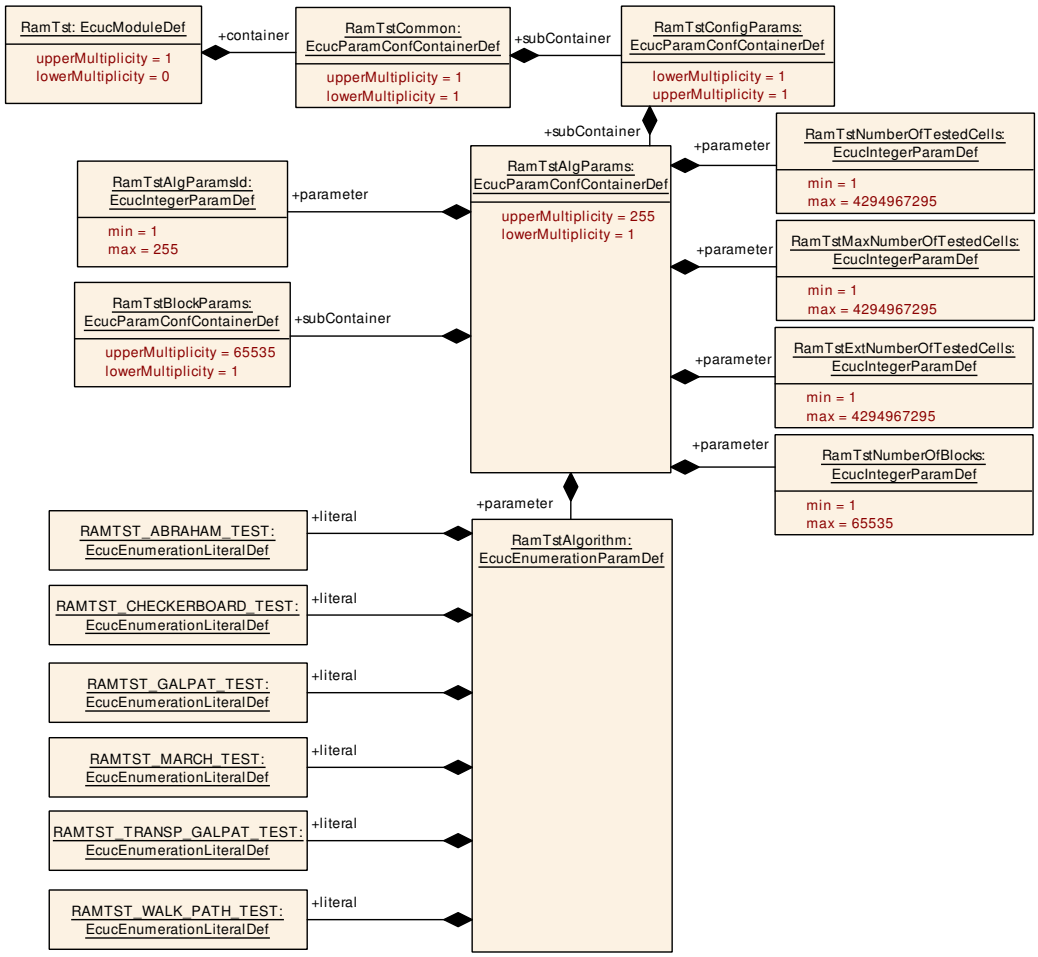


Figure 10.5: RamTstAlgParams

[SWS_RamTst_CONSTR_01016] ⌈The ECUC partitions referenced by `RamTstBlockParamsEcucPartitionRef` shall be a subset of the ECUC partitions referenced by `RamTstEcucPartitionRef`.⌋

10.2.8 RamTstBlockParams

[ECUC_RamTst_00091] Definition of EcucParamConfContainerDef RamTstBlockParams ⌈

Container Name	RamTstBlockParams
Parent Container	RamTstAlgParams
Upstream mapping defined	–
Description	This container holds the description for one block of RAM. For each RAM block to be tested by a given algorithm, there is one container which describes the block. Multiple instances of this container are included in each container RamTst_AlgParams.
Multiplicity	1..65535

▽

△

Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
RamTstBlockId	1	[ECUC_RamTst_00143]
RamTstEndAddress	1	[ECUC_RamTst_00144]
RamTstFillPattern	1	[ECUC_RamTst_00176]
RamTstStartAddress	1	[ECUC_RamTst_00145]
RamTstTestPolicy	1	[ECUC_RamTst_00177]

No Included Containers

⌋

[ECUC_RamTst_00143] Definition of EcucIntegerParamDef RamTstBlockId ⌈

Parameter Name	RamTstBlockId	
Parent Container	RamTstBlockParams	
Upstream mapping defined	–	
Description	ID of the RAM block	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_RamTst_00144] Definition of EcucIntegerParamDef RamTstEndAddress ⌈

Parameter Name	RamTstEndAddress	
Parent Container	RamTstBlockParams	
Upstream mapping defined	–	
Description	End Address of the RAM block. Constraint: It must be larger than the RamTstStartAddress.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_RamTst_00176] Definition of EcucIntegerParamDef RamTstFillPattern ⌈

Parameter Name	RamTstFillPattern	
Parent Container	RamTstBlockParams	
Upstream mapping defined	–	
Description	Pattern to be filled into each memory cell after destructive test of this block.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_RamTst_00145] Definition of EcucIntegerParamDef RamTstStartAddress ⌈

Parameter Name	RamTstStartAddress	
Parent Container	RamTstBlockParams	
Upstream mapping defined	–	
Description	Start Address of the RAM block. Constraint: It must be smaller than the RamTstEndAddress.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_RamTst_00177] Definition of EcucEnumerationParamDef RamTstTestPolicy ⌈

Parameter Name	RamTstTestPolicy	
Parent Container	RamTstBlockParams	
Upstream mapping defined	–	
Description	Policy regarding destruction or non-destruction of memory content.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	RAMTEST_DESTRUCTIVE	RAM test does not restore memory content.
	RAMTEST_NON_DESTRUCTIVE	RAM test restores memory content.
Pre-Compile only	true	

⌋

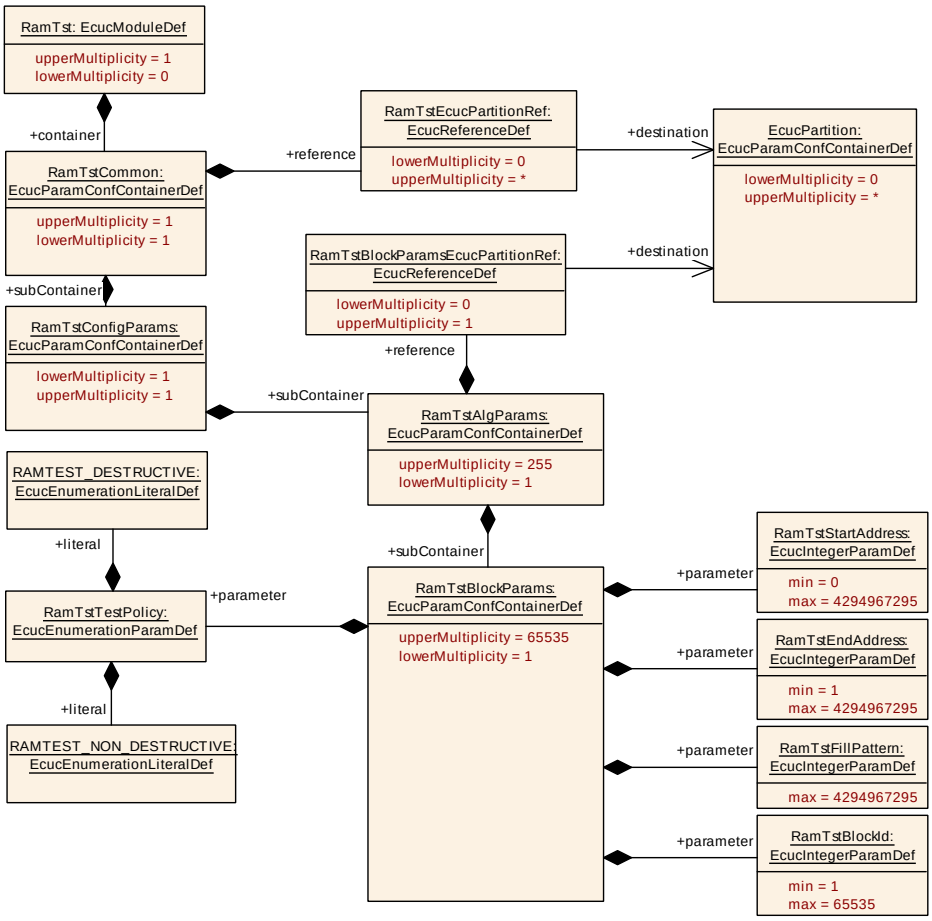


Figure 10.6: RamTstBlockParams

[SWS_RamTst_CONSTR_01017] ⌈If `RamTstEcucPartitionRef` references one or more ECUC partitions, `RamTstBlockParamsEcucPartitionRef` shall have a multiplicity of one and reference one of these ECUC partitions as well.⌋

10.3 Published Information

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

10.3.1 RamTstPublishedInformation

[ECUC_RamTst_00186] Definition of EcucParamConfContainerDef RamTstPublishedInformation ⌈

Container Name	RamTstPublishedInformation
Parent Container	RamTst
Upstream mapping defined	–
Description	Container holding all RamTst specific published information parameter.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
RamTstCellSize	1	[ECUC_RamTst_00187]

No Included Containers

⌋

[ECUC_RamTst_00187] Definition of EcucIntegerParamDef RamTstCellSize ⌈

Parameter Name	RamTstCellSize	
Parent Container	RamTstPublishedInformation	
Upstream mapping defined	–	
Description	Size of RAM cells (in bits) which can be tested individually by the given implementation.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 64	
Default value	–	
Pre-Compile only	true	

⌋

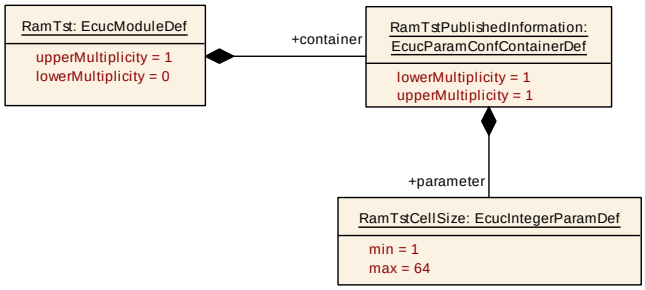


Figure 10.7: RamTstPublishedInformation

10.4 Implementation Specific Information and Parameters

[SWS_RamTst_00081]

Upstream requirements: SRS_BSW_00402

⌈The implementer shall provide measured or calculated runtime information in the documentation of the module for each algorithm implementation. The information is to be

presented as shown in the following table, specifying whether the parameters are measured or calculated.⌋

Microcontroller	Frequency	RamCellSize [bit]:	No of cells/ cycle	Average Runtime	Interrupt lock time	Internal used RAM
–	–	–	–	–	–	–

[SWS_RamTst_00205] ⌈If an implementation of the RAM Test module supports vendor specific test algorithms or other additional configuration parameters, the implementer shall provide a formal vendor-specific definition of these parameters including their documentation (as part of the BSW Module Description).⌋

A Not applicable requirements

[SWS_RamTst_NA_00999] Requirements non-applicable to this Specification

Upstream requirements: SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00336, SRS_BSW_00375, SRS_BSW_00383, SRS_BSW_00386, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00404, SRS_BSW_00405, 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_SPAL_12063, SRS_SPAL_12064, SRS_SPAL_12067, SRS_SPAL_12068, SRS_SPAL_12069, SRS_SPAL_12075, SRS_SPAL_12125, SRS_SPAL_12267, SRS_SPAL_12461, SRS_SPAL_12462, SRS_SPAL_12463, SRS_SPAL_12078, SRS_SPAL_12092, SRS_SPAL_12265

⌈These requirements are not applicable to this specification.⌋

B Change history of AUTOSAR traceable items

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

B.1 Traceable item history of this document according to AUTOSAR Release R26-11

B.1.1 Added Specification Items in R26-11

none

B.1.2 Changed Specification Items in R26-11

Number	Heading
[ECUC_RamTst_00065]	Definition of EcucParamConfContainerDef RamTstAlgorithms
[ECUC_RamTst_00066]	Definition of EcucParamConfContainerDef RamTstConfigParams
[ECUC_RamTst_00070]	Definition of EcucParamConfContainerDef RamTstCommon
[ECUC_RamTst_00090]	Definition of EcucParamConfContainerDef RamTstAlgParams
[ECUC_RamTst_00091]	Definition of EcucParamConfContainerDef RamTstBlockParams
[ECUC_RamTst_00118]	Definition of EcucBooleanParamDef RamTstChangeNumOfTestedCellsApi
[ECUC_RamTst_00120]	Definition of EcucBooleanParamDef RamTstAllowApi
[ECUC_RamTst_00121]	Definition of EcucBooleanParamDef RamTstDevErrorDetect
[ECUC_RamTst_00122]	Definition of EcucBooleanParamDef RamTstGetExecutionStatusApi
[ECUC_RamTst_00123]	Definition of EcucBooleanParamDef RamTstGetNumberOfTestedCellsApi
[ECUC_RamTst_00124]	Definition of EcucBooleanParamDef RamTstGetTestAlgorithmApi
[ECUC_RamTst_00125]	Definition of EcucBooleanParamDef RamTstGetTestResultApi
[ECUC_RamTst_00126]	Definition of EcucBooleanParamDef RamTstGetTestResultPerBlockApi



△

Number	Heading
[ECUC_RamTst_00127]	Definition of EcucBooleanParamDef RamTstStopApi
[ECUC_RamTst_00129]	Definition of EcucBooleanParamDef RamTstAbrahamTestSelected
[ECUC_RamTst_00130]	Definition of EcucBooleanParamDef RamTstCheckerboardTestSelected
[ECUC_RamTst_00132]	Definition of EcucBooleanParamDef RamTstGalpatTestSelected
[ECUC_RamTst_00133]	Definition of EcucBooleanParamDef RamTstMarchTestSelected
[ECUC_RamTst_00134]	Definition of EcucBooleanParamDef RamTstTranspGalpatTestSelected
[ECUC_RamTst_00135]	Definition of EcucBooleanParamDef RamTstWalkPathTestSelected
[ECUC_RamTst_00138]	Definition of EcucFunctionNameDef RamTstTestCompletedNotification
[ECUC_RamTst_00139]	Definition of EcucFunctionNameDef RamTstTestErrorNotification
[ECUC_RamTst_00141]	Definition of EcucIntegerParamDef RamTstNumberOfBlocks
[ECUC_RamTst_00142]	Definition of EcucIntegerParamDef RamTstNumberOfTestedCells
[ECUC_RamTst_00143]	Definition of EcucIntegerParamDef RamTstBlockId
[ECUC_RamTst_00144]	Definition of EcucIntegerParamDef RamTstEndAddress
[ECUC_RamTst_00145]	Definition of EcucIntegerParamDef RamTstStartAddress
[ECUC_RamTst_00150]	Definition of EcucModuleDef RamTst
[ECUC_RamTst_00152]	Definition of EcucIntegerParamDef RamTstExtNumberOfTestedCells
[ECUC_RamTst_00153]	Definition of EcucIntegerParamDef RamTstMaxNumberOfTestedCells
[ECUC_RamTst_00154]	Definition of EcucIntegerParamDef RamTstMinNumberOfTestedCells
[ECUC_RamTst_00155]	Definition of EcucBooleanParamDef RamTstResumeApi
[ECUC_RamTst_00156]	Definition of EcucBooleanParamDef RamTstSuspendApi
[ECUC_RamTst_00176]	Definition of EcucIntegerParamDef RamTstFillPattern

▽

△

Number	Heading
[ECUC_RamTst_00177]	Definition of EcucEnumerationParamDef RamTstTestPolicy
[ECUC_RamTst_00178]	Definition of EcucEnumerationParamDef RamTstAlgorithm
[ECUC_RamTst_00179]	Definition of EcucIntegerParamDef RamTstAlgParamsId
[ECUC_RamTst_00180]	Definition of EcucIntegerParamDef RamTstNumberOfAlgParamSets
[ECUC_RamTst_00181]	Definition of EcucIntegerParamDef RamTstDefaultAlgParamsId
[ECUC_RamTst_00182]	Definition of EcucBooleanParamDef RamTstSelectAlgParamsApi
[ECUC_RamTst_00183]	Definition of EcucBooleanParamDef RamTstGetAlgParamsApi
[ECUC_RamTst_00184]	Definition of EcucBooleanParamDef RamTstRunFullTestApi
[ECUC_RamTst_00185]	Definition of EcucBooleanParamDef RamTstRunPartialTestApi
[ECUC_RamTst_00186]	Definition of EcucParamConfContainerDef RamTstPublishedInformation
[ECUC_RamTst_00187]	Definition of EcucIntegerParamDef RamTstCellSize
[ECUC_RamTst_00188]	Definition of EcucParamConfContainerDef RamTstDemEventParameterRefs
[ECUC_RamTst_00190]	Definition of EcucReferenceDef RamTstEcucPartitionRef
[ECUC_RamTst_00191]	Definition of EcucReferenceDef RamTstBlockParamsEcucPartitionRef
[ECUC_RamTst_00192]	Definition of EcucReferenceDef RAMTST_RUNFL_RAM_FAILURE
[ECUC_RamTst_00193]	Definition of EcucReferenceDef RAMTST_PART_RAM_FAILURE
[ECUC_RamTst_00194]	Definition of EcucReferenceDef RAMTST_MAIN_RAM_FAILURE
[SWS_RamTst_00098]	Definition of imported datatypes of module RamTst
[SWS_RamTst_01002]	RAM failure during test.
[SWS_RamTst_01005]	RAM failure during test.
[SWS_RamTst_01008]	RAM failure during test.

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

B.1.3 Deleted Specification Items in R26-11

Number	Heading
[ECUC_RamTst_00128]	Definition of EcucBooleanParamDef RamTstVersionInfoApi
[SWS_RamTst_00109]	Definition of API function RamTst_GetVersionInfo
[SWS_RamTst_00222]	

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

B.1.4 Added Constraints in R26-11

none

B.1.5 Changed Constraints in R26-11

none

B.1.6 Deleted Constraints in R26-11

none

B.2 Traceable item history of this document according to AUTOSAR Release R25-11

B.2.1 Added Specification Items in R25-11

none

B.2.2 Changed Specification Items in R25-11

Number	Heading
[SWS_RamTst_01002]	RAM failure during test.
[SWS_RamTst_01005]	RAM failure during test.

▽

△

Number	Heading
[SWS_RamTst_01008]	RAM failure during test.

Table B.3: Changed Specification Items in R25-11

B.2.3 Deleted Specification Items in R25-11

none

B.2.4 Added Constraints in R25-11

none

B.2.5 Changed Constraints in R25-11

none

B.2.6 Deleted Constraints in R25-11

none