Document Title	Specification of RAM Test
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Identification No	076
Document Classification	Standard

Document Version	1.2.2
Document Status	Final
Part of Release	3.1
Revision	0001

Document Change History

Date	Version	Changed by	Change Description
23.06.2008	1.2.2	AUTOSAR Administration	Legal disclaimer revised
22.01.2008	1.2.1	AUTOSAR Administration	• Correction of figures in Chapter 1 and Chapter 9.
11.12.2007	1.2.0	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
24.01.2007	1.1.1	AUTOSAR Administration	• “Advice for users” revised • “Revision Information” added
15.12.2006	1.1.0	AUTOSAR Administration	• File include structure updated • “Modified Hamming code” test removed • RamTst_Stop() & RamTst_Continue() changed to “asynchronous” • Dem API updated • Configuration description corrected • descriptions optimized • Legal disclaimer revised
18.05.2006	1.0.0	AUTOSAR Administration	Initial release

Page left intentionally blank

Disclaimer

Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).
Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard.
Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction and Functional Overview

This specification specifies the functionality, API and configuration of the AUTOSAR Basic Software module 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.

There are several RAM Test algorithms available. These RAM Test algorithms are chosen based upon the IEC 61508 specification. The different RAM Test algorithms are divided into 3 groups of diagnostic coverage rates:

Group 1 (Low): Diagnostic coverage rate = <60%
Group 2 (Medium): Diagnostic coverage rate = 60% – 90%
Group 3 (High): Diagnostic coverage rate = 90% – ≤99%.

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 are then selected at compile time. At run time, the application software may choose between the compiled algorithms.

The RAM Test main function may be called asynchronously by the user or may be called in a cyclic manner by an OS task or other cyclic calling method. The user may start and stop the test, and get status reports. Development errors are reported the Development Error Tracer (DET) and production errors are reported to the Diagnostic Event Manager (DEM).

The RamTst module consists of a MainFunction(), some configuration and status API's (Application Programming Interface), and several configuration containers.

TEST FUNCTION API's	***DEFINITION***
INIT	Prepare resources for testing as necessary. Initialize MainFunction() as necessary. Proceed to "test stopped" state after initialization is complete.
DEINIT	Reset all used registers to reset values, and release all used resources.
ALLOW	Permit the MainFunction() to perform testing at its next scheduled call.
STOP	Prohibit the MainFunction() from performing tests at its next scheduled call. When STOP is called, testing stops after the current atomic sequence. Test status is retained, but test parameters (block number, loop count, etc.) are discarded. ALLOW must be called after STOP to permit the MainFunction() to test when called by the scheduler.
SUSPEND	Temporarily prohibit the MainFunction() from performing tests at its next scheduled call. When SUSPEND is called, testing stops after the current atomic sequence. Test status and test parameters are retained.
RESUME	Permits the Mainfunction() to continue testing at the point where it was suspended, at its next scheduled call. Testing continues according to the saved test parameters.
RUN FULL TEST	Test the entire RAM space without interruption. STOP must be called prior to calling this API.
RUN PARTIAL TEST	Test the portion of the RAM defined by the API. STOP or SUSPEND must be called prior to calling this API.

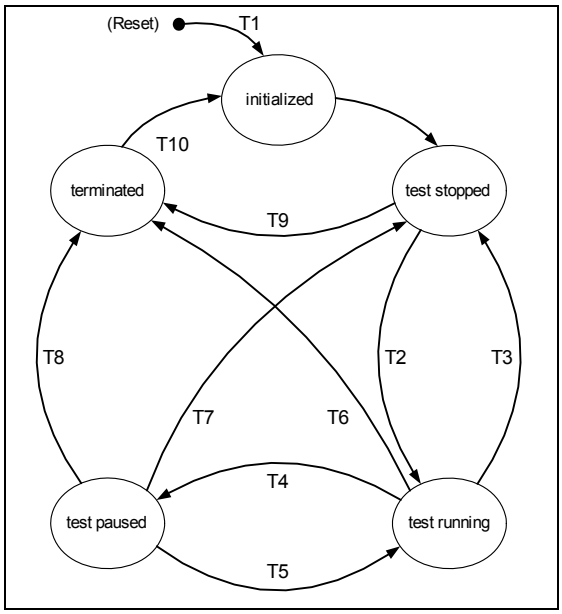
TEST PARAMETER AND FEEDBACK API's
VERSION_INFO
GET_EXECUTION_STATUS
GET_TEST_RESULT
GET_TEST_RESULT_PER_BLOCK
GET_TEST_ALGORITHM
GET_NUMBER_OF_TESTED_CELLS
CHANGE_TEST_ALGORITHM
CHANGE_NUMBER_OF_TESTED_CELLS
CHANGE_MAX_NUMBER_OF_TESTED_CELLS
SELECT_TEST_BLOCK

MainFunction() is the actual testing function of RamTst. MainFunction() may be run in background or foreground mode.

- In background mode, 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 algorithm configuration. This complete test is split up over many scheduled tasks.

- In foreground mode, MainFunction() is called once, and is not interruptible. It tests with one algorithm over the memory space defined by the algorithm configuration.

The state chart below shows the various states of MainFunction() (not of RamTst). While MainFunction() is running non-interrupted in the foreground, it is in “test running” state. While MainFunction() is running as a background task and is maintaining test and test result status, it is also in the “test running” state. It does not reach “test stopped” state until a full test is completed over the memory space defined in the current algorithm.



Event	Event Trigger
T1	API: INIT
T2	API: RUN_FULL_TEST API: RUN_PARTIAL_TEST API: ALLOW (or start by scheduler)
T3	API: STOP (or test completely finished)
T4	API: SUSPEND (or end of RUN_PARTIAL_TEST) (or error during test)
T5	API: RESUME API: RUN_PARTIAL_TEST
T6	API: DEINIT
T7	API: STOP
T8	API: DEINIT
T9	API: DEINIT
T10	API: INIT

Note: The state “test running” does not necessarily mean that testing is continuously being performed. For foreground testing, it does mean that MainFunction() is actively testing. For background testing, it only means that MainFunction() is permitted to test a small portion of the RAM when it is called periodically by the scheduler.

All API’s and configuration variables are fully defined elsewhere within this document.

The following table shows what API’s 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 (except that no DET errors are reported in the “Initialized” State).

API: Application Programming Interface

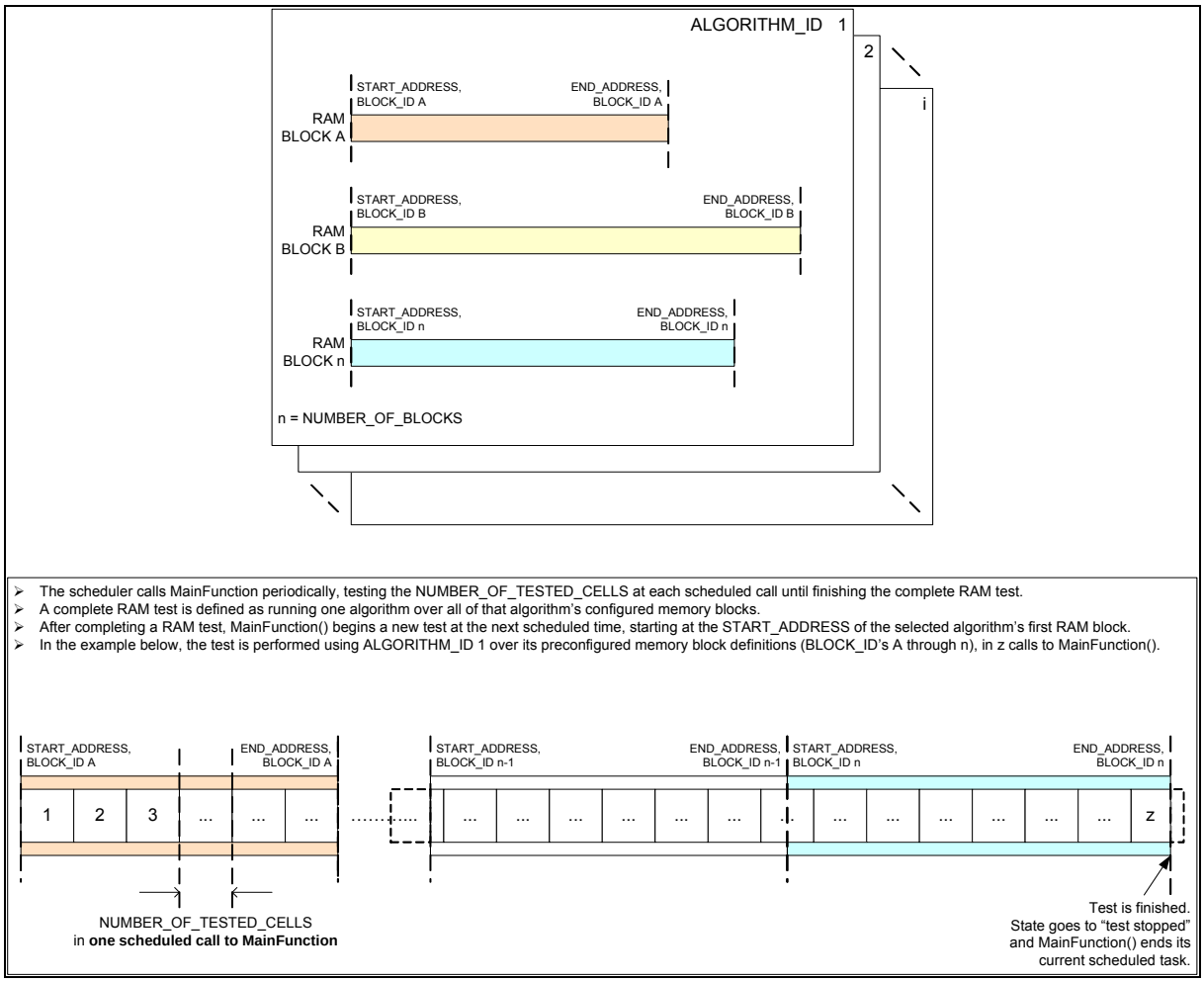
	API allowable in this State?				
API’s which cause a change of state in the state chart	**Initialized**	**Test Stopped**	**Test Running**	**Test Paused**	**Test Terminated**
INIT	N	N	N	N	Y
RUN_FULL_TEST [1]	N	Y	N	N	N
RUN_PARTIAL_TEST [2]	N	Y	N	Y	N
SUSPEND [3]	N	N	Y	N	N
RESUME	N	N	N	Y	N
STOP	N	N	Y	Y	N
ALLOW [4]	N	Y	N	N	N

DEINIT	N	Y	N	Y	N

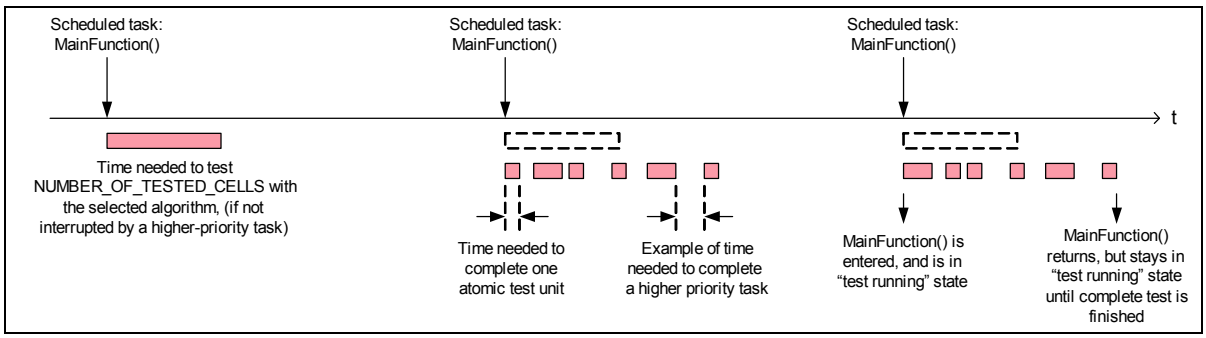
	API allowable in this State?				
API's which do not cause a change of state	**Initialized**	**Test Stopped**	**Test Run-ning**	**Test Paused**	**Test Ter-minated**
VERSION_INFO	N	Y	Y	Y	N
GET_EXECUTION_STATUS	N	Y	Y	Y	N
GET_TEST_RESULT	N	Y	Y	Y	N
GET_TEST_RESULT_PER_BLOCK	N	Y	Y	Y	N
GET_TEST_ALGORITHM	N	Y	Y	Y	N
GET_NUMBER_OF_TESTED_CELLS	N	Y	Y	Y	N
CHANGE_TEST_ALGORITHM [5]	N	Y	N	N	N
CHANGE_NUMBER_OF_TESTED_CELLS [6]	N	Y	N	Y	N
CHANGE_MAX_NUMBER_OF_TESTED_CELLS	N	Y	N	Y	N
SELECT_TEST_BLOCK [5]	N	Y	N	N	N

NOTES:
[1] RUN_FULL_TEST should run uninterrupted (except for watchdog servicing if necessary). It may be called only while MainFunction() is in "test stopped" state.
[2] RUN_PARTIAL_TEST should run uninterrupted. It may be called only while MainFunction() is in "test stopped" or "test paused" states.
[3] SUSPEND causes a state change to "test paused" at the end of the current MainFunction() atomic sequence if MainFunction() is actively testing.
[4] ALLOW is called to permit the MainFunction() to test when called. ALLOW does not initiate any test itself.
[5] STOP must first be called before changing configuration parameters CHANGE_TEST_ALGORITHM and SELECT_TEST_BLOCK.
[6] CHANGE_NUMBER_OF_TESTED_CELLS operates at the end of the current MainFunction() atomic sequence if MainFunction() is actively testing. For a foreground test, CHANGE_NUMBER_OF_TESTED_CELLS must be called before starting the test (RUN_FULL_TEST or RUN_PARTIAL_TEST).

The following figure shows how blocks are configured for an algorithm, and how MainFunction() then tests the memory cells for each block.



The following figure shows how MainFunction() is called by the scheduler, and how it can be interrupted between atomic pieces by higher priority tasks (when MainFunction() is run as a background task).



NUMBER_OF_TESTED_CELLS
The NUMBER_OF_TESTED_CELLS 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 algorithm. NUMBER_OF_TESTED_CELLS can be changed during runtime using the API CHANGE_NUMBER_OF_TESTED_CELLS. 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.

For background RAM testing, NUMBER_OF_TESTED_CELLS is not allowed to exceed MAX_NUMBER_OF_TESTED_CELLS. For foreground RAM testing only, NUMBER_OF_TESTED_CELLS is set equal to EXT_NUMBER_OF_TESTED_CELLS. Both circumstances are discussed below.

The absolute maximum size of NUMBER_OF_TESTED_CELLS for a given algorithm 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 in the algorithm. The integrator sets EXT_NUMBER_OF_TESTED_CELLS to this absolute maximum value (pre-compile or link) in the RamTstAlgParams container. EXT_NUMBER_OF_TESTED_CELLS is not changeable during run time.

The integrator configures (pre-compile or link) the MAX_NUMBER_OF_TESTED_CELLS for each algorithm in the RamTstAlgParams container. The integrator must carefully select MAX_NUMBER_OF_TESTED_CELLS such that it puts an upper limit on the run time of MainFunction() in a background task according to the system needs for throughput, as calculated for the scheduler. In no case should MAX_NUMBER_OF_TESTED_CELLS be set to a value greater than the extended number of cells EXT_NUMBER_OF_TESTED_CELLS (the absolute maximum size defined by the implementer).

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

Any time that CHANGE_NUMBER_OF_TESTED_CELLS is called, MainFunction() must verify that MIN_NUMBER_OF_TESTED_CELLS ≤ NUMBER_OF_TESTED_CELLS ≤ MAX_NUMBER_OF_TESTED_CELLS.

Before the foreground tests RUN_FULL_TEST or RUN_PARTIAL_TEST, CHANGE_NUMBER_OF_TESTED_CELLS is used to set NUMBER_OF_TESTED_CELLS = EXT_NUMBER_OF_TESTED_CELLS. This al-

lows these tests to run to completion without interruption. At the end of these tests, NUMBER_OF_TESTED_CELLS is restored.

When MainFunction() reaches the end of a block, it should continue testing at the beginning of the next block until the NUMBER_OF_TESTED_CELLS has been reached in that scheduled task. This is so that the foreground tests use the same MainFunction() as does the background task, because if the MainFunction() concluded its task at a block border then the foreground tests would never be able to run to completion.

When NUMBER_OF_TESTED_CELLS has been reached, the scheduled task is finished.

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

MainFunction()
MainFunction() is called with no parameters. The MainFunction() is the only means provided in the RamTst module for performing a RAM test.

A **background** test is performed by the scheduler periodically calling the MainFunction() to test a NUMBER_OF_TESTED_CELLS of memory using the selected algorithm, until the entire defined area of RAM is tested. This MainFunction() can be interrupted at the end of each atomic sequence during a scheduled task.
A **foreground** test is generally performed uninterrupted by calling the appropriate API, which configures MainFunction() and then calls MainFunction().

MainFunction():

- Is made up of atomic pieces which are as small as practically possible. Generally, this means that a MIN_NUMBER_OF_TESTED_CELLS is completely tested during the atomic sequence.
- At the end of each atomic piece, the flags are checked to see if an OS task has changed any parameter of the state chart, and to respond to question-type API's.
- If NUMBER_OF_TESTED_CELLS is changed, it is checked for validity.
  - **If** NUMBER_OF_TESTED_CELLS > MAX_NUMBER_OF_TESTED_CELLS, **then** NUMBER_OF_TESTED_CELLS is set equal to MAX_NUMBER_OF_TESTED_CELLS. This is a DET error.
  - **If** NUMBER_OF_TESTED_CELLS < MIN_NUMBER_OF_TESTED_CELLS, **then** NUMBER_OF_TESTED_CELLS is set equal to MIN_NUMBER_OF_TESTED_CELLS. This is a DET error.
- 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.
- 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 NUMBER_OF_TESTED_CELLS is reached, MainFunction() ends testing for that scheduled task, and starts the next scheduled task at the next (saved) address.
- When the end of a block is reached during a scheduled task, MainFunction() continues testing at the beginning of the next block, and continues until NUMBER_OF_TESTED_CELLS is reached. (Note: The atomic test sequence should be careful to take into account any issues regarding crossing into the next block.)
- When all blocks are fully tested, MainFunction() goes to "test stopped" state and issues a notification.

- If there is an error during testing, MainFunction() goes to “test paused” state and issues a notification.

RUN_FULL_TEST, RUN_PARTIAL_TEST

“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 algorithm is selected by calling the API CHANGE_TEST_ALGORITHM before calling the foreground test API.

RUN_FULL_TEST API:
The user calls RUN_FULL_TEST with no parameters. The test algorithm to run is chosen before calling this API. This test is normally used for a full RAM check at system startup or shutdown.

Sequence:
STOP
CHANGE_TEST_ALGORITHM to ChosenAlgorithm
RUN_FULL_TEST

RUN_FULL_TEST function:
- Save the value of NUMBER_OF_TESTED_CELLS.
- Call CHANGE_NUMBER_OF_TESTED_CELLS, setting NUMBER_OF_TESTED_CELLS equal to EXT_NUMBER_OF_TESTED_CELLS.
- Call MainFunction(). MainFunction() runs uninterrupted (except for watchdog service if necessary) over the memory space defined in the ChosenAlgorithm.
- After test finishes, restore NUMBER_OF_TESTED_CELLS.

MainFunction()
- Executes with ChosenAlgorithm
- Tests all memory blocks defined in ChosenAlgorithm.

RamTstAlgParams Container for ChosenAlgorithm
- ALGORITHM_ID = ChosenAlgorithm
- NUMBER_OF_TESTED_CELLS (changed from previous) = EXT_NUMBER_OF_TESTED_CELLS.
- (other parameters are configured to a predefined value).

RUN_PARTIAL_TEST API:
The user calls RUN_PARTIAL_TEST with one parameter. The parameter passed is which block to run the test on. This test is used primarily to check a specified memory section immediately before using that memory. This capability is to enable a system safety concept.

Sequence:
STOP or SUSPEND
CHANGE_TEST_ALGORITHM to ChosenAlgorithm
RUN_PARTIAL_TEST (ChosenBlock)

API function:
- Save the value of NUMBER_OF_TESTED_CELLS, and all test parameters if a test was currently allowed (current block under test, current test address, current status, etc.).
- Call SELECT_TEST_BLOCK to limit the test to the ChosenBlock.
- Call CHANGE_NUMBER_OF_TESTED_CELLS, to set NUMBER_OF_TESTED_CELLS equal to EXT_NUMBER_OF_TESTED_CELLS.
- Call MainFunction(). MainFunction() runs in a non-interruptible form over the memory space defined in the ChosenAlgorithm.

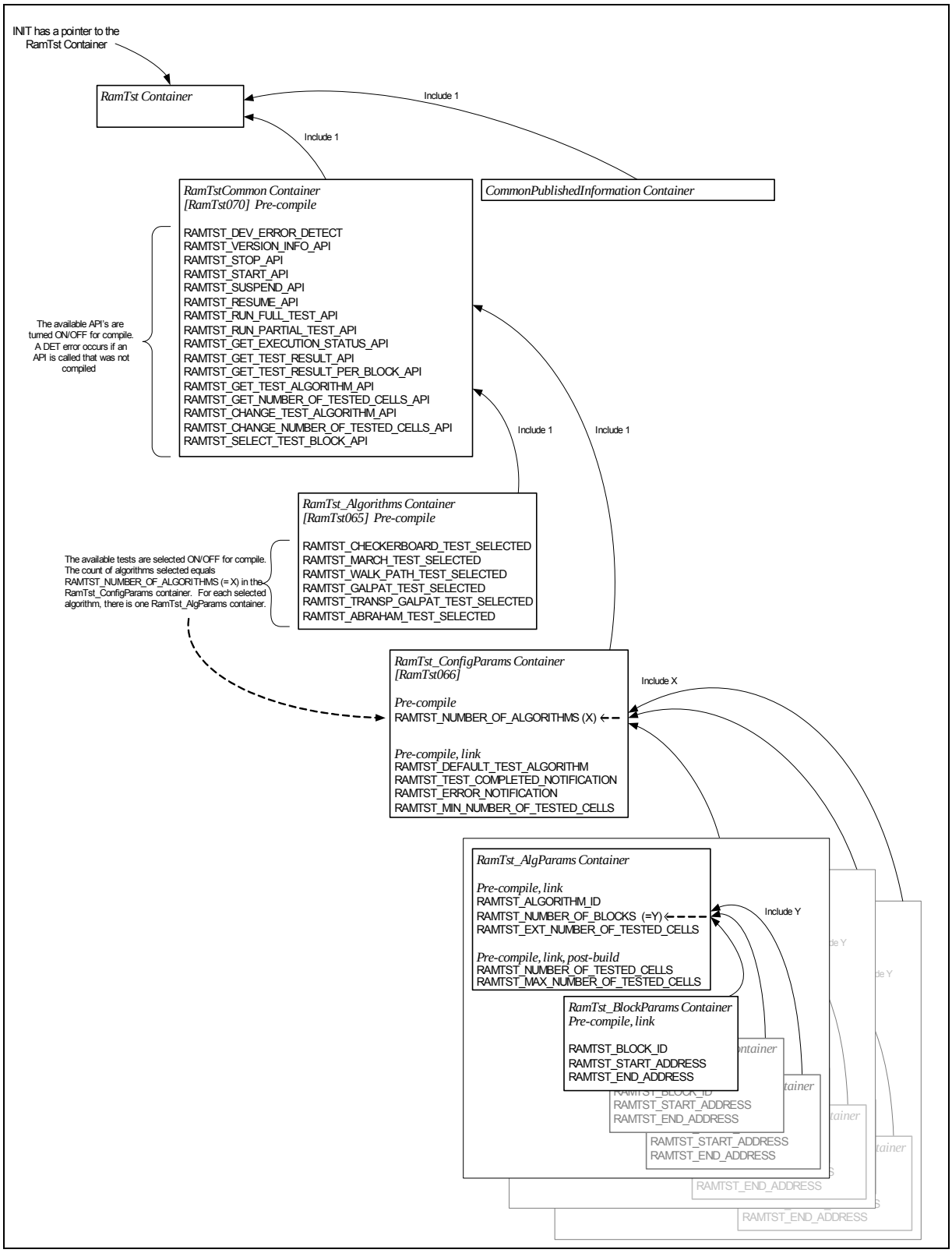
- After test finishes, restore NUMBER_OF_TESTED_CELLS, and any other test parameters that changed when RUN_PARTIAL_TEST was called, in order to resume testing where it was paused (if a testing had been allowed prior to the RUN_PARTIAL_TEST call).

MainFunction()

- Executes with ChosenAlgorithm
- Tests only the memory block selected by the ChosenBlock.

RamTstAlgParams Container for ChosenAlgorithm

- ALGORITHM_ID = ChosenAlgorithm
- NUMBER_OF_TESTED_CELLS (changed from previous) = EXT_NUMBER_OF_TESTED_CELLS.
- NUMBER_OF_BLOCKS = 1.
- (other parameters are configured to a predefined value).


INIT has a pointer to the RamTst Container
RamTst Container
Include 1
Include 1
RamTstCommon Container
[RamTst070] Pre-compile
CommonPublishedInformation Container
RAMTST_DEV_ERROR_DETECT
RAMTST_VERSION_INFO_API
RAMTST_STOP_API
RAMTST_START_API
RAMTST_SUSPEND_API
RAMTST_RESUME_API
RAMTST_RUN_FULL_TEST_API
RAMTST_RUN_PARTIAL_TEST_API
RAMTST_GET_EXECUTION_STATUS_API
RAMTST_GET_TEST_RESULT_API
RAMTST_GET_TEST_RESULT_PER_BLOCK_API
RAMTST_GET_TEST_ALGORITHM_API
RAMTST_GET_NUMBER_OF_TESTED_CELLS_API
RAMTST_CHANGE_TEST_ALGORITHM_API
RAMTST_CHANGE_NUMBER_OF_TESTED_CELLS_API
RAMTST_SELECT_TEST_BLOCK_API
The available API's are turned ON/OFF for compile. A DET error occurs if an API is called that was not compiled
Include 1
Include 1
RamTst_Algorithms Container
[RamTst065] Pre-compile
RAMTST_CHECKERBOARD_TEST_SELECTED
RAMTST_MARCH_TEST_SELECTED
RAMTST_WALK_PATH_TEST_SELECTED
RAMTST_GALPAT_TEST_SELECTED
RAMTST_TRANSP_GALPAT_TEST_SELECTED
RAMTST_ABRAHAM_TEST_SELECTED
The available tests are selected ON/OFF for compile. The count of algorithms selected equals RAMTST_NUMBER_OF_ALGORITHMS (= X) in the RamTst_ConfigParams container. For each selected algorithm, there is one RamTst_AlgParams container.
RamTst_ConfigParams Container
[RamTst066]
Pre-compile
RAMTST_NUMBER_OF_ALGORITHMS (X)
Pre-compile, link
RAMTST_DEFAULT_TEST_ALGORITHM
RAMTST_TEST_COMPLETED_NOTIFICATION
RAMTST_ERROR_NOTIFICATION
RAMTST_MIN_NUMBER_OF_TESTED_CELLS
Include X
RamTst_AlgParams Container
Pre-compile, link
RAMTST_ALGORITHM_ID
RAMTST_NUMBER_OF_BLOCKS (=Y)
RAMTST_EXT_NUMBER_OF_TESTED_CELLS
Pre-compile, link, post-build
RAMTST_NUMBER_OF_TESTED_CELLS
RAMTST_MAX_NUMBER_OF_TESTED_CELLS
Include Y
RamTst_BlockParams Container
Pre-compile, link
RAMTST_BLOCK_ID
RAMTST_START_ADDRESS
RAMTST_END_ADDRESS
RAMTST_START_ADDRESS
RAMTST_END_ADDRESS
RAMTST_START_ADDRESS
RAMTST_END_ADDRESS
RAMTST_END_ADDRESS
RAMTST_END_ADDRESS

2 Acronyms and Abbreviations

Abbreviation / Acronym:	*Description:*
API	Application Programming Interface
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DMA	Direct Memory Access
NMI	Non Maskable Interrupt
RAM	Random Access Memory

Definitions
Note: These definition are copied from the AUTOSAR_Glossary.doc

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

[1] AUTOSAR List of Basic Software Modules
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_BasicSoftwareModules.pdf

[2] Layered Software Architecture
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_General.pdf

[4] General Requirements on SPAL
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_SPAL_General.pdf

[5] Requirements on RAM Test
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_RAM_Test.pdf

[6] AUTOSAR Basic Software Module Description Template,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_BSWMDTemplate.pdf

3.2 Related standards and norms

[7] D1.5-General Architecture; ITEA/EAST-EEA, Version 1.0; chapter 3, page 72 et seq.

[8] D2.1-Embedded Basic Software Structure Requirements; ITEA/EAST-EEA, Version 1.0 or higher.

[9] D2.2-Description of existing solutions; ITEA/EAST-EEA, Version 1.0 or higher.

[10] CEI/IEC 61508-2:2000: Requirements for electrical/electronic/programmable electronic safety-related systems

[11] CEI/IEC 61508-7:2000: Requirements for electrical/electronic/programmable electronic safety-related systems

4 Constraints and Assumptions

Note: To achieve IEC certification (SIL1, SIL2, SIL3), the software implementation must be according to the requirements of IEC61508-3.

4.1 Limitations

RamTst002: During RAM Test, no other software shall be allowed to modify the RAM area under test. The RAM Test shall be implemented in small atomic pieces in order to accomplish this.

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).

RamTst082: The implementer shall provide integration hints for each algorithm, e.g. “do not use in parallel with a DMA”.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to Other Modules

The RAM Test is an algorithm which tests a the RAM by testing contiguous sets of cells until the entire desired memory space is tested. Each cell set is tested by calling the RAM Test main function.

The set of available algorithms for the RAM Test must be selected at pre-compile time. The software responsible for monitoring the RAM state of health must switch between the compile-time configured algorithms, according to the results of the ECU safety analysis.

The definition of the number of blocks to be tested and the start/end addresses of the blocks must be selected at pre-compile or link time.

The size of the cell set to be tested is set as a default at pre-compile or link time based upon the needs of the scheduler. The size of the cell set may be changed during runtime to accommodate a change in the schedule. One cell set is equal to the NUMBER_OF_TESTED_CELLS, described later in this document.

If the user calls a RAM Test API to test all or part of the RAM immediately (in the foreground), then the user is responsible to mask interrupts as desired while the main RAM Test function is running.

For cyclic calling of the main function, the ECU State Manager or the BSW scheduler must schedule the RAM Test main function.

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

5.1 File structure

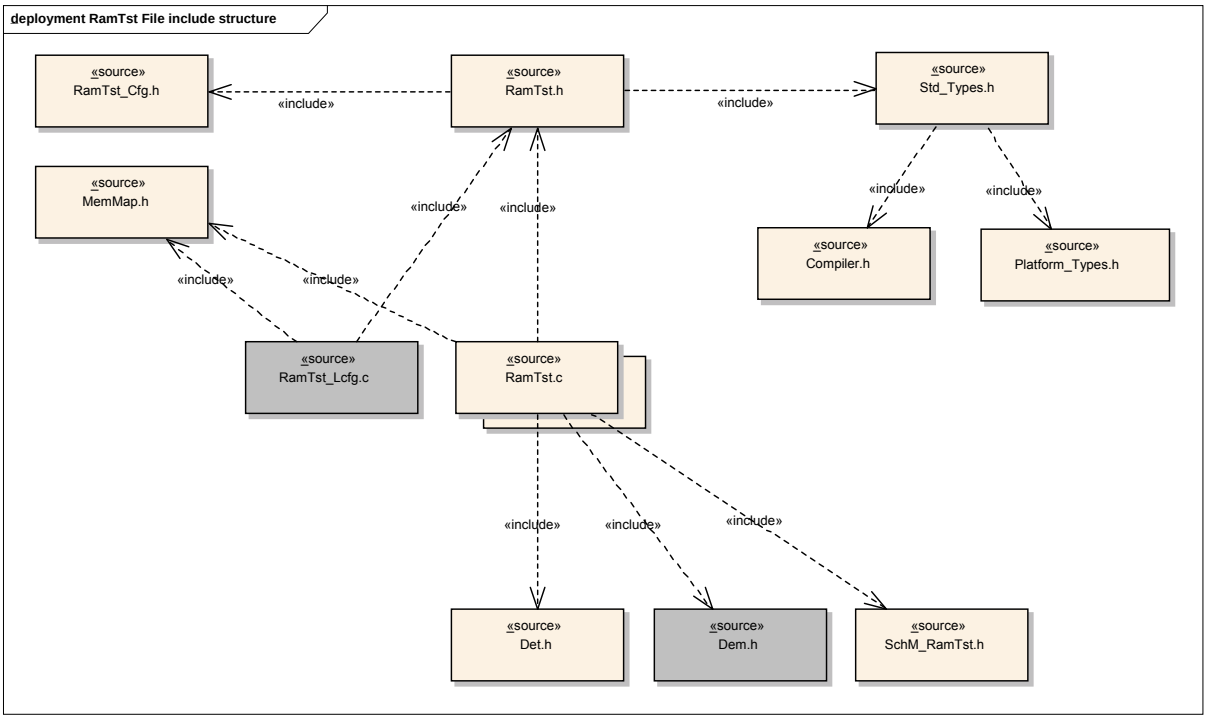
5.1.1 Code file structure

RamTst086: The code file structure for the RAM Test module shall not be defined within this specification completely. At this point, it shall be pointed out that the code-file structure shall include the following files named:

- `RamTst_Lcfg.c` – for link time configurable parameters

5.1.2 Header file structure

RamTst003: The include structure for the source code of the RAM Test module shall be as follows:



RamTst072: The module shall include the `Dem.h` file. By this inclusion, the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in `Dem_IntErrId.h`.

RamTst087: References to c-configuration parameter (link time) shall be placed into the `RamTst_Cfg.h` file.

6 Requirements Traceability

Document: AUTOSAR requirements on Basic Software, general [3]

Requirement	*Satisfied by*
Functional Requirements	
[BSW101] Initialization interface	RamTst007, RamTst099
[BSW004] Version check	RamTst080
[BSW159] Tool-based configuration	Both static and runtime configuration parameters are located outside the source code of the module. This is the prerequisite for automatic configuration.
[BSW167] Static configuration checking	
[BSW168] Diagnostic interface of SW components	Not applicable (not a SW-Component)
[BSW00323] API parameter checking	RamTst033, RamTst037, RamTst039, RamTst040, RamTst068, RamTst095, RamTst115
[BSW00336] Shutdown interface	Not applicable (this module does not need such a function)
[BSW00337] Classification of errors	RamTst067
[BSW00338] Detection and reporting of development errors	RamTst067, RamTst068, RamTst069, RamTst070, RamTst084, RamTst097, RamTst116
[BSW00339] Reporting of production relevant error status	RamTst073, RamTst067, RamTst076, RamTst071, RamTst111
[BSW00344] Reference to link-time configuration	RamTst025, RamTst026, RamTst027, RamTst066, RamTst090, RamTst091
[BSW00345] Pre-compile-time configuration	RamTst065, RamTts058, RamTst066, RamTst068, RamTst070, RamTst079, RamTst090, RamTst091
[BSW00369] Do not return development error codes via API	RamTst033, RamTst037, RamTst039, RamTst040, RamTst089, RamTst095
[BSW00375] Notification of wake-up reason	Not applicable (wakeups are not supported by this module)
[BSW00380] Separate C-files for configuration parameters	RamTst086
[BSW00381] Separate configuration header file for pre-compile time parameters	
[BSW00383] List dependencies of configuration files	Not applicable (there are no dependencies to other configuration files)
[BSW00384] List dependencies to other modules	RamTst072
[BSW00385] List possible error notifications	RamTst067
[BSW00386] Configuration for detecting an error	Not applicable (no configuration for error detection)
[BSW00387] Specify the configuration class of callback function	This version supports only pointer at link time.
[BSW00388] Introduce containers	RamTst065, RamTst066, RamTst070, RamTst090, RamTst091
[BSW00389] Containers shall have names	RamTst065, RamTst066, RamTst070, RamTst090, RamTst091
[BSW00390] Parameter content shall be unique within the module	RamTst065, RamTst066, RamTst070, RamTst090, RamTst091
[BSW00391] Parameter shall have unique names	RamTst065, RamTst066, RamTst070, RamTst090, RamTst091 Prefix “RamTst” added to each parameter
[BSW00392] Parameters shall have a type	RamTst065, RamTst066, RamTst070, RamTst090, RamTst091
[BSW00393] Parameters shall have a range	RamTst065, RamTst066, RamTst070,

	RamTst090, RamTst091
[BSW00394] Specify the scope of the parameters	RamTst065, RamTst066, RamTst070, RamTst090, RamTst091 "Local" marked as Module (template and SRS General are inconsistent)
[BSW00395] List the required parameters (per parameter)	All parameter in section 10.2 Containers and configuration parameters are required.
[BSW00396] Configuration classes	See section 10.2.1 Variants
[BSW00397] Pre-compile-time parameters	RamTst065, RamTst066, RamTst070, RamTst090, RamTst091
[BSW00398] Link-time parameters	RamTst066, RamTst090, RamTst091
[BSW00399] Loadable post-build time parameters	Not applicable (post build time configuration is not supported)
[BSW00400] Selectable post-build time parameters	Not applicable (post build time configuration is not supported)
[BSW00402] Published information	RamTst080, RamTst081, RamTst118
[BSW00404] Reference to post build time configuration	Not applicable (post build time configuration is not supported)
[BSW00405] Reference to multiple configuration sets	Not applicable (post build time is not supported)
[BSW00406] Check module initialization	RamTst006, RamTst012, RamTst013
[BSW00407] Function to read out published parameters	RamTst078, RamTst079, RamTst109
[BSW00409] Header files for production code error IDs	
[BSW00412] Separate H-file for configuration parameters	RamTst087
[BSW00416] Sequence of Initialization	Not applicable (this is a general software integration requirement)
[BSW00417] Reporting of error events by non-basic software	Not applicable (this is a basic software module)
[BSW00419] Separate C-files for pre-compile time configuration parameters	
[BSW00422] Debouncing of production relevant error status	Not applicable (it makes no sense to debounce a Ram error)
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR interfaces	Not applicable (this module has no connection to the RTE)
[BSW00424] BSW main processing function task allocation	Not applicable (the scheduling of a BSW is not part of this SWS)
[BSW00425] Trigger conditions for schedulable objects	Not applicable (requirement for the implementer)
[BSW00426] Exclusive areas in BSW modules	Not applicable (requirement for the implementer)
[BSW00427] ISR description for BSW modules	Not applicable (this module has no interrupt service routines)
[BSW00428] Execution order dependencies of main processing functions	Not applicable (requirement for the implementer and integrator)
[BSW00429] Restricted BSW OS functionality access	Not applicable (this module does not use OS services)
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (this is a special requirement for the BSW scheduler)
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (this module does not have send/receive functionality)
[BSW00433] Calling of main processing functions	*New API to be defined to support this requirement.*
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (this is a special requirement for the BSW

	scheduler)
[BSW00437] No-init area in RAM	Not applicable (this is a requirement on the memory manager)
[BSW00438] Post-build configuration data structure	Not applicable (post build time configuration is not supported)
Non-functional Requirements	
[BSW003] Version identification	RamTst080, RamTst117
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable (this is a requirement on architecture)
[BSW006] Platform independency	Not applicable (requirement for the implementer)
[BSW007] HIS MISRA C	Common AUTOSAR non-functional requirement for the implementer.
[BSW009] Module User Documentation	Not applicable (requirement for the implementer)
[BSW010] Memory resource documentation	Not applicable (requirement for the implementer)
[BSW158] Separation of configuration from implementation	RamTst086
[BSW160] Human-readable configuration data	Common AUTOSAR non-functional requirement for the implementer.
[BSW161] Microcontroller abstraction	Not applicable (this is a requirement on architecture)
[BSW162] ECU layout abstraction	Not applicable (this is a requirement on architecture)
[BSW164] Implementation of interrupt service routines	Not applicable (this module has no interrupt service routines)
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable (not a SW-Component)
[BSW171] Configurability of optional functionality	RamTst065
[BSW172] Compatibility and documentation of scheduling strategy	Not applicable (requirement for the implementer)
[BSW00300] Module naming convention	Common AUTOSAR non-functional requirement for the implementer.
[BSW00301] Limit imported information	Not applicable (requirement for the implementer)
[BSW00302] Limit exported information	Not applicable (requirement for the implementer)
[BSW00304] AUTOSAR integer data types	See section 8.1 Imported types
[BSW00305] Self-defined data types naming convention	See section 8.2 Type definitions
[BSW00306] Avoid direct use of compiler and platform specific keywords	Not applicable (requirement for the implementer)
[BSW00307] Global variables naming convention	Common AUTOSAR non functional requirement for the implementer.
[BSW00308] Definition of global data	Not applicable (requirement for the implementer)
[BSW00309] Global data with read-only constraint	Not applicable (requirement for the implementer)
[BSW00310] API naming convention	See chapter 8 API specification
[BSW00312] Shared code shall be reentrant	Not applicable (requirement for the implementer)
[BSW00314] Separation of interrupt frames and service routines	Not applicable (this module has no interrupt service routines)
[BSW00318] Format of module version numbers	RamTst080
[BSW00321] Enumeration of module version numbers	Not applicable (requirement for the implementer)
[BSW00325] Runtime of interrupt service routines	Not applicable

	(this module has no interrupt service routines)
[BSW00326] Transition from ISRs to OS tasks	Not applicable (this module has no interrupt service routines)
[BSW00327] Error values naming convention	See section 7.2 Error classification
[BSW00328] Avoid duplication of code	Not applicable (requirement for the implementer)
[BSW00329] Avoidance of generic interfaces	See chapter 8 API specification
[BSW00330] Usage of macros / inline functions instead of functions	RamTst117
[BSW00331] Separation of error and status values	RamTst102, RamTst103 8.2.1 RamTst_ExecutionStatusType and 8.2.2 RamTst_TestResultType
[BSW00333] Documentation of callback function context	Not applicable (requirement for the implementer)
[BSW00334] Provision of XML file	Not applicable (requirement for the implementer)
[BSW00335] Status values naming convention	See section 8.2 Type definitions
[BSW00341] Microcontroller compatibility documentation	Not applicable (requirement for the implementer)
[BSW00342] Usage of source code and object code	Common AUTOSAR non-functional requirement for the implementer.
[BSW00343] Specification and configuration of time	Common AUTOSAR non-functional requirement for the implementer.
[BSW00346] Basic set of module files	RamTst086, RamTst003
[BSW00347] Naming separation of different instances of BSW drivers	Common AUTOSAR non-functional requirement for the implementer and integrator.
[BSW00348] Standard type header	See chapter 8.1Imported types
[BSW00350] Development error detection keyword	RamTst068, RamTst070
[BSW00353] Platform specific type header	Not applicable (requirement for the implementer)
[BSW00355] Do not redefine AUTOSAR integer data types	See section 8.2 Type definitions
[BSW00357] Standard API return type	RamTst074 See section 8.1 Imported types
[BSW00358] Return type of init() functions	RamTst099 See section 8.3.1 RamTst_Init
[BSW00359] Return type of callback functions	See section 8.6.3 Configurable interfaces
[BSW00360] Parameters of callback functions	See section 8.6.3 Configurable interfaces
[BSW00361] Compiler specific language extension header	Not applicable (requirement for the implementer)
[BSW00370] Separation of callback interface from API	Not applicable (the notification functions will be handled via a function pointer in the configuration init structure)
[BSW00371] Do not pass function pointers via API	Not applicable (requirement for the implementer)
[BSW00373] Main processing function naming convention	RamTst110 See section 8.5.1 RamTst_MainFunction
[BSW00374] Module vendor identification	Not applicable (requirement for the implementer)
[BSW00376] Return type and parameters of main processing functions	RamTst110 See section 8.5.1 RamTst_MainFunction
[BSW00377] Module specific API return types	See sections 8.1 Imported types and 8.2 Type definitions
[BSW00378] AUTOSAR boolean type	Not applicable (requirement for the implementer)
[BSW00379] Module identification	Not applicable

	(requirement for the implementer)
[BSW00401] Documentation of multiple instances of configuration parameters	Not applicable (requirement for the implementer)
[BSW00408] Configuration parameter naming convention	See section 10.2 Containers and configuration parameters
[BSW00410] Compiler switches shall have defined values	See section 10.2 Containers and configuration parameters
[BSW00411] Get version info keyword	RamTst070, RamTst079
[BSW00413] Accessing instances of BSW modules	Not applicable (instances makes no sense for this module)
[BSW00414] Parameter of init function	RamTst093 See section 8.3.1 RamTst_Init
[BSW00415] User dependent include files	RamTst003
[BSW00435] Module header file structure for the basic software scheduler	RamTst003
[BSW00436] Module header file structure for the basic software memory mapping	RamTst003

Document: AUTOSAR requirements on Basic Software, cluster SPAL, General Requirements [4]

Requirement	***Satisfied by***
General Requirements	
[BSW157] Notification mechanisms of drivers and handlers	RamTst042, RamTst043, RamTst044, RamTst045, RamTst046, RamTst066
[BSW12056] Configuration of notification mechanisms	RamTst042, RamTst043, RamTst044, RamTst045, RamTst046, RamTst066
[BSW12057] Driver module initialization	RamTst007, RamTst025, RamTst026, RamTst027, RamTst066
[BSW12063] Raw value mode	Not applicable (this module does not provide physical signals)
[BSW12064] Change of operation mode during running operation	Not applicable (only one operation mode available)
[BSW12067] Setting of wake-up conditions	Not applicable (this module has no wakeup sources)
[BSW12068] MCAL initialization sequence	Not applicable (this is a requirement for the implementer)
[BSW12069] Wake-up notification of ECU State Manager	Not applicable (this module has no wakeup sources)
[BSW12075] Use of application buffers	Not applicable (no use of a buffering mechanism)
[BSW12125] Initialization of hardware resources	Not applicable (this module accesses only RAM, so no hardware resource initialization is necessary)
[BSW12129] Resetting of interrupt flags	Not applicable (there is no interrupt service routine in this module)
[BSW12163] Driver module deinitialization	*New API to be defined to support this requirement.*
[BSW12169] Control of operation mode	*To be defined.*
[BSW12263] Object code compatible configuration concept	RamTst025, RamTst026, RamTst027, RamTst066
[BSW12267] Configuration of wake-up sources	Not applicable (this module has no wakeup sources)
[BSW12448] Behavior after development error detection	RamTst033, RamTst037, RamTst039, RamTst040, RamTst084, RamTst095
[BSW12461] Responsibility for register initialization	Not applicable (this module uses no registers)

[BSW12462] Provide settings for register initialization	Not applicable (this module uses no registers)
[BSW12463] Combine and forward settings for register initialization	Not applicable (this module uses no registers)
Non-Functional Requirements	
[BSW12077] Non-blocking implementation	RamTst081 Not applicable (requirement for the implementer)
[BSW12078] Runtime and memory efficiency	Not applicable (requirement for the implementer)
[BSW12092] Access to drivers	Not applicable (requirement for the implementer)
[BSW12264] Specification of configuration items	See chapter 10.2 Containers and configuration parameters
[BSW12265] Configuration data shall be kept constant	Not applicable (requirement for the implementer)

Document: AUTOSAR Requirements on Basic Software, Cluster SPAL, RAM Test [3]

Requirement	***Satisfied by***
[BSW13800] Number of tested cells shall be changeable at runtime	RamTst026, RamTst036, RamTst107
[BSW13801] Test cell size shall be configurable at pre-compile time	RamTst049, RamTst090
[BSW13802] Multiple RAM areas shall be configurable at post build/ link time	RamTst026, RamTst091
[BSW13803] A subset of available RAM Test algorithms shall be selectable at pre-compile time	RamTst026, RamTst027, RamTst063, RamTst065, RamTst083, RamTst084, RamTst085, RamTst097, RamTst105
[BSW13804] A subset of the pre-compile time selected RAM Test algorithms shall be selectable at runtime	RamTst021, RamTst083, RamTst105, RamTst106
[BSW13805] Checkerboard test algorithm shall be available	RamTst050 See sections 8.2.3 RamTst_AlgorithmType and section 10.2 Containers and Configuration Parameters
[BSW13806] Walk path test algorithm shall be available	RamTst052 See sections 8.2.3 RamTst_AlgorithmType and section 10.2 Containers and Configuration Parameters
[BSW13807] March test algorithm shall be available	RamTst051 See sections 8.2.3 RamTst_AlgorithmType and section 10.2 Containers and Configuration Parameters
[BSW13808] Galpat test algorithm shall be available	RamTst053 See sections 8.2.3 RamTst_AlgorithmType and section 10.2 Containers and Configuration Parameters
[BSW13809] RAM Test execution management	RamTst008, RamTst026, RamTst059, RamTst070, RamTst090, RamTst091, RamTst107, RamTst108
[BSW13810] Current status of RAM Test execution per block shall be available through a get status interface	RamTst010, RamTst011, RamTst019, RamTst024, RamTst038, RamTst104
[BSW13811] Non-destructive RAM Test	RamTst060, RamTst061
[BSW13812] Destructive RAM Test	RamTst061
[BSW13813] Abraham test algorithm shall be	RamTst055

available	See sections 8.2.3 RamTst_AlgorithmType and section 10.2 Containers and Configuration Parameters
[BSW13816] Effects of instruction / data queue shall be taken into account	RamTst062
[BSW13818] Transparent Galpat test algorithm shall be available	RamTst054 See sections 8.2.3 RamTst_AlgorithmType and section 10.2 Containers and Configuration Parameters
[BSW13820] RAM Test execution status shall be provided by a notification mechanism	RamTst042, RamTst043, RamTst044, RamTst045, RamTst046, RamTst113, RamTst114
[BSW13821] The RAM Test Module shall be designed to fulfill SIL3 Requirements	RamTst050, RamTst051, RamTst052, RamTst053, RamTst054, RamTst055

Implementation requirements originating within this SWS document.

Requirement
RamTst002
RamTst005, RamTst082
RamTst009
RamTst014
RamTst018
RamTst047
RamTst082
RamTst094
RamTst096 (should be a general requirement?)
RamTst100 (should be an SRS requirement)
RamTst101 (should be an SRS requirement)

7 Functional Specification

7.1 Requirements

RamTst005: The RAM Test module shall provide the RAM test as an asynchronous service.

RamTst063: 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 RamTst050, RamTst051, RamTst052, RamTst053, RamTst054, RamTst055.

RamTst060: 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.

RamTst061: 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.

RamTst062: 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.

RamTst050: The RAM Test module shall provide a checkerboard test algorithm as stated in [11] , A.5.1.

RamTst051: The RAM Test module shall provide a March test algorithm as stated in [11], A.5.1.

RamTst052: The RAM Test module shall provide a WalkPath test algorithm as stated in [11], A.5.2.

RamTst053: The RAM Test module shall provide a Galpat test algorithm as stated in [11], A.5.3.

RamTst054: The RAM Test module shall provide a Transparent Galpat test algorithm as stated in, [11], A.5.3.

RamTst055: The RAM Test module shall provide an Abraham test algorithm as stated in [11], A.5.4.

7.2 Error classification

RamTst073: Values for production code Event Ids are assigned externally by the configuration of the DEM. They are published in the file `Dem_IntErrId.h` and included via `Dem.h`.

RamTst074: Development error values are of type uint8.

RamTst067: The following errors and exceptions shall be detectable by the RAM Test depending on its build version (development/production mode):

Type or error	*Relevance*	*Related error code*	*Value [hex]*	*Requirement*
Failure within RAM Test execution status	Development	`RAMTST_E_STATUS_FAILURE`	`0x01`	RamTst033, RamTst037, RamTst095, RamTst097
API parameter out of specified range	Development	`RAMTST_E_OUT_OF_RANGE`	`0x02`	RamTst039, RamTst040, RamTst084
API service used without module initialization	Development	`RAMTST_E_UNINIT`	`0x03`	RamTst089
RAM Failure	Production	`RAMTST_E_RAM_FAILURE`	`Assigned by DEM`	RamTst011

7.3 Error detection

RamTst068: The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch `RamTstDevErrorDetect` (see chapter 10) shall activate or deactivate the detection of all development errors.

RamTst115: If the `RamTstDevErrorDetect` switch is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in section 7.2 and chapter 8.

RamTst076: The detection of production code errors cannot be switched off.

RamTst089: 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 Development Error Tracer (if development error detection is enabled).

7.4 Error notification

RamTst069: 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 RamTst067.

RamTst071: Production errors shall be reported to Diagnostic Event Manager (DEM) via the `Dem_ReportErrorStatus` API.

RamTst116: Detected development errors shall be reported to the `Det_ReportError` service of the Development Error Tracer (DET) if the pre-processor switch `RamTstDevErrorDetect` is set (see chapter 10).

8 API Specification

8.1 Imported types

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

RamTst098:

Header file	*Imported Type*
Dem_Types.h	Dem_EventIdType
Std_Types.h	Std_VersionInfoType

8.2 Type definitions

8.2.1 RamTst_ExecutionStatusType

Name:	`RamTst_ExecutionStatusType`	
Type:	`Enumeration`	
Range:	`RAMTST_EXECUTION_UNINIT`	The RAM Test is not initialized or not usable.
	`RAMTST_EXECUTION_INIT`	The RAM Test is initialized and ready to be started.
	`RAMTST_EXECUTION_RUNNING`	The RAM Test is currently running.
	`RAMTST_EXECUTION_STOPPED`	The RAM Test is stopped.
	`RAMTST_EXECUTION_SUSPENDED`	The RAM Test is waiting to be re-sumed.
	`RAMTST_EXECUTION_ALLOWED`	The RAM Test is allowed to continue (ALLOW or RESUME was called).
	`RAMTST_EXECUTION_STATE_UNDEFINED`	State used for manufacturer's white box testing.
Description:	This is a status value returned by the API service RamTst_GetExecutionStatus().	

RamTst006: For the type `RamTst_ExecutionStatusType`, the enumeration value `RAMTST_EXECUTION_UNINIT` shall be the default value after a reset. This enumeration value shall have the numeric value 0.

8.2.2 RamTst_TestResultType

Name:	`RamTst_TestResultType`	
Type:	`Enumeration`	
Range:	`RAMTST_RESULT_NOT_TESTED`	The RAM Test is not executed.
	`RAMTST_RESULT_OK`	The RAM Test has been tested with OK result
	`RAMTST_RESULT_NOT_OK`	The RAM Test has been tested with NOT-OK result.
	`RAMTST_RESULT_UNDEFINED`	The RAM Test is currently running.
Description:	This is a status value returned by the API service RamTst_GetTestResult().	

RamTst012: For the type `RamTst_TestResultType`, the enumeration value `RAMTST_RESULT_NOT_TESTED` shall be the default value after a reset. This enumeration value shall have the numeric value 0.

8.2.3 RamTst_AlgorithmType

Name:	`RamTst_AlgorithmType`	
Type:	`Enumeration`	
Range:	`RAMTST_ALGORITHM_UNDEFINED`	Undefined algorithm (uninitialized value)
	`RAMTST_CHECKERBOARD_TEST`	Checkerboard test algorithm
	`RAMTST_MARCH_TEST`	March test algorithm
	`RAMTST_WALK_PATH_TEST`	Walk path test algorithm
	`RAMTST_GALPAT_TEST`	Galpat test algorithm
	`RAMTST_TRANSP_GALPAT_TEST`	Transparent Galpat test algorithm
	`RAMTST_ABRAHAM_TEST`	Abraham test algorithm
Description:	This is a value returned by the API service RamTst_GetTestAlgorithm().	

RamTst013: For the type `RamTst_AlgorithmType`, the enumeration value `RAMTST_ALGORITHM_UNDEFINED` shall be the default value after reset. This enumeration value shall have the numeric value 0.

RamTst058: The type `RamTst_AlgorithmType` shall contain only the enumerations of the algorithms selected at pre-compile time.

8.2.4 RamTst_ConfigType

Name:	`RamTst_ConfigType`
Type:	`Structure`
Range:	Implementation specific.
Description:	This type of the external data structure shall contain the initialization data for the RAM Test

RamTst025: The type `RamTst_ConfigType` shall denote the external data structure which contains the configuration data for the RAM Test module.

RamTst026: Within the configuration data for the RAM Test module, there shall be an algorithm specific configuration container for each test algorithm that was pre-compile selected to be available to the user, as listed in the `RamTst_Algorithm` type. The test agorithm configuration container is defined in `RamTstAlgParams`.

RamTst027: Within the configuration data for the RAM Test module, each test algorithm listed in the `RamTst_Algorithm` type and defined in a `RamTstAlgParams` container as required in RamTst026 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.

8.2.5 RamTst_NumberOfTestedCellsType

Name:	`RamTst_NumberOfTestedCellsType`
Type:	`uint8,uint16,uint32`
Range:	`1..2n-1` --
Description:	Data type of number of tested RAM cells

8.3 Function definitions

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

8.3.1 RamTst_Init

RamTst099:

Service name:	RamTst_Init

Syntax:	`void RamTst_Init(`
	
	`)`

Service ID[hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (in-out):	None
Parameters (out):	None
Return value:	None
Description:	Service for RAM Test initialization.

Note: See also RamTst093: in 10.2.1 Variants.

RamTst007: The function `RamTst_Init` shall initialize all RAM Test relevant registers and global variables and change the execution status to `RAMTST_EXECUTION_INIT`. The test is initialized to use the default test algorithm as configured by its `RamTstAlgParams` container.

RamTst096: The `RamTest_Init` function shall be called only once after a reset, unless a `RamTst_Deinit` call is made before calling `RamTst_Init` again.

Note: The RAM test starts at the first call of `RamTst_MainFunction`.

The function `RamTst_Init` requires the configuration container "RamTstConfigParams".

8.3.2 RamTst_DeInit

RamTst146:

Service name:	RamTst_DeInit

Syntax:	`void RamTst_DeInit(`
	
	`)`

Service ID[hex]:	0x0c
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (in-out):	None
Parameters (out):	None
Return value:	None
Description:	Service for RAM Test deinitialization.

RamTst147: The function `RamTst_DeInit` shall deinitialize all RAM Test relevant registers and global variables that were initialized by `RamTst_Init`.

If the RAM Test is in the `"terminated"` state after a RamTst_DeInit call, a call to any RamTst Module function may result in unknown software behavior.

8.3.3 RamTst_Stop

RamTst100:

Service name:	RamTst_Stop

Syntax:	`void RamTst_Stop(`
	
	`)`

Service ID[hex]:	0x02
Sync/Async:	Asynchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (in-out):	None
Parameters (out):	None
Return value:	None
Description:	Service for stopping the RAM Test.

RamTst014: When the `RamTst_Stop` function is called, `RamTst_MainFunction` shall finish the current atomic sequence it is running, then go to the STOPPED state and change the execution status to `RAMTST_EXECUTION_STOPPED`. Test status is retained, but test parameters and loop data are not. The test algorithm is set to the default test algorithm.

RamTst148: After a RamTst_Stop call, RamTst_MainFunction shall not begin testing again when called by the scheduler until after a RamTst_Allow call.

RamTst033: If the DET is enabled and the execution status of the RAM Test is not `RAMTST_EXECUTION_RUNNING or RAMTST_EXECUTION_PAUSED`, the function `RamTst_Stop` shall report the error value `RAMTST_E_STATUS_FAILURE` to the DET, and then immediately return from the function.

The function RamTst_Stop requires the configuration parameter RamTstStopApi within the container "RamTstCommon".

8.3.4 RamTst_Allow

RamTst149:

Service name:	RamTst_Allow

Syntax:	`void RamTst_Allow(`
	
	`)`

Service ID[hex]:	0x03
Sync/Async:	Asynchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (in-out):	None
Parameters (out):	None
Return value:	None
Description:	Service for continuing the RAM Test after calling 'RamTst_Stop.

The function RamTst_Allow requires the configuration parameter RAMTST_STOP_ALLOW within the container "RamTstCommon".

8.3.5 RamTst_Suspend

RamTst150:

Service name:	RamTst_Suspend

Syntax:	`void RamTst_Suspend(`
	
	`)`

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

The function RamTst_Suspend requires the configuration parameter RAMTST_SUSPEND_API within the container "RamTstCommon".

8.3.6 RamTst_Resume

RamTst101:

Service name:	RamTst_Resume

Syntax:	`void RamTst_Resume(`
	
	`)`

Service ID[hex]:	0x0e
Sync/Async:	Asynchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (in-out):	None
Parameters (out):	None
Return value:	None
Description:	Service for continuing the RAM Test at the point is was suspended.

RamTst018: The function `RamTst_Resume` shall change the execution status to `RAMTST_EXECUTION_RESUME` when commanded to continue.

RamTst037: If DET is enabled and the execution status of the RAM Test module is not `RAMTST_EXECUTION_STOPPED`, the RAM Test module shall report the error value RAMTST_E_STATUS_FAILURE to the DET, and then immediately return from the function.

The function `RamTst_Resume` requires the configuration parameter `RAMTST_RESUME_API` within the container "RamTstCommon".

8.3.7 RamTst_GetExecutionStatus

RamTst102:

Service name:	RamTst_GetExecutionStatus	

Syntax:	`RamTst_ExecutionStatusType RamTst_GetExecutionStatus(`
	
	`)`	

Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (in-out):	None	
Parameters (out):	None	
Return value:	RamTst_ExecutionStatusType	See type definition
Description:	Service returns the current RAM Test execution status.	

RamTst019: The function `RamTst_GetExecutionStatus` shall return the current RAM Test execution status.

The function `RamTst_GetExecutionStatus` requires the configuration parameter `RamTstGetExecutionStatusApi` within the container "RamTstCommon".

8.3.8 RamTst_GetTestResult

RamTst103:

Service name:	RamTst_GetTestResult	

Syntax:	`RamTst_TestResultType RamTst_GetTestResult(`
	
	`)`	

Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (in-out):	None	
Parameters (out):	None	
Return value:	RamTst_TestResultType	See type definition
Description:	Service returns the current RAM Test result.	

RamTst024: The function `RamTst_GetTestResult` shall return the current RAM test result.

The function `RamTst_GetTestResult` requires the configuration parameter `RamTstGetTestResultApi` within the container "RamTstCommon".

8.3.9 RamTst_GetTestResultPerBlock

RamTst104:

Service name:	RamTst_GetTestResultPerBlock	

Syntax:	`RamTst_TestResultType RamTst_GetTestResultPerBlock(`
	`  RamTst_NumberOfBlocksType BlockID`
	`)`	

Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	BlockID	Number of blocks
Parameters (in-out):	None	
Parameters (out):	None	
Return value:	RamTst_TestResultType	See type definition
Description:	Service returns the current RAM Test result for the specified block.	

RamTst038: The function `RamTst_GetTestResultPerBlock` shall return the current RAM test result for the specified block.

RamTst039: 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_STATE_UNDEFINED.

The function RamTst_GetTestResultPerBlock requires the configuration parameter RamTstGetTestResultPerBlockApi within the container "RamTstCommon".

8.3.10 RamTst_GetVersionInfo

RamTst109:

Service name:	RamTst_GetVersionInfo	
Syntax:	`void RamTst_GetVersionInfo( Std_VersionInfoType* versioninfo )`	
Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (in-out):	None	
Parameters (out):	versioninfo	Pointer to the location / address where to store the version information of this module.
Return value:	None	
Description:	Service returns the version information of this module.	

RamTst078: The function RamTst_GetVersionInfo shall return the version information of this module. The version information includes:

- Module Id
- Vendor Id
- Vendor specific version numbers (BSW00407).

RamTst079: The function RamTst_GetVersionInfo shall be configurable at pre-compile time (On/Off) by the configuration parameter RamTstVersionInfoApi in the container "RamTstCommon".

RamTst117: If source code for caller and callee of RamTst_GetVersionInfo is available, the RAM test module should realize RamTst_GetVersionInfo as a macro, defined in the module's header file.

8.3.11 RamTst_GetTestAlgorithm

RamTst106:

Service name:	RamTst_GetTestAlgorithm
Syntax:	`RamTst_AlgorithmType RamTst_GetTestAlgorithm( )`
Service ID[hex]:	0x07
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant

Parameters (in):	None	
Parameters (in-out):	None	
Parameters (out):	None	
Return value:	RamTst_AlgorithmType	See type definition
Description:	Service returns the current RAM Test algorithm ID.	

RamTst021: The function `RamTst_GetTestAlgorithm` shall return the current RAM Test algorithm ID.

The function `RamTst_GetTestAlgorithm` requires the configuration parameter `RamTstGetTestAlgorithmApi` within the container "RamTstCommon".

8.3.12 RamTst_GetNumberOfTestedCells

RamTst108:

Service name:	RamTst_GetNumberOfTestedCells	
Syntax:	`RamTst_NumberOfTestedCellsType RamTst_GetNumberOfTestedCells( )`	
Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (in-out):	None	
Parameters (out):	None	
Return value:	RamTst_NumberOfTestedCellsType	Number of currently tested cells of RAM Test will be returned.
Description:	Service returns the current number of tested cells per cycle.	

RamTst034: The function `RamTst_GetNumberofTestedCells` shall read the current number of tested cells per cycle.

The function `RamTst_GetNumberofTestedCells` requires the configuration parameter `RamTstGetNumberOfTestedCellsApi` in the container "RamTstCommon".

8.3.13 RamTst_ChangeTestAlgorithm

RamTst105:

Service name:	RamTst_ChangeTestAlgorithm	
Syntax:	`void RamTst_ChangeTestAlgorithm( RamTst_AlgorithmType NewTestAlgorithm )`	
Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	NewTestAlgorithm	See type definition RamTst_AlgorithmType

Parameters (in-out):	None
Parameters (out):	None
Return value:	None
Description:	Service changes the used TestAlgorithm.

RamTst083: The function `RamTst_ChangeTestAlgorithm` shall change the test algorithm used by the RAM Test module to the new test algorithm given as parameter.

The test algorithm is initialized by the `RamTst_Init` function with the configuration parameter `DefaultTestAlgorithm`.

RamTst085: The function `RamTst_ChangeTestAlgorithm` shall re-initialize all RAM Test relevant registers and global variables with the values for the "NewTestAlgorithm".

RamTst084: If DET is enabled and the parameter "NewTestAlgorithm" is out of range, the function `RamTst_ChangeTestAlgorithm` shall report the error value `RAMTST_E_OUT_OF_RANGE` to the DET, leaving the current TestAlgorithm unchanged.

RamTst097: If DET is enabled and the execution status of the RAM Test module is not `RAMTST_EXECUTION_STOPPED`, the function `RamTst_ChangeTestAlgorithm` shall report the error value `RAMTST_E_STATUS_FAILURE` to the DET, then immediately return from the function.

RamTst094: The function `RamTst_ChangeTestAlgorithm` shall not change the test result status (of type `RamTst_TestResultType`). The function shall leave the result of the previous test valid.

The function `RamTst_ChangeTestAlgorithm` requires configuration of the default value in `RamTst_ConfigType`:

- Container: RamTstConfigParams
- Parameter: DefaultTestAlgorithm

The function `RamTst_ChangeTestAlgorithm` also requires the configuration parameter `RamTstChangeTestAlgorithmApi` within the container "RamTstCommon".

8.3.14 RamTst_ChangeNumberOfTestedCells

RamTst107:

Service name:	RamTst_ChangeNumberOfTestedCells
Syntax:	`void RamTst_ChangeNumberOfTestedCells( RamTst_NumberOfTestedCellsType NewNumberOfTestedCells`

	)	
Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	NewNumberOfTestedCells	See type definition
Parameters (in-out):	None	
Parameters (out):	None	
Return value:	None	
Description:	Service changes the current number of tested cells.	

RamTst036: The function `RamTst_ChangeNumberOfTestedCells` shall change the current number of tested cells.

RamTst040: If DET is enabled and the parameter `NewNumberOfTestedCells` is out of range (range: min= `MinNumberOfTestedCells` / max = `MaxNumberOfTestedCells` as defined by the integrator), 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.

RamTst095: 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 but shall report the error value RAMTST_E_STATUS_FAILURE to the DET.

The function `RamTst_ChangeNumberOfTestedCells` requires configuration of the start value in `RamTst_ConfigType`:

- Container: `RamTstAlgParams`
- Parameter: `NumberOfTestedCells`

The function `RamTst_ChangeNumberOfTestedCells` also requires the configuration parameter `RamTstChangeNumOfTestedCellsApi` in the container "RamTstCommon".

8.3.15 RamTst_ChangeMaxNumberOfTestedCells

RamTst151:

Service name:	RamTst_ChangeMaxNumberOfTestedCells	
Syntax:	`void RamTst_ChangeMaxNumberOfTestedCells( RamTst_NumberOfTestedCellsType NewNumberOfTestedCells )`	
Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	NewNumberOfTestedCells	See type definition
Parameters (in-out):	None	
Parameters (out):	None	

Return value:	None
Description:	Service changes the current maximum number of tested cells.

8.4 Callback notifications

Since the RAM Test is a driver module, it does not provide any callback functions for lower layer modules.

8.5 Scheduled functions

The Basic Software Scheduler calls these functions directly. The following functions shall have no return value and no parameter. All functions shall be non-reentrant.

Terms and definitions:

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

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

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

8.5.1 RamTst_MainFunction

RamTst110:

Service name:	RamTst_MainFunction

Syntax:	`void RamTst_MainFunction(`
	
	`)`

Service ID[hex]:	0x01
Timing:	VARIABLE_CYCLIC_OR_ON_PRECONDITION
Description:	Service for executing the RAM Test.

RamTst008: The function `RamTst_MainFunction` shall test the defined RAM blocks, starting with the first RAM block in the RamTstConfigParams.

RamTst009: The function `RamTst_MainFunction` shall set the RAM Test execution status from `RAMTST_EXECUTION_INIT` or `RAMTST_EXECUTION_ALLOW` to `RAMTST_EXECUTION_RUNNING` when calling the function the first time after initialization or after calling `RamTst_Stop`.

RamTst010: The function RamTst_MainFunction shall set the RAM Test result status of every block to RAMTST_RESULT_OK or RAMTST_RESULT_NOT_OK depending on the result of the test.

RamTst011: The function RamTst_MainFunction shall set the overall result status to RAMTST_RESULT_OK if all blocks are tested with result status RAMTST_RESULT_OK. If at least one block test result is RAMTST_RESULT_NOT_OK, then the function shall set the overall test result status to RAMTST_RESULT_NOT_OK regardless whether all blocks are already tested or not and report the production error RAMTST_E_RAM_FAILURE to the DEM.

RamTst047: After the function RamTst_MainFunction has completed testing all RAM blocks, the next call of the function RamTst_MainFunction shall restart the test from the beginning.

RamTst059: 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.

For pre-conditions on the function RamTst_MainFunction, see requirement RamTst002

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

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

RamTst111:

API function	*Description*
Dem_ReportErrorStatus	Reports errors to the DEM.

8.6.2 Optional Interfaces

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

RamTst112:

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

8.6.3 Configurable interfaces

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

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).

RamTst042: The callback notifications shall be configurable as function pointers within the initialization data structure (RamTst_ConfigType).

RamTst043: The callback notifications shall have no parameters and no return value.

RamTst044: If a callback notification is configured as null pointer, the RAM Test module shall not execute the callback.

8.6.3.1 RamTst Test Completed Notification

RamTst113:

Service name:	RamTst_Test_Completed_Notification

Syntax:	`void RamTst_Test_Completed_Notification(`
	
	`)`

Sync/Async:	--
Reentrancy:	Don't care
Parameters (in):	None
Parameters (in-out):	None
Parameters (out):	None
Return value:	None
Description:	The function RamTst_TestCompleted shall be called every time when all RAM blocks had been tested.

RamTst045: The RAM Test module shall call the callback notification `RamTst_TestCompleted` every time when it has tested all RAM blocks.

The callback notification `RamTst_TestCompleted` requires configuration of the parameter `RAMTST_TEST_COMPLETED_NOTIFICATION` within the container "RamTstConfigParams".

8.6.3.2 RamTst Error Notification

RamTst114:

Service name:	RamTst_Error_Notification

Syntax:	`void RamTst_Error_Notification(`
	
	`)`

Sync/Async:	--
Reentrancy:	Don't care
Parameters (in):	None
Parameters (in-out):	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 algorithm.

RamTst046: The RAM test module shall call the callback notification `RamTst_Error` every time when the selected test algorithm has detected a RAM failure.

The callback notification `RamTst_TestCompleted` requires configuration of the parameter `RamTstTestErrorNotification` within the container "RamTstConfigParams".

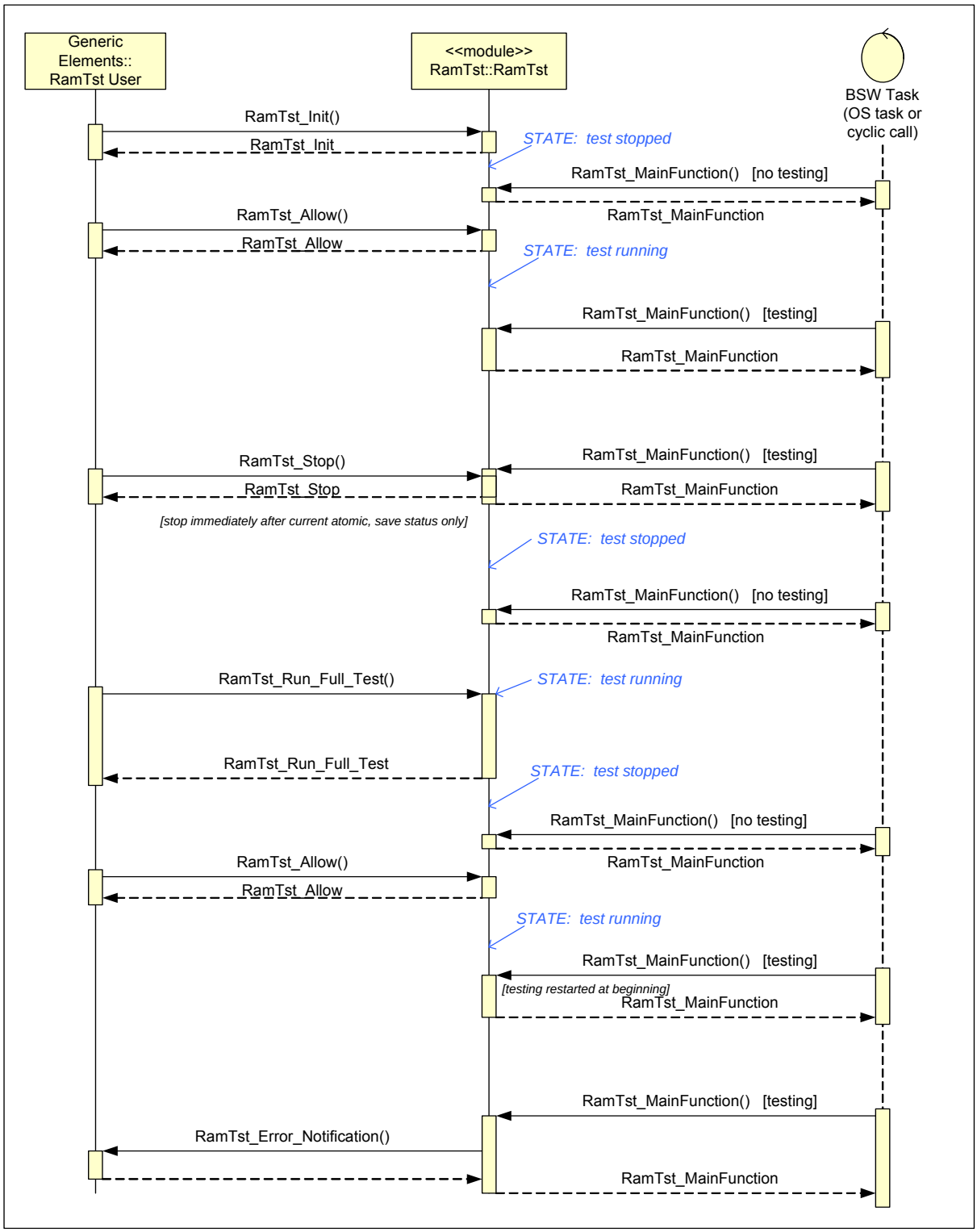
9 Sequence Diagrams

9.1 RamTst_MainFunction (Example)

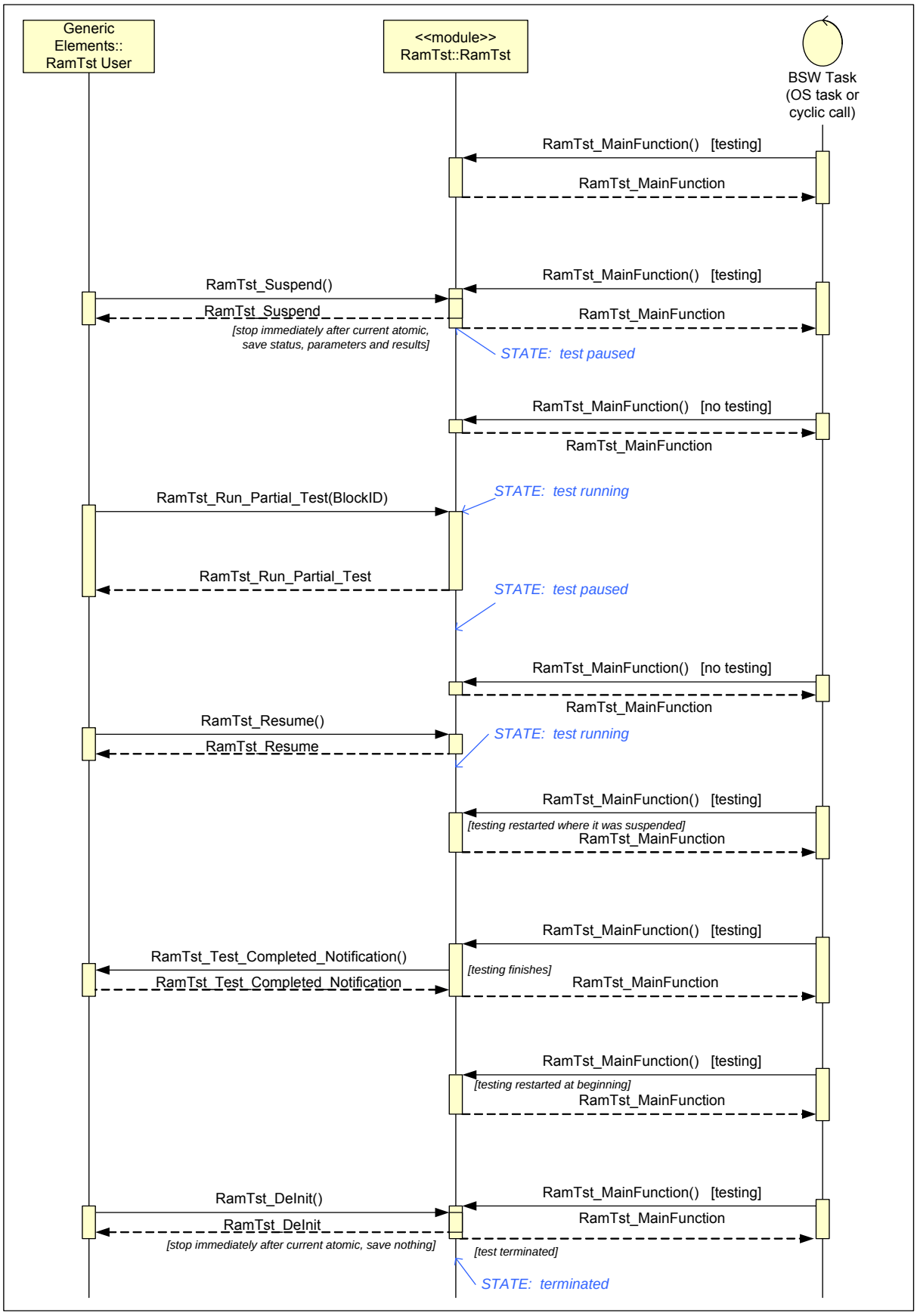
The following example sequence shows the initializaton of the RAM Test module, a foreground Run Full Test request, error notification, and the cyclic call background testing.

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.

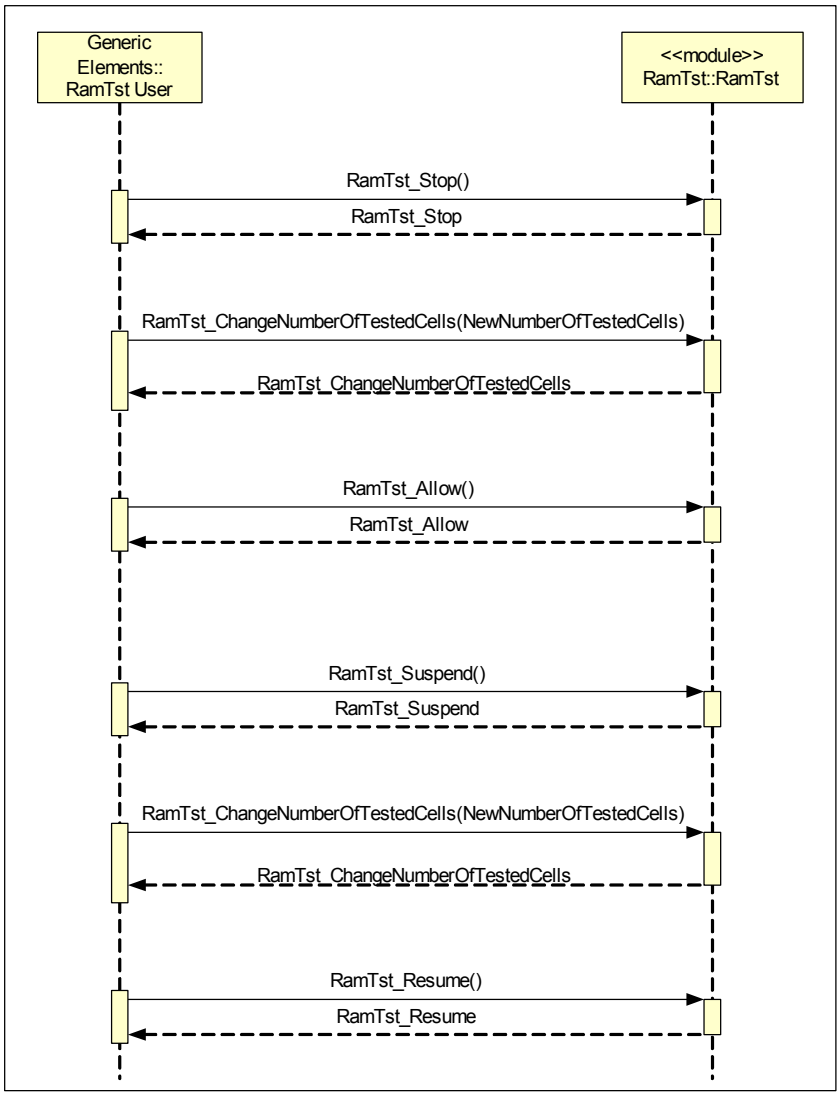
The stop request is handled following the currently running atomic sequence of the main routine, or at the next cyclic call of the main routine if it is not currently running. The allow request is handled at the next cyclic call of the main routine.



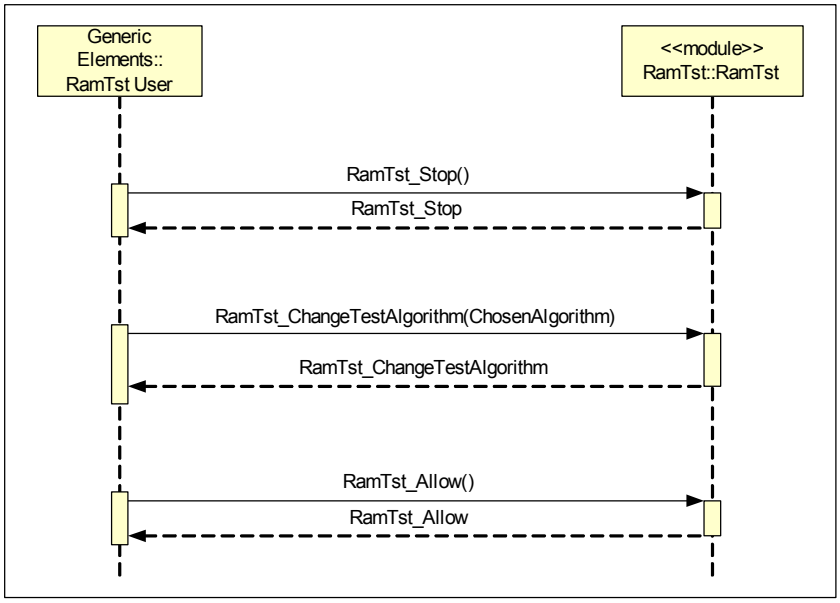
The next example shows the Suspend and Resume commands, a foreground Run Partial Test request, a Test Completed notification, and DeInit command.


Generic Elements:: RamTst User
<<module>> RamTst::RamTst
BSW Task (OS task or cyclic call)
RamTst_MainFunction() [testing]
RamTst_MainFunction
RamTst_Suspend()
RamTst_Suspend
[stop immediately after current atomic, save status, parameters and results]
STATE: test paused
RamTst_MainFunction() [no testing]
RamTst_Run_Partial_Test(BlockID)
STATE: test running
RamTst_Run_Partial_Test
STATE: test paused
RamTst_Resume()
RamTst_Resume
[testing restarted where it was suspended]
RamTst_Test_Completed_Notification()
RamTst_Test_Completed_Notification
[testing finishes]
[testing restarted at beginning]
RamTst_DeInit()
RamTst_DeInit
[stop immediately after current atomic, save nothing]
[test terminated]
STATE: terminated

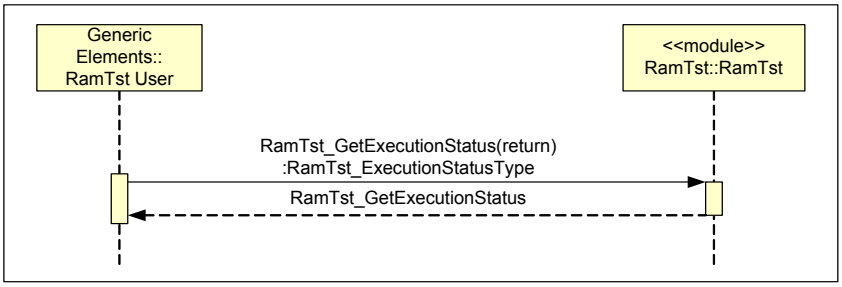
9.2 RamTst_ChangeNumberOfTestedCells



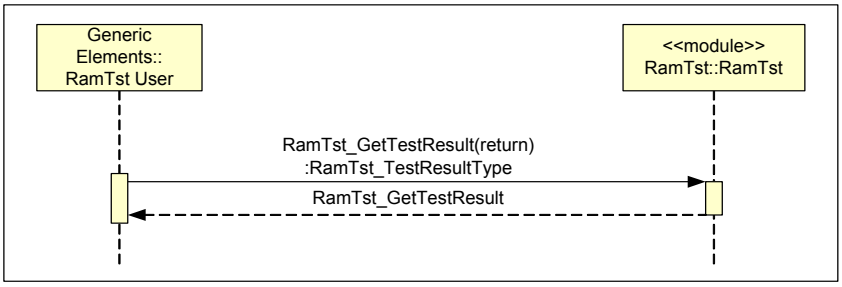
9.3 RamTst_ChangeTestAlgorithm



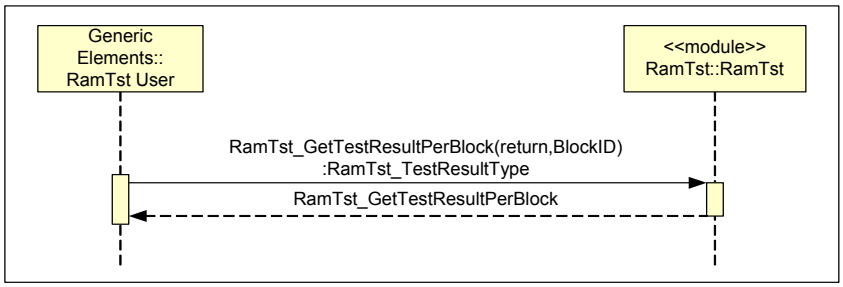
9.4 RamTst_GetExecutionStatus



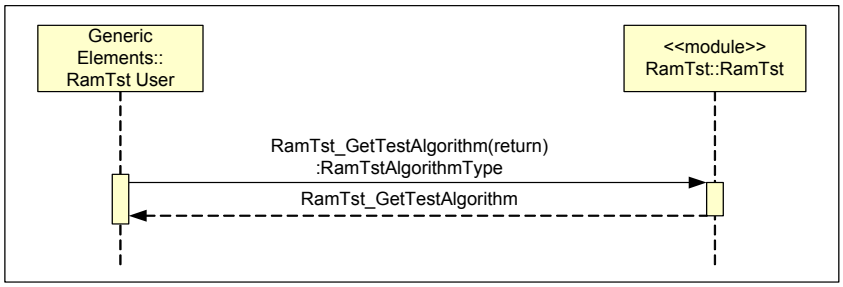
9.5 RamTst_GetTestResult



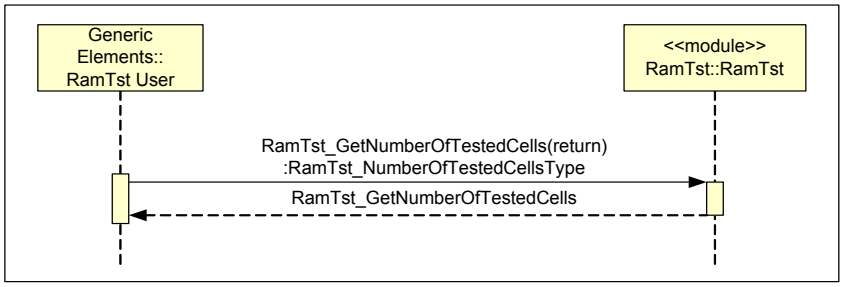
9.6 RamTst_GetTestResultPerBlock



9.7 RamTst_GetTestAlgorithm



9.8 RamTst_GetNumberOfTestedCells



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) that shall be used for the parameter specification. It is intended 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.

10.1 How to read this chapter

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

- AUTOSAR Layered Software Architecture [2]
- AUTOSAR ECU Configuration Specification [5]
    - This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

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

10.1.1 Configuration and configuration parameters

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

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

10.1.2 Containers

Containers structure the set of configuration parameters. This means:

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

10.1.3 Specification template for configuration parameters

The following tables consist of three sections:

- the general section
- the configuration parameter section
- the section of included/referenced containers

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

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

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

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

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

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

10.2 Containers and Configuration Parameters

The following sections summarize the containers of RAM Test configuration parameters. The detailed descriptions of the configuration parameters are described in Chapter 8 API Specification.

10.2.1 Variants

Variant PC: This variant is limited to pre-compile configuration parameters only.
The configuration parameters are fixed before compilation starts.
The configuration of the SW element is done at the source code level.
The intention of this variant is to optimize the parameter configuration for a source code delivery. See BSW00397.

Variant LT: This variant is limited to link time-configuration parameters only.
The configuration parameters are fixed after compiling and before linking (locating).
The configuration of the SW element is done at the object code level.
The intention of this variant is to optimize the parameter configuration for an object code delivery. See BSW00398.

RamTst093: The initialization function of this module shall always have a “void” as parameter. This means that, in contradiction to BSW00414 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

Module Name	**RamTst**
Module Description	Configuration of the RamTst module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
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 test algorithms are in the compiled code.

10.2.3 RamTstCommon

SWS Item	**RamTst070 :**
Container Name	RamTstCommon{RamTst_Common}
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 test algorithms are in the compiled code.
Configuration Parameters	

SWS Item	**RamTst120 :**		
Name	RamTstAllowApi {RAMTST_ALLOW_API}		
Description	Preprocessor switch to disable / enable the function "RamTst_Allow".		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**--**		
Name	RamTstChangeMaxNumOfTestedCellsApi {RAMTST_CHANGE_MAX_NUMBER_OF_TESTED_CELLS_API}		
Description	Preprocessor switch to disable / enable the function "RamTst_ChangeMaxNumberOfTestedCells".		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst118 :**		
Name	RamTstChangeNumOfTestedCellsApi {RAMTST_CHANGE_NUMBER_OF_TESTED_CELLS_API}		
Description	Preprocessor switch to disable / enable the function "RamTst_ChangeNumberOfTestedCells"		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst119 :**		
Name	RamTstChangeTestAlgorithmApi {RAMTST_CHANGE_TEST_ALGORITHM_API}		
Description	Preprocessor switch to disable / enable the function "RamTst_ChangeTestAlgorithm"		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst121 :**
Name	RamTstDevErrorDetect {RAMTST_DEV_ERROR_DETECT}
Description	Preprocessor switch to select the development error tracer (DET) ON or OFF

Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst122 :**		
Name	RamTstGetExecutionStatusApi {RAMTST_GET_EXECUTION_STATUS_API}		
Description	Preprocessor switch to disable / enable the function "RamTst_GetExecutionStatus"		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst123 :**		
Name	RamTstGetNumberOfTestedCellsApi {RAMTST_GET_NUMBER_OF_TESTED_CELLS_API}		
Description	Preprocessor switch to disable / enable the function "RamTst_GetNumberOfTestedCells"		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst124 :**		
Name	RamTstGetTestAlgorithmApi {RAMTST_GET_TEST_ALGORITHM_API}		
Description	Preprocessor switch to disable / enable the function "RamTst_GetTestAlgorithm"		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst125 :**		
Name	RamTstGetTestResultApi {RAMTST_GET_TEST_RESULT_API}		
Description	Preprocessor switch to disable / enable the function "RamTst_GetTestResult"		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	

	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst126 :**		
Name	RamTstGetTestResultPerBlockApi {RAMTST_GET_TEST_RESULT_PER_BLOCK_API}		
Description	Preprocessor switch to disable / enable the function "RamTst_GetTestResultPerBlock"		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst128 :**		
Name	RamTstGetVersionInfoApi {RAMTST_GET_VERSION_INFO_API}		
Description	Preprocessor switch to disable / enable the function "RamTst_GetVersionInfo"		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**--**		
Name	RamTstResumeApi {RAMTST_RESUME_API}		
Description	Preprocessor switch to disable / enable the function "RamTst_Resume".		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst127 :**		
Name	RamTstStopApi {RAMTST_STOP_API}		
Description	Preprocessor switch to disable / enable the function "RamTst_Stop"		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**--**
Name	RamTstSuspendApi {RAMTST_SUSPEND_API}
Description	Preprocessor switch to disable / enable the function "RamTst_Suspend".
Multiplicity	1

Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
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. The multiplicity of the included container depends on the pre-compile configuration in the container "RamTst_Algorithms".

10.2.4 RamTstAlgorithms

SWS Item	**RamTst065 :**
Container Name	RamTstAlgorithms{RamTst_Algorithms}
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.
Configuration Parameters	

SWS Item	**RamTst129 :**		
Name	RamTstAbrahamTestSelected {RAMTST_ABRAHAM_TEST_SELECTED}		
Description	Preprocessor switch to select the Abraham Test ON or OFF		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst130 :**		
Name	RamTstCheckerboardTestSelected {RAMTST_CHECKERBOARD_TEST_SELECTED}		
Description	Preprocessor switch to select the Checkerboard Test ON or OFF		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst132 :**
Name	RamTstGalpatTestSelected {RAMTST_GALPAT_TEST_SELECTED}
Description	Preprocessor switch to select the Galpat Test ON or OFF

Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst133 :**		
Name	RamTstMarchTestSelected {RAMTST_MARCH_TEST_SELECTED}		
Description	Preprocessor switch to select the March Test ON or OFF		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst134 :**		
Name	RamTstTranspGalpatTestSelected {RAMTST_TRANSP_GALPAT_TEST_SELECTED}		
Description	Preprocessor switch to select the Transparent Galpat Test ON or OFF		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst135 :**		
Name	RamTstWalkPathTestSelected {RAMTST_WALK_PATH_TEST_SELECTED}		
Description	Preprocessor switch to select the Walking Path Test ON or OFF		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.2.5 RamTstConfigParams

SWS Item	**RamTst066 :**
Container Name	RamTstConfigParams{RamTst_ConfigParams}
Description	This container specifies configuration parameters which are set at pre-compile or link time. The multiplicity of the included container depends on the pre-compile configuration in the container "RamTst_Algorithms".

Configuration Parameters			
SWS Item	**RamTst136 :**		
Name	RamTstDefaultTestAlgorithm {RAMTST_DEFAULT_TEST_ALGORITHM}		
Description	This is the default algorithm after the "RamTst_Init(..)" function. This is the initial value for a RAM variable which could be changed by the function "RamTst_ChangeTestAlgorithm"		
Multiplicity	1		
Type	EnumerationParamDef		
Range	RAMTST_ABRAHAM_TEST	--	
	RAMTST_CHECKERBOARD_TEST	--	
	RAMTST_GALPAT_TEST	--	
	RAMTST_MARCH_TEST	--	
	RAMTST_MOD_HAMMINGCODE_TEST	--	
	RAMTST_TRANSP_GALPAT_TEST	--	
	RAMTST_WALK_PATH_TEST	--	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**--**		
Name	RamTstMinNumberOfTestedCells {RAMTST_MIN_NUMBER_OF_TESTED_CELLS}		
Description	Minimum number of tested cells, as defined by implementer.		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst137 :**		
Name	RamTstNumberOfAlgorithms {RAMTST_NUMBER_OF_ALGORITHMS}		
Description	Number of configured algorithms in the container "RamTst_Algorithms" calculationFormula = count of the container RamTst_AlgParams		
Multiplicity	1		
Type	DerivedIntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module dependency: "RamTstNumberOfAlgorithms" is derived by the count of "RamTstAlgParams" which is part of the same subContainer and has a multiplicity of 1 to *.		

SWS Item	**RamTst138 :**
Name	RamTstTestCompletedNotification {RAMTST_TEST_COMPLETED_NOTIFICATION}
Description	This function will be called after finishing the RAM test without an error.
Multiplicity	1

Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst139 :**		
Name	RamTstTestErrorNotification {RAMTST_TEST_ERROR_NOTIFICATION}		
Description	This function will be called if an error occurs during the RAM test.		
Multiplicity	1		
Type	FunctionNameDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
RamTstAlgParams	1..*	This container holds parameters for configuring an algorithm. Each algorithm selected in the RamTst_Algorithms container has a corresponding RamTstAlgParams container. The multiplicity of the included container depends on the number of separate blocks of RAM which are defined for the particular algorithm.

10.2.6 RamTstAlgParams

SWS Item	**RamTst090 :**
Container Name	RamTstAlgParams{RamTst_AlgParams}
Description	This container holds parameters for configuring an algorithm. Each algorithm selected in the RamTst_Algorithms container has a corresponding RamTstAlgParams container. The multiplicity of the included container depends on the number of separate blocks of RAM which are defined for the particular algorithm.
Configuration Parameters	

SWS Item	**RamTst140 :**		
Name	RamTstAlgorithmId {RAMTST_ALGORITHM_ID}		
Description	This is the algorithm ID		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**--**
Name	RamTstExtNumberOfTestedCells {RAMTST_EXT_NUMBER_OF_TESTED_CELLS}
Description	This is the absolute maximum value for the number of cells that NUM-

	BER_OF_TESTED_CELLS and MAX_NUMBER_OF_TESTED_CELLS can be.		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**--**		
Name	RamTstMaxNumberOfTestedCells {RAMTST_MAX_NUMBER_OF_TESTED_CELLS}		
Description	This is the initial value for a RAM variable, which can be changed by the function "RamTst_ChangeMaxNumberOfTestedCells".		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst141 :**		
Name	RamTstNumberOfBlocks {RAMTST_NUMBER_OF_BLOCKS}		
Description	Number of RAM blocks configured using the container "RamTst_BlockParams" calculationFormula = Count of enabled RamTstBlockParams for this algorithm ID.		
Multiplicity	1		
Type	DerivedIntegerParamDef		
Default value	--		
calculationFormula	count of enabled ramtstblockparams for this algorithm id.		
calculationLanguage	informal		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module dependency: "RamTstNumberOfBlocks" is derived by the count of the number of "RamTstBlockParams" containers which are part of the same subcontainer and have a multiplicity of 1 to *.		

SWS Item	**RamTst142 :**		
Name	RamTstNumberOfTestedCells {RAMTST_NUMBER_OF_TESTED_CELLS}		
Description	This is the initial value for a RAM variable, which can be changed by the function "RamTst_ChangeNumberOfTestedCells"		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

Included Containers

Container Name	*Multiplicity*	*Scope / Dependency*
RamTstBlockParams	1..*	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 RamTst_AlgParams for each algorithm selected in RamTst_Algorithms.

10.2.7 RamTstBlockParams

SWS Item	**RamTst091 :**
Container Name	RamTstBlockParams{RamTst_BlockParams}
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 RamTst_AlgParams for each algorithm selected in RamTst_Algorithms.
Configuration Parameters	

SWS Item	**RamTst143 :**		
Name	RamTstBlockId {RAMTST_BLOCK_ID}		
Description	ID of the RAM block		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst144 :**		
Name	RamTstEndAddress {RAMTST_END_ADDRESS}		
Description	End Address of the RAM block		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**RamTst145 :**		
Name	RamTstStartAddress {RAMTST_START_ADDRESS}		
Description	Start Address of the RAM block		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

RamTst080: The integration of incompatible files shall be avoided. At a minimum, the implementation shall include a version check of the header files.

For included header files:
- RAMTST_AR_MAJOR_VERSION
- RAMTST_AR_MINOR_VERSION

shall be identical.

For the module internal c and h files:
- RAMTST_SW_MAJOR_VERSION
- RAMTST_SW_MINOR_VERSION
- RAMTST_AR_MAJOR_VERSION
- RAMTST_AR_MINOR_VERSION
- RAMTST_AR_PATCH_VERSION

shall be identical.

RamTst081: 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
--	--	--	--	--	--	--

10.3 Published Information

The standard common published information like

vendorId (<Module>_VENDOR_ID),
moduleId (<Module>_MODULE_ID),
arMajorVersion (<Module>_AR_MAJOR_VERSION),
arMinorVersion (<Module>_ AR_MINOR_VERSION),
arPatchVersion (<Module>_ AR_PATCH_VERSION),
swMajorVersion (<Module>_SW_MAJOR_VERSION),
swMinorVersion (<Module>_ SW_MINOR_VERSION),
swPatchVersion (<Module>_ SW_PATCH_VERSION),
vendorApiInfix (<Module>_VENDOR_API_INFIX)

is provided in the BSW Module Description Template (see [6] Figure 4.1 and Figure 7.1).

Additional published parameters are listed below if applicable for this module.

11 Changes to Version 1.0.0

11.1 Deleted SWS Items

SWS Item	*Rationale*
RamTst056	Bug 12583: Modified Hamming Code test removed
RamTst064	Bug 14103: contradiction removed
RamTst004	Bug 14242: File include structure, the internal file structure “Algo” is up to the implementer

11.2 Changed SWS Items

SWS Item	*Rationale*
RamTst002	Bug 12356: The term “RAM area” specified more in detail
RamTst008	Bug 12356: The term “RAM area” specified more in detail
RamTst045	Bug 12356: The term “RAM area” specified more in detail
RamTst047	Bug 12356: The term “RAM area” specified more in detail
RamTst060	Bug 12356: The term “RAM area” specified more in detail
RamTst088	Bug 13718: description extended, to make the usage more clear.
RamTst049	Bug 14234: “RamTst_DataType” moved in the configuration area.
RamTst040	Bug 15763: (Range: min = 1 / max = up to the implementer)
RamTst071	Bug 12960: Dem_ReportErrorEvent changed to Dem_ReportErrorStatus
RamTst003	Bug 14242 & 15974: File include structure updated (MemMap.h and SchM_<modul>.h) added

11.3 Added SWS Items

SWS Item	*Rationale*
RamTst094	Bug 15155: behavior of RamTst_GetTestResult
RamTst095	Bug 15763: behavior of the RamTst_ChangeTestAlgorithm() function, in case of a wrong sequence call order described more in detail.

12 Changes during SWS Improvements by Technical Office

12.1 Deleted SWS Items

SWS Item	*Rationale*
RamTst088	SWS Improvement: Requirement not ID necessary, already covered by other requirements

12.2 Changed SWS Items

Many requirements have been changed to improve understandability without changing the technical contents.

12.3 Added SWS Items

SWS Item	*Rationale*
RamTst096	SWS Improvement: Requirement had no ID
RamTst098	UML Model linking of imported types
RamTst099	UML Model linking of RamTst_Init
RamTst100	UML Model linking of RamTst_Stop
RamTst101	UML Model linking of RamTst_Continue
RamTst102	UML Model linking of RamTst_GetExecutionStatus
RamTst103	UML Model linking of RamTst_GetTestResult
RamTst104	UML Model linking of RamTst_GetTestResultPerBlock
RamTst105	UML Model linking of RamTst_ChangeTestAlgorithm
RamTst106	UML Model linking of RamTst_GetTestAlgorithm
RamTst107	UML Model linking of RamTst_ChangeNumberOfTestedCells
RamTst108	UML Model linking of RamTst_GetNumberOfTestedCells
RamTst109	UML Model linking of RamTst_GetVersionInfo
RamTst110	UML Model linking of RamTst_MainFunction
RamTst111	UML Model linking of mandatory interface
RamTst112	UML Model linking of optional interface
RamTst113	UML Model linking of RamTst_Test_Completed_Notification
RamTst114	UML Model linking of RamTst_Error_Notification
RamTst115	Standard requirement for error detection
RamTst116	Standard requirement for DET error notification
RamTst117	Hint RamTst_GetVersionInfo

13 Changes to Version 1.1.2

13.1 Deleted SWS Items

SWS Item	*Rationale*
RamTst001	Missing from document before Technical Office review.
RamTst015	Missing from document before Technical Office review.
RamTst016	Missing from document before Technical Office review.
RamTst017	Missing from document before Technical Office review.
RamTst020	Missing from document before Technical Office review.
RamTst022	Missing from document before Technical Office review.
RamTst023	Missing from document before Technical Office review.
RamTst028	Missing from document before Technical Office review.
RamTst029	Missing from document before Technical Office review.
RamTst030	Missing from document before Technical Office review.
RamTst031	Missing from document before Technical Office review.
RamTst032	Missing from document before Technical Office review.
RamTst035	Missing from document before Technical Office review.
RamTst041	Missing from document before Technical Office review.
RamTst048	Missing from document before Technical Office review.
RamTst057	Missing from document before Technical Office review.
RamTst075	Missing from document before Technical Office review.
RamTst077	Missing from document before Technical Office review.
RamTst092	Missing from document before Technical Office review.

13.2 Changed SWS Items

SWS Item	*Rationale*

13.3 Added SWS Items

SWS Item	*Rationale*
RamTst118	

14 Changes to Version 1.1.5

14.1 Deleted SWS Items

None

14.2 Changed SWS Items

None

14.3 Added SWS Items

SWS Item	*Rationale*
RamTst146	UML Model linking of RamTst_DeInit
RamTst147	Requirement on RamTst_DeInit
RamTst148	Requirement on RamTst_Stop
RamTst149	Requirement on RamTst_Allow
RamTst150	Requirement on RamTst_Suspend
RamTst151	Requirement on RamTst_ChangeMaxNumberOfTestedCells