Document Title	Specification of RAM Test
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Version	1.1.1
Document Status	Final
Part of Release	2.1
Revision	0014

Document Change History			
Date	Version	Changed by	Change Description
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 6
2 Acronyms and abbreviations 7
3 Related documentation 8
3.1 Input documents 8
3.2 Related standards and norms 8
4 Constraints and assumptions 9
4.1 Limitations 9
4.2 Applicability to car domains 9
5 Dependencies to other modules 10
5.1 File structure 10
5.1.1 Code file structure 10
5.1.2 Header file structure 11
6 Requirements traceability 12
7 Functional specification 18
7.1 Requirements 18
7.2 Error classification 18
7.3 Error detection 19
7.4 Error notification 19
8 API specification 20
8.1 Imported types 20
8.1.1 Standard types 20
8.2 Type definitions 20
8.2.1 RamTst_ExecutionStatusType 20
8.2.2 RamTst_TestResultType 20
8.2.3 RamTst_AlgorithmType 21
8.2.4 RamTst_ConfigType 21
8.2.5 RamTst_NumberOfTestedCellsType 21
8.2.6 RamTst_NumberOfBlocksType 21
8.2.7 RamTst_NumberOfAlgorithmsType 21
8.2.8 RamTst_BlockAddressType 22
8.3 Function definitions 22
8.3.1 RamTst_Init 22
8.3.2 RamTst_Stop 22
8.3.3 RamTst_Continue 23
8.3.4 RamTst_GetExecutionStatus 24
8.3.5 RamTst_GetTestResult 24
8.3.6 RamTst_GetTestResultPerBlock 24
8.3.7 RamTst_ChangeTestAlgorithm 25
8.3.8 RamTst_GetTestAlgorithm 26
8.3.9 RamTst_ChangeNumberOfTestedCells 26
8.3.10 RamTst_GetNumberOfTestedCells 27
8.3.11 RamTst_GetVersionInfo 27

8.4 Callback notifications 28
8.5 Scheduled functions 28
8.5.1 RamTst_MainFunction 28
8.6 Expected Interfaces 29
8.6.1 Mandatory Interfaces 29
8.6.2 Optional Interfaces 29
8.6.3 Configurable interfaces 29
8.6.3.1 RamTst Test Completed Notification 30
8.6.3.2 RamTst Error Notfication 30
9 Sequence diagrams 31
9.1 RamTst_MainFunction (Example) 31
9.2 RamTst_GetExecutionStatus 33
9.3 RamTst_GetTestResult 33
9.4 RamTst_GetTestResultPerBlock 34
9.5 RamTst_GetTestAlgorithm 34
9.6 RamTst_ChangeNumberOfTestedCells 35
9.7 RamTst_GetNumberOfTestedCells 35
10 Configuration specification 36
10.1 How to read this chapter 36
10.1.1 Configuration and configuration parameters 36
10.1.2 Containers 36
10.1.3 Specification template for configuration parameters 36
10.2 Containers and configuration parameters 38
10.2.1 Variants 38
10.2.2 Container RamTst_Common 38
10.2.3 Container RamTst_Algorithms 41
10.2.4 Container RamTst_ConfigParams 43
10.2.5 Container RamTst_AlgParams 45
10.2.6 Container RamTst_BlockParams 46
10.3 Published Information 47
11 Changes to release 1.0.0 49
11.1 Deleted SWS Items 49
11.2 Changed SWS Items 49
11.3 Added SWS Items 49

1 Introduction and functional overview

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

There are several RAM Test algorithms available. These RAM Test algorithms are coming from 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%.

The result of the ECU safety analysis states the requirements which RAM Test diagnostic coverage rates (Low, Medium or High) shall be used.

2 Acronyms and abbreviations

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

3 Related documentation

3.1 Input documents

[1] AUTOSAR List of Basic Software Modules
https://svn.autosar.org/repos/10Releases/
AUTOSAR_BasicSoftwareModules.pdf

[2] Layered Software Architecture
https://svn.autosar.org/repos/10Releases/
AUTOSAR_LayeredSoftwareArchitecture.pdf

[3] Specification of Development Error Tracer,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_DET.pdf

[4] General Requirements on Basic Software Modules,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_General.pdf

[5] Specification of ECU Configuration,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_ECU_Configuration.pdf

[6] General Requirements on SPAL
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_SPAL_General.pdf

[7] Requirements on RAM Test
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_RAM_Test.pdf

3.2 Related standards and norms

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

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

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

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

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

4 Constraints and assumptions

4.1 Limitations

RamTst002: During RAM Test the tested Ram area (This the actual test area with the size configured in RAMTST_NUMBER_OF_TESTED_CELLS) shall not be modified because this module can not insure data consistency.(e.g. NMI,DMA)

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

This module could be scheduled in the callouts of the EcuStateManger and in the BSW scheduler.

Test during Startup: could be implemented in the "Callout from STARTUP or WAKEUP"

Test during RUN State: BSW Module cyclic scheduling

Test before Shutdown: could be implemented in the "Callout from SHUTDOWN"

5.1 File structure

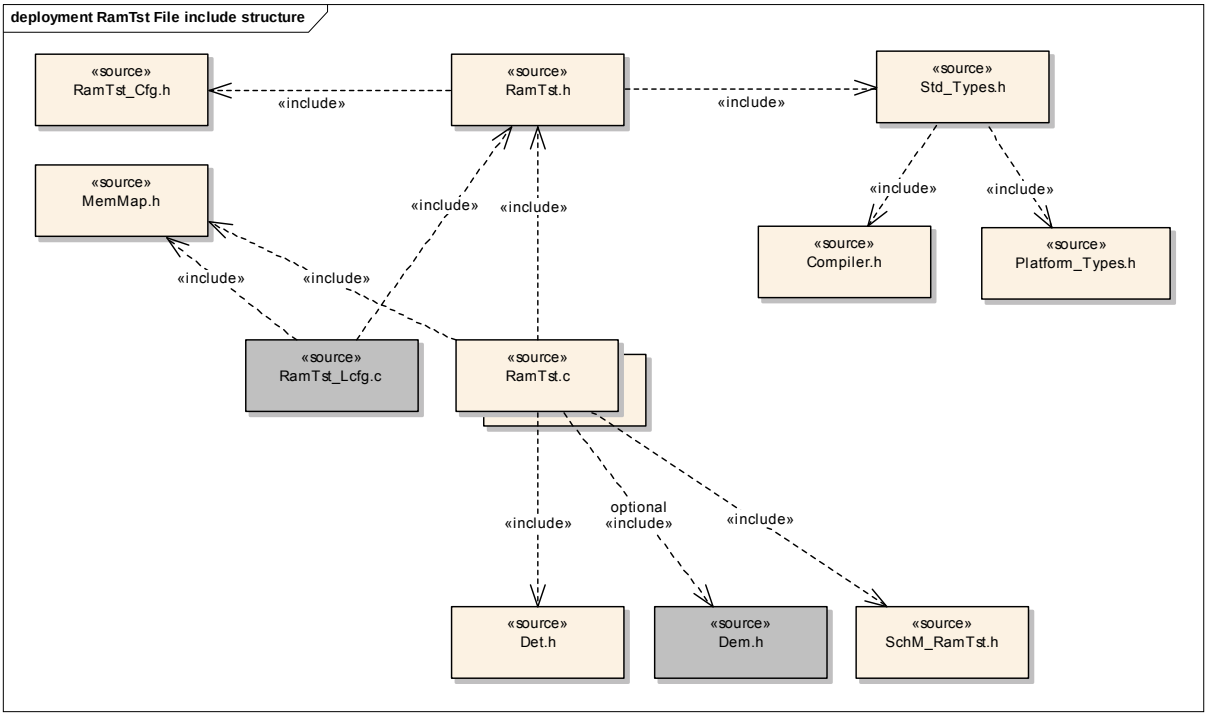
5.1.1 Code file structure

RamTst086: The code file structure 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 file include structure 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 time) shall be placed into the RamTst_Cfg.h file.

6 Requirements traceability

Document: AUTOSAR requirements on Basic Software, general

Requirement	*Satisfied by*
Functional Requirements	
[BSW00323] API parameter checking	RamTst033, RamTst037, RamTst039, RamTst040, RamTst068, RamTst095
[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	RamTst068, RamTst074, RamTst067, RamTst069
[BSW00339] Reporting of production relevant error status	Not applicable (only DEM error events shall be supported)
[BSW00344] Reference to link-time configuration	RamTst025, RamTst026, RamTst027, RamTst066
[BSW00345] Pre-compile-time configuration	RamTst065,RamTst070, RamTst068, RamTst079
[BSW00369] Do not return development error codes via API	RamTst033, RamTst037, RamTst039, RamTst040, RamTst068; 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	RamTst086
[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 (only API check)
[BSW00387] Specify the configuration class of callback function	This version supports only pointer at link time.
[BSW00388] Introduce containers	RamTst066,RamTst065, RamTst070, RamTst090, RamTst091
[BSW00389] Containers shall have names	RamTst066,RamTst065, RamTst070, RamTst090, RamTst091
[BSW00390] Parameter content shall be unique within the module	RamTst066,RamTst065, RamTst070, RamTst090, RamTst091
[BSW00391] Parameter shall have unique names	RamTst066,RamTst065, RamTst070, RamTst090, RamTst091 Prefix “RamTst” added to each parameter
[BSW00392] Parameters shall have a type	RamTst066,RamTst065, RamTst070, RamTst090, RamTst091
[BSW00393] Parameters shall have a range	RamTst066,RamTst065, RamTst070, RamTst090, RamTst091
[BSW00394] Specify the scope of the parameters	RamTst066,RamTst065, RamTst070, RamTst090, RamTst091 “local” marked as Module. (template and SRS General are inconsistent)
[BSW00395] List the required parameters (per parameter)	All parameter in chapter ”Containers and configuration parameters” are required.
[BSW00396] Configuration classes	see chapter “Variants”
[BSW00397] Pre-compile-time parameters	RamTst065, RamTst070
[BSW00398] Link-time parameters	RamTst066, RamTst090, RamTst091
[BSW00399] Loadable Post-build time parameters	Not applicable (post build time is not supported)

[BSW004] Version check	RamTst080, RamTst077
[BSW00400] Selectable Post-build time parameters	Not applicable (post build time is not supported)
[BSW00402] Published information	RamTst077
[BSW00404] Reference to post build time configuration	Not applicable (no use case for that)
[BSW00405] Reference to multiple configuration sets	Not applicable (post build time is not supported)
[BSW00406] Check module initialization	RamTst007, RamTst006
[BSW00407] Function to read out published parameters	RamTst078, RamTst079
[BSW00409] Header files for production code error IDs	Not applicable (this is a requirement for the DEM)
[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	RamTst086
[BSW00420] Production relevant error event rate detection	Not applicable (this is a requirement for the DEM)
[BSW00421] Reporting of production relevant error events	RamTst073, RamTst067, RamTst076, RamTst071,RamTst088
[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 (requirement for the implementer)
[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 has no interrupt routine)
[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 has not send/receive functionality)
[BSW00433] Calling of main processing functions	Conflict, discussion in [Bug 7550]
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (this is a special requirement for the BSW scheduler)
[BSW101] Initialization interface	RamTst007
[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	Requirement on configuration tool
[BSW168] Diagnostic Interface of SW components	Not applicable (not a SW-Component)

Non-functional Requirements	
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable (no configuration and not a SW-Component)
[BSW171] Configurability of optional functionality	RamTst065
[BSW003] Version identification	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 chapter 8.1Imported types
[BSW00305] Self-defined data types naming convention	See Chapter 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)
[BSW00435] Module Header File Structure for the Basic Software Scheduler	RamTst003
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	RamTst003
[BSW00318] Format of module version numbers	See chapter 10.3 Published Information
[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 chapter 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	Not applicable (requirement for the implementer)
[BSW00331] Separation of error and status values	Not applicable (requirement for the implementer)
[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 chapter 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
[BSW00353] Platform specific type header	Not applicable (requirement for the implementer)
[BSW00355] Do not redefine AUTOSAR integer data types	See chapter 8.2 Type definitions
[BSW00357] Standard API return type	See chapter 8.1Imported types
[BSW00358] Return type of init() functions	See chapter 8.3.1 RamTst_Init
[BSW00359] Return type of callback functions	See chapter 8.6.3 Configurable interfaces
[BSW00360] Parameters of callback functions	See chapter 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	See chapter 8.5.1 RamTst_MainFunction
[BSW00374] Module vendor identification	Not applicable (requirement for the implementer)
[BSW00376] Return type and parameters of main processing functions	See chapter 8.5.1 RamTst_MainFunction
[BSW00377] Module specific API return types	See chapter 8.2 Type definitions and chapter 8.1Imported types
[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 chapter 10.2 Containers and configuration parameters
[BSW00410] Compiler switches shall have defined values	See chapter 10.2 Containers and configuration parameters
[BSW00411] Get version info keyword	RamTst079
[BSW00413] Accessing instances of BSW modules	Not applicable, because instances makes no sense for this module
[BSW00414] Parameter of init function	See chapter 8.3.1 RamTst_Init, RamTst093
[BSW00415] User dependent include files	RamTst003 (non-functional requirement)
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable (this is a non-functional requirement)
[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 non-functional requirement)
[BSW162] ECU layout abstraction	Not applicable (this is a non-functional requirement)
[BSW164] Implementation of interrupt service routines	Not applicable (this module has no interrupt service routines)
[BSW172] Compatibility and documentation of scheduling strategy	Not applicable (requirement for the implementer)

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

Requirement	***Satisfied by***
General Requirements	
[BSW12056] Configuration of notification mechanisms	RamTst042,RamTst043,RamTst044, RamTst045, RamTst046, RamTst066
[BSW12057] Driver module initialization	RamTst066, RamTst007, RamTst025, RamTst026, RamTst027
[BSW12059] General initialization of overall registers	Not applicable (this module has does not access other hardware resources than the RAM, no initialization necessary)
[BSW12063] Raw value mode	Not applicable (this module does not provide physical signals)
[BSW12064] Change of operation mode during running operation	Not applicable (requirement for the implementer)
[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 has does not access other hardware resources than the RAM, no initialization necessary)
[BSW12129] Resetting of interrupt flags	Not applicable (there is no interrupt service routine in this module)
[BSW12163] Driver module deinitialization	Not applicable (this module has does not access other hardware resources than the RAM, no deinitialization necessary)
[BSW12169] Control of operation mode	RamTst014, RamTst018, RamTst019
[BSW12263] Object code compatible configuration concept	RamTst066, RamTst025, RamTst026, RamTst027, RamTst007
[BSW12267] Configuration of wake-up sources	Not applicable (this module has no wakeup sources)
[BSW12448] Behavior after development error detection	RamTst033, RamTst037, RamTst039, RamTst040 , RamTst095
[BSW12461] Responsibility for register initialization	Not applicable (this module use no registers)
[BSW12462] Provide settings for register initialization	Not applicable (this module use no registers)
[BSW12463] Combine and forward settings for register initialization	Not applicable (this module use no registers)
[BSW157] Notification mechanisms of drivers and handlers	RamTst042, RamTst043, RamTst044, RamTst045, RamTst046, RamTst066

Non-Functional Requirements	
[BSW12077] Non-blocking implementation	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)
[BSW12265] Configuration data shall be kept constant	Not applicable (requirement for the implementer)
[BSW12264] Specification of configuration items	See chapter 10.2 Containers and configuration parameters

Document: AUTOSAR Requirements on Basic Software, Cluster SPAL, RAM Test Requirements (Chapter 6.1.1) [4]

Requirement	*Satisfied by*
[BSW13800] Number of tested cells shall be changeable at runtime	RamTst036, RamTst034
[BSW13801] Test cell size shall be configurable at pre-compile time	RamTst049, RamTst066
[BSW13802] Multiple RAM areas shall be configurable at post build/ link time	RamTst026, RamTst066
[BSW13803] A subset of available RAM Test algorithms shall be selectable at pre-compile time	RamTst026, RamTst027, RamTst063, RamTst065, RamTst083, RamTst084, RamTst085
[BSW13804] A subset of the pre-compile time selected RAM Check test algorithms shall be selectable at runtime	RamTst063, RamTst058, RamTst065, RamTst021
[BSW13805] Checkerboard test algorithm shall be available	RamTst050
[BSW13807] March test algorithm shall be available	RamTst051
[BSW13806] Walk path test algorithm shall be available	RamTst052
[BSW13808] Galpat test algorithm shall be available	RamTst053
[BSW13818] Transparent Galpat test algorithm shall be available	RamTst054
[BSW13813] Abraham test algorithm shall be available	RamTst055
[BSW13809] RAM Test Execution Management	RamTst014, RamTst018, RamTst019, RamTst008, RamTst009, RamTst010, RamTst011, RamTst059, RamTst001, RamTst002
[BSW13810] Current status of RAM Test execution per block shall be available through a get status interface	RamTst019, RamTst024, RamTst038
[BSW13820] RAM Test execution status shall be provided by a notification mechanism.	RamTst042, RamTst043, RamTst044, RamTst045, RamTst046
[BSW13811] None Destructive RAM Test	RamTst060, RamTst061
[BSW13812] Destructive RAM Test	RamTst061
[BSW13816] Effects of Instruction / Data queue shall be taken into account	RamTst062
[BSW13821] The RAM Test Module shall be designed to fulfill SIL3 Requirements	RamTst050, RamTst051, RamTst052, RamTst053, RamTst054, RamTst055, RamTst056

7 Functional specification

7.1 Requirements

RamTst005: The RAM Test shall be designed as an asynchronous service.

RamTst063: From the different RAM Test algorithms (see following requirements: RamTst050, RamTst051, RamTst052, RamTst053, RamTst054, RamTst055RamTst056) only a subset may be chosen. The subset shall be selectable during pre-compile time.

RamTst060: If non-destructive RAM Test is chosen, the tested RAM area (This is the actual test area with the size configured in RAMTST_NUMBER_OF_TESTED_CELLS) shall be saved before it will be modified. The complete procedure (saving, changing, restoring) shall be done within interrupt lock.

RamTst061: For destructive and non-destructive option it shall be assured that the RAM Test internal variables are not overwritten by the test algorithm.

RamTst062: After writing to a cell and before reading back it shall be possible to inject instruction(s) to force the controller to clear its CPU internal cache.

RamTst050: A checkerboard test algorithm shall be implemented as stated in [12] , A.5.1

RamTst051: A March test algorithm shall be implemented as stated in [12], A.5.1

RamTst052: A WalkPath test algorithm shall be implemented as stated in [12], A.5.2

RamTst053: A Galpat test algorithm shall be implemented as stated in [12], A.5.3

RamTst054: A Transparent Galpat test algorithm shall be implemented as stated in, [12], A.5.3

RamTst055: An Abraham test algorithm shall be implemented as stated in [12], 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]*
Failure within RAM Test execution status	Development	RAMTST_E_STATUS_FAILURE	0x01
API parameter out of specified range	Development	RAMTST_E_OUT_OF_RANGE	0x02
API service used without module initialization	Development	RAMTST_E_UNINIT	0x03
RAM Failure	Production	RAMTST_E_RAM_FAILURE	Assigned by DEM

7.3 Error detection

RamTst068: The detection of all development errors shall be configurable (on/off) with the preprocessor switch RAMTST_DEV_ERROR_DETECT.
Detected development errors shall be reported to the error hook of the Development Error Tracer (DET) if the preprocessor switch RAMTST_DEV_ERROR_DETECT is set.

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

RamTst089: The service RamTst_Init() shall be called first before calling any other RAM test services. If not respected, the error code RAMTST_E_UNINIT will 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 RamTst067.

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

8 API specification

8.1 Imported types

8.1.1 Standard types

In this chapter, all types included from the following files are listed:

- Std_Types.h
  - Std_ReturnType
  - Std_VersionInfoType

8.2 Type definitions

8.2.1 RamTst_ExecutionStatusType

Type:	Enumeration	
Range:	`RAMTST_EXECUTION_UNINIT`	The RAM Test is not initialized or not usable. **RamTst006**: This shall be the default value after reset. This status shall have the value 0.
	`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_CONTINUE`	The RAM Test is ready for continue.
	`RAMTST_EXECUTION_STATE_UNDEFINED`	State used for white box testing.
Description:	This is a status value returned by the API service RamTst_GetExecutionStatus().	

8.2.2 RamTst_TestResultType

Type:	Enumeration	
Range:	`RAMTST_RESULT_NOT_TESTED`	The RAM Test is not executed. **RamTst012**: This shall be the default value after reset. This status shall have the value 0.
	`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`	This should never happen. Just for redundancy.
Description:	This is a status value returned by the API service RamTst_GetTestResult().	

8.2.3 RamTst_AlgorithmType

Type:	Enumeration	
Range:	`RAMTST_ALGORITHM_UNDEFINED`	Undefined algorithm (uninitialized value) **RamTst013**: This shall be the default value after reset. RAMTST_ALGORITHM_UNDEFINED shall have the value 0.
	`RAMTST_CHECKERBOARD_TEST`	Checkerboard test algorithm, see RamTst050
	`RAMTST_MARCH_TEST`	March test algorithm, see RamTst051
	`RAMTST_WALK_PATH_TEST`	Walk path test algorithm, see RamTst052
	`RAMTST_GALPAT_TEST`	Galpat test algorithm, see RamTst053
	`RAMTST_TRANSP_GALPAT_TEST`	Transparent Galpat test algorithm, see RamTst054
	`RAMTST_ABRAHAM_TEST`	Abraham test algorithm, see RamTst055
Description:	This is a value returned by the API service RamTst_GetTestAlgorithm(). **RamTst058**: The RamTst_AlgorithmType shall contain only the enumerations of the algorithms selected at pre-compile time.	

8.2.4 RamTst_ConfigType

Type:	Struct
Range:	Implementation specific
Description:	**RamTst025**: This type of the external data structure shall contain the initialization data for the RAM Test **RamTst026**: A RAM Test algorithm specific configuration container shall include at least the following parameters: algorithm type, number of tested cells, number of tested RAM blocks, RAM blocks start and end addresses **RamTst027**: Every RAM Test algorithm shall contain at least one own configuration container for all RAM Test specific configuration data which are defined in RamTst026

8.2.5 RamTst_NumberOfTestedCellsType

Type:	uint8 …uint32 (implementation specific)
Range:	$1..2^n-1$
Description:	Data type of number of tested RAM cells

8.2.6 RamTst_NumberOfBlocksType

Type:	uint8..uint32 (implementation specific)
Range:	$1..2^n-1$
Description:	Data type of number of blocks

8.2.7 RamTst_NumberOfAlgorithmsType

Type:	uint8..uint32 (implementation specific)

Range:	1..2^n-1
Description:	Data type of number of configured container for algorithms

8.2.8 RamTst_BlockAddressType

Type:	uint8..uint32 (implementation specific)
Range:	The range of this type is µC specific and has to be described by the supplier.
Description:	Data type of block address pointer

8.3 Function definitions

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

8.3.1 RamTst_Init

Service name:	RamTst_Init
Syntax:	`void RamTst_Init ( void )`
Service ID [hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	non reentrant
Parameters (in):	None --
Parameters (out):	None --
Return value:	None --
Description:	**RamTst007**: Service for RAM Test initialization. The initialization function shall initialize all RAM Test relevant registers and global variables. Change execution status to RAMTST_EXECUTION_INIT. The RAM Test starts at the first call of the “RamTst_MainFunction”.
Caveats:	This service shall not be called during a running operation.
Configuration:	Container “RamTst_ConfigParams”

8.3.2 RamTst_Stop

Service name:	RamTst_Stop
Syntax:	`void RamTst_Stop ( void )`
Service ID [hex]:	0x02
Sync/Async:	Asynchronous
Reentrancy:	non reentrant

Parameters (in):	None --
Parameters (out):	None --
Return value:	None --
Description:	**RamTst014**: Service for stopping the RAM Test. If this service is called while a call of "RamTst_MainFunction()" is active, this call shall be finished even if the execution status is changed to RAMTST_EXECUTION_STOPPED by this service. **RamTst033**: If the DET Is enabled then a development error shall be issued when the execution status of the RAM Test is not in the status RAMTST_EXECUTION_RUNNING, the desired function will be skipped and the function will be left without any action. Related error value: RAMTST_E_STATUS_FAILURE.
Caveats:	None
Configuration:	Parameter "RAMTST_STOP_API" in the Container RamTst_Common.

8.3.3 RamTst_Continue

Service name:	RamTst_Continue
Syntax:	`void RamTst_Continue ( void )`
Service ID [hex]:	0x03
Sync/Async:	Asynchronous
Reentrancy:	non reentrant
Parameters (in):	None --
Parameters (out):	None --
Return value:	None --
Description:	**RamTst018**: Service for continuing the RAM Test after calling "RamTst_Stop" . Change execution status to RAMTST_EXECUTION_CONTINUE. **RamTst037**: If the DET Is enabled then a development error shall be issued when the execution status of the RAM Test is not in the status RAMTST_EXECUTION_STOPPED, the desired function will be skipped and the function will be left without any action. Related error value: RAMTST_E_STATUS_FAILURE.
Caveats:	None
Configuration:	Parameter "RAMTST_CONTINUE_API" in the Container RamTst_Common.

8.3.4 RamTst_GetExecutionStatus

Service name:	RamTst_GetExecutionStatus
Syntax:	`RamTst_ExecutionStatusType RamTst_GetExecutionStatus ( void )`
Service ID [hex]:	0x04
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None --
Parameters (out):	None --
Return value:	RamTst_ExecutionStatusType See type definition
Description:	**RamTst019**: The service shall return the current RAM Test execution status
Caveats:	None
Configuration:	Parameter “RAMTST_GET_EXECUTION_STATUS_API” in the Container RamTst_Common.

8.3.5 RamTst_GetTestResult

Service name:	RamTst_GetTestResult
Syntax:	`RamTst_TestResultType RamTst_GetTestResult ( void )`
Service ID [hex]:	0x05
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None --
Parameters (out):	None --
Return value:	RamTst_TestResultType See type definition
Description:	**RamTst024**: The service shall return the current RAM Test result.
Caveats:	None
Configuration:	Parameter “RAMTST_GET_TEST_RESULT_API” in the Container RamTst_Common.

8.3.6 RamTst_GetTestResultPerBlock

Service name:	RamTst_GetTestResultPerBlock
Syntax:	`RamTst_TestResultType RamTst_GetTestResultPerBlock (`

	`RamTst_NumberOfBlocksType BlockID` `)`
Service ID [hex]:	0x06
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	`BlockID` See type definition RamTst_NumberOfBlocksType
Parameters (out):	`None` --
Return value:	`RamTst_TestResultType` See type definition
Description:	**RamTst038**: The service shall return the current RAM Test result for the specified block. **RamTst039**: If development errors are enabled then a development error shall be issued when the BlockID is out of range. TestResult RAMTST_RESULT_STATE_UNDEFINED shall be returned. Related error value: RAMTST_E_OUT_OF_RANGE
Caveats:	None
Configuration:	Parameter “RAMTST_GET_TEST_RESULT_PER_BLOCK_API” in the Container RamTst_Common.

8.3.7 RamTst_ChangeTestAlgorithm

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 (out):	`None` --
Return value:	None --
Description:	**RamTst083**: The service shall change the used TestAlgorithm, initialized by the RamTst_Init() function with the configuration parameter “DefaultTestAlgorithm”. **RamTst085**: This service shall re-initialize all RAM Test relevant registers and global variables with the values for the “NewTestAlgorithm”. **RamTst084**: If development errors are enabled then a development error shall be issued when the parameter NewTestAlgorithm is out of range. The used TestAlgorithm shall remain unchanged. Related error value: RAMTST_E_OUT_OF_RANGE. A development error shall be issued when the execution status of the RAM Test is not in the status RAMTST_EXECUTION_STOPPED, the desired function will be skipped and the function will be left without any action. Related error value: RAMTST_E_STATUS_FAILURE. **RamTst094:** The test result status (RamTst_TestResultType) shall not be

	changed. The result of the previous test is still valid.
Caveats:	None
Configuration:	The default value has to be configured in the RamTst_ConfigType: Container: RamTst_ConfigParams Parameter: DefaultTestAlgorithm Parameter "RAMTST_CHANGE_TEST_ALGORITHM_API" in the Container RamTst_Common.

8.3.8 RamTst_GetTestAlgorithm

Service name:	RamTst_GetTestAlgorithm
Syntax:	`RamTst_AlgorithmType RamTst_GetTestAlgorithm ( void )`
Service ID [hex]:	0x07
Sync/Async:	Synchronous
Reentrancy:	non reentrant
Parameters (in):	None --
Parameters (out):	None --
Return value:	`RamTst_AlgorithmType` See type definition
Description:	**RamTst021**: The service shall return the current RAM Test algorithm ID
Caveats:	None
Configuration:	Parameter "RAMTST_GET_TEST_ALGORITHM_API" in the Container RamTst_Common.

8.3.9 RamTst_ChangeNumberOfTestedCells

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 RamTst_NumberOfTestedCellsType
Parameters (out):	None --
Return value:	None --
Description:	**RamTst036**: The service shall change the current number of tested cells. **RamTst040**: If development errors are enabled then a development error shall be issued when the parameter NewNumberOfTestedCells is out of range (Range: min=1 / max = up to the implementer).The number of tested cells shall remain unchanged. Related error value: RAMTST_E_OUT_OF_RANGE.

	RamTst095: If the execution status of the RAM Test is not in the status RAMTST_EXECUTION_STOPPED, the operation shall be skipped and a development error RAMTST_E_STATUS_FAILURE shall be reported.
Caveats:	None
Configuration:	The start value has to be configured in the RamTst_ConfigType: Container: RamTst_AlgParams Parameter: NumberOfTestedCells Parameter "RAMTST_CHANGE_NUMBER_OF_TESTED_CELLS_API" in the Container RamTst_Common.

8.3.10 RamTst_GetNumberOfTestedCells

Service name:	RamTst_GetNumberOfTestedCells	
Syntax:	`RamTst_NumberOfTestedCellsType RamTst_GetNumberOfTestedCells ( void )`	
Service ID [hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	None	--
Parameters (out):	None	--
Return value:	RamTst_NumberOfTestedCellsType	Number of currently tested cells of RAM Test will be returned.
Description:	**RamTst034**: Service for reading the current number of tested cells per cycle.	
Caveats:	None	
Configuration:	Parameter "RAMTST_GET_NUMBER_OF_TESTED_CELLS_API" in the Container RamTst_Common.	

8.3.11 RamTst_GetVersionInfo

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 (out):	versioninfo	Pointer to the location / address where to store the version information of this module.
Return value:	None	--
Description:	**RamTst078**: This service returns the version information of this module. The version information includes: Module Id	

	Vendor Id Vendor specific version numbers (BSW00407). **RamTst079**: This function shall be configurable at pre compile time by the configuration parameter: RAMTST_VERSION_INFO_API values On/Off Hint: If source code for caller of this function is available, this function should be realized as a macro. The macro should be defined in the module header file.
Caveats:	None
Configuration:	Parameter “RAMTST_VERSION_INFO_API” in the Container RamTst_Common.

8.4 Callback notifications

There are no callback notifications in this module.

8.5 Scheduled functions

Basic Software Scheduler directly calls these functions. 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 will be called when conditions are fulfilled. Alternatively, the function may be called cyclically however the cycle time will be assigned dynamically during runtime by other modules.

8.5.1 RamTst_MainFunction

Service name:	RamTst_MainFunction
Service ID [hex]:	0x01
Description:	**RamTst008**: Service for executing the RAM Test. This service shall test the defined RAM blocks, starting with the first RAM block in the RamTst_ConfigParams. **RamTst009**: The service shall set the RAM Test execution status from RAMTST_EXECUTION_INIT or RAMTST_EXECUTION_CONTINUE to RAMTST_EXECUTION_RUNNING when calling the first time after initialization or after calling RamTst_Stop. **RamTst010**: The service shall set the RAM Test result status of every block to RAMTST_RESULT_OK or RAMTST_RESULT_NOT_OK depending of the result of the test. **RamTst011**: The service shall set the overall result status to RAMTST_RESULT_OK if all blocks are tested with OK result .If at least one block

	test result was RAMTST_RESULT_NOT_OK then the overall test result status shall be set to RAMTST_RESULT_NOT_OK even all blocks are already tested or not. In case of test result RAMTST_RESULT_NOT_OK the production error RAMTST_E_RAM_FAILURE shall be issued to the DEM. **RamTst047**: When all RAM blocks are completely tested, the test shall restart from the beginning at the next call of this service. **RamTst059**: With one call the service shall test a defined number of RAM cells. This number shall be given by the service RamTst_ChangeNumberOfTestedCells or by initialization.
Timing:	variable cyclic / on pre condition (depends on the used algorithm and system requirements)
Pre condition:	See requirements RamTst001 and RamTst002
Configuration:	None

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.

API function	*Module*	*Description*
Dem_ReportErrorStatus	Dem	**RamTst088**: Used for error detection in production code. Here for reporting a ram check failure. The "EventStatus" will be 'DEM_EVENT_STATUS_FAILED' in case of an RAM error. Other states are not allowed, because a self-healing mechanism makes no sense in this situation.

8.6.2 Optional Interfaces

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

API function	*Module*	*Description*	*Configuration parameter (description see chapter 10)*
Det_ReportError	Det	Development error notification	RAMTST_DEV_ERROR_DETECT

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, no callback shall be executed.

8.6.3.1 RamTst Test Completed Notification

Name:	RamTst Test Completed Notification
Syntax:	`void <Function Name>(void)`
Reentrancy:	don't care
Parameters (in):	`None` --
Parameters (out):	`None` --
Return value:	`None` --
Description:	**RamTst045**: The function RamTst_TestCompleted shall be called every time when all RAM blocks had been tested.
Caveats:	None
Configuration:	Container RamTst_ConfigParams Parameter " RAMTST_TEST_COMPLETED_NOTIFICATION"

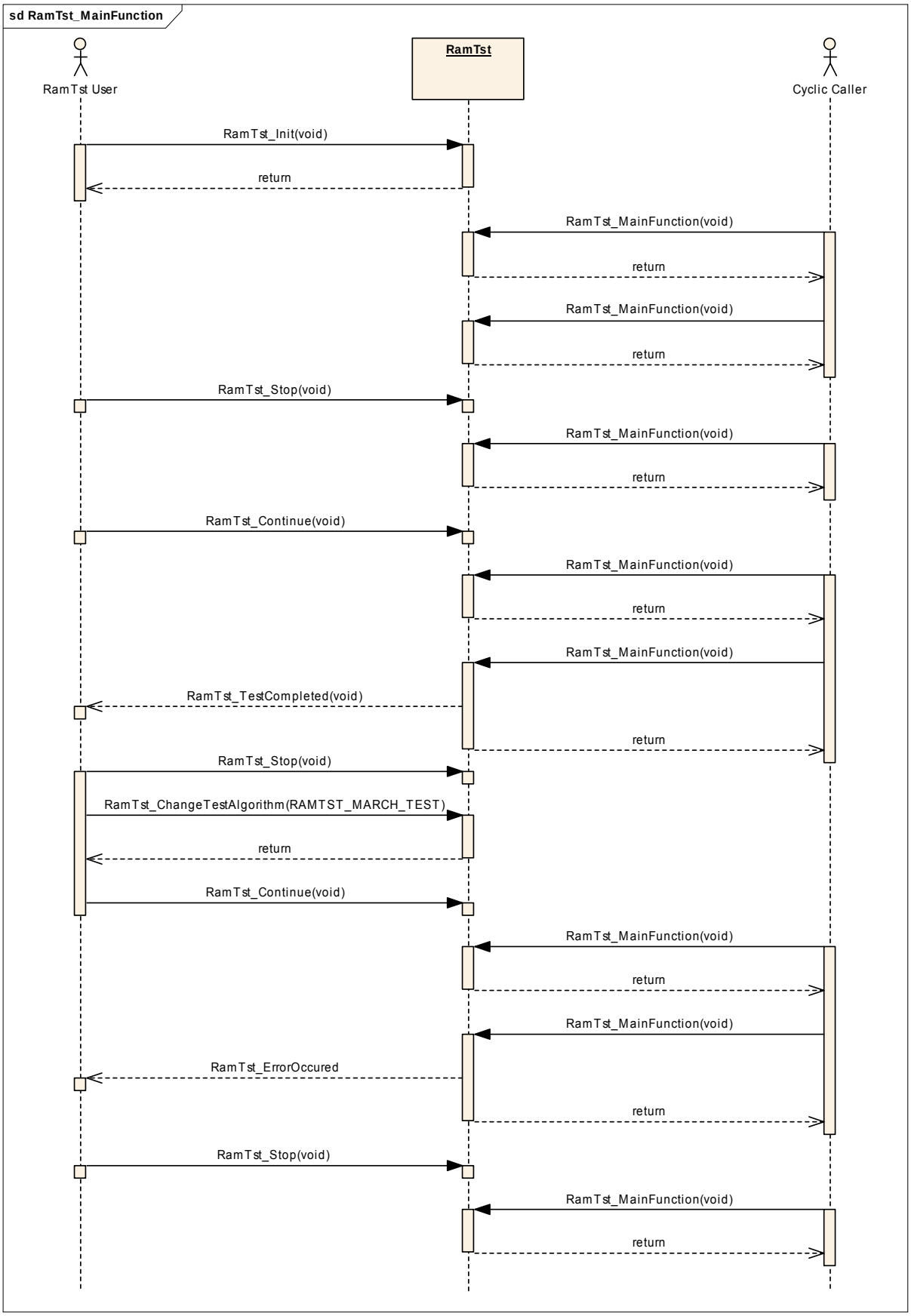
8.6.3.2 RamTst Error Notfication

Name:	RamTst Error Notification
Syntax:	`void <Function Name>(void)`
Reentrancy:	don't care
Parameters (in):	`None` --
Parameters (out):	`None` --
Return value:	`None` --
Description:	**RamTst046**: The function RamTst_Error shall be called every time when a RAM failure has been detected by the selected algorithm.
Caveats:	None
Configuration:	Container RamTst_ConfigParams Parameter " RAMTST_TEST_ERROR_NOTIFICATION"

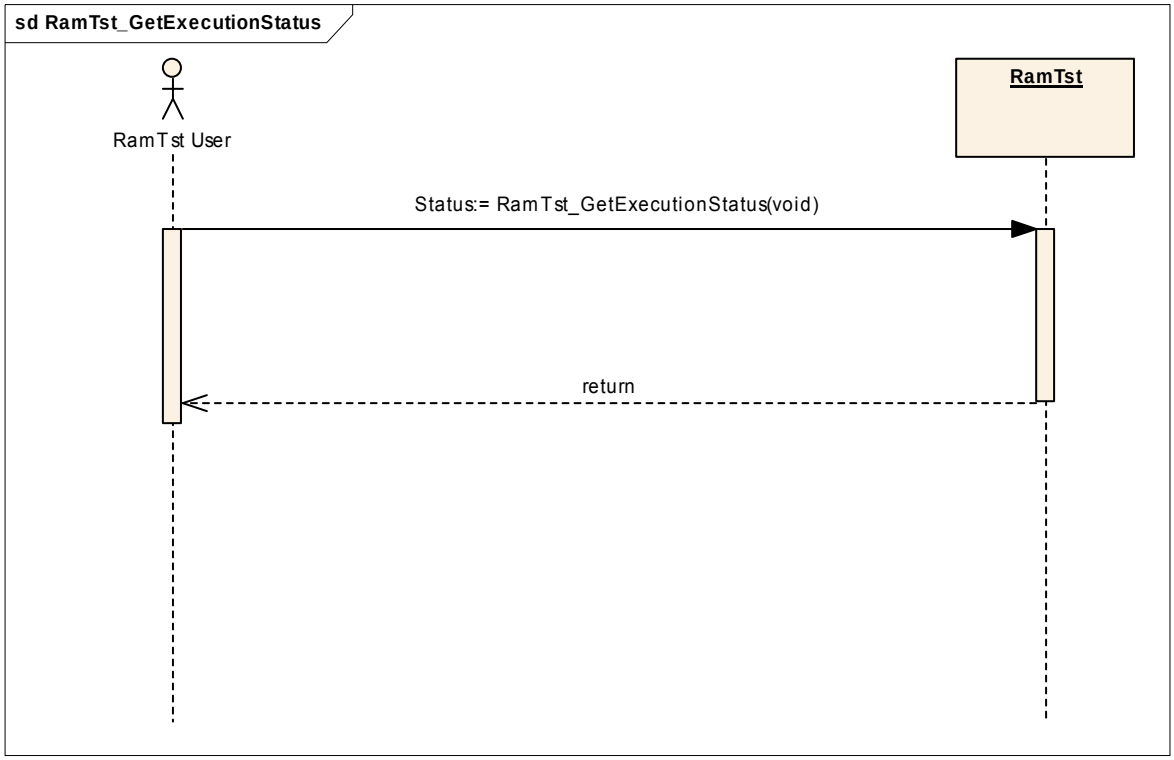
9 Sequence diagrams

9.1 RamTst_MainFunction (Example)

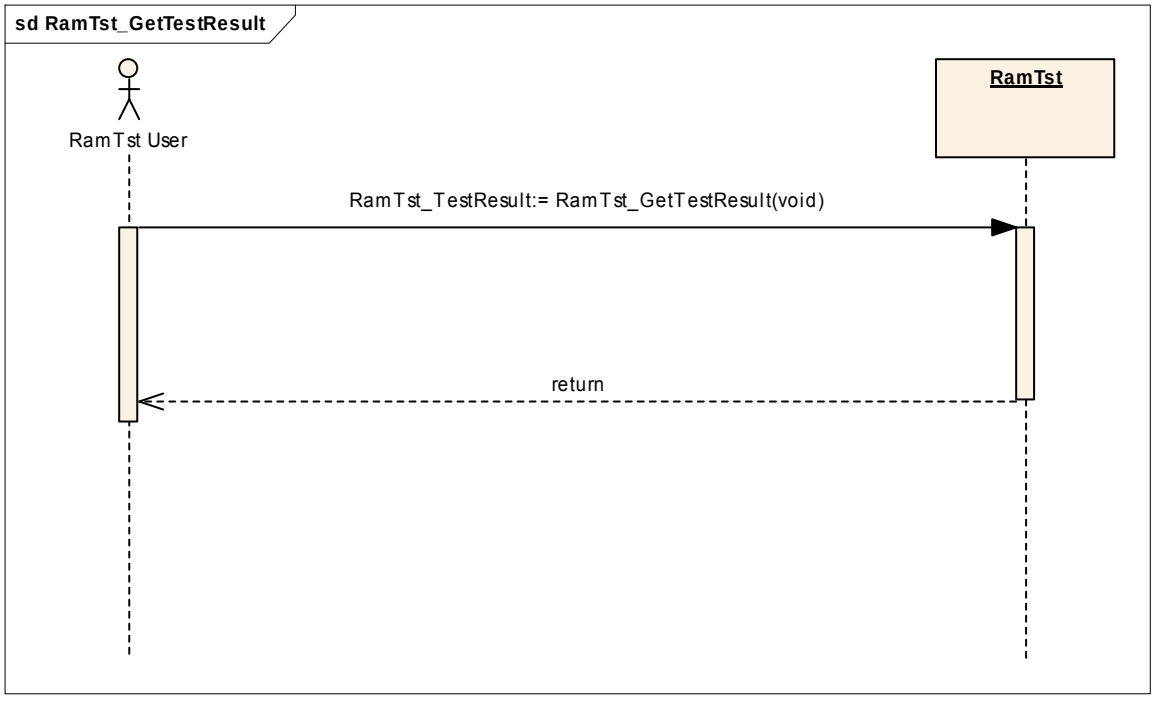
The following sequence initializes the RAM Test. Following this, the cyclic caller will call the RAM Test main routine cyclically. In between the user requests asynchronously a stop/continue of the RAM Test. The stop/continue of the RAM Test will be handled within the next call of the main routine.
After receiving the “TestCompleted” callback the user changes the TestAlgorithm and continuous the RAM Test with the new algorithm.
During the test of this configuration, an error has been detected and the user stops the ram test.


sd RamTst_MainFunction
RamTst User
RamTst
Cyclic Caller
RamTst_Init(void)
return
RamTst_MainFunction(void)
return
RamTst_MainFunction(void)
return
RamTst_Stop(void)
RamTst_MainFunction(void)
return
RamTst_Continue(void)
RamTst_MainFunction(void)
return
RamTst_MainFunction(void)
RamTst_TestCompleted(void)
return
RamTst_Stop(void)
RamTst_ChangeTestAlgorithm(RAMTST_MARCH_TEST)
return
RamTst_Continue(void)
RamTst_MainFunction(void)
return
RamTst_MainFunction(void)
RamTst_ErrorOccured
return
RamTst_Stop(void)
RamTst_MainFunction(void)
return

9.2 RamTst_GetExecutionStatus

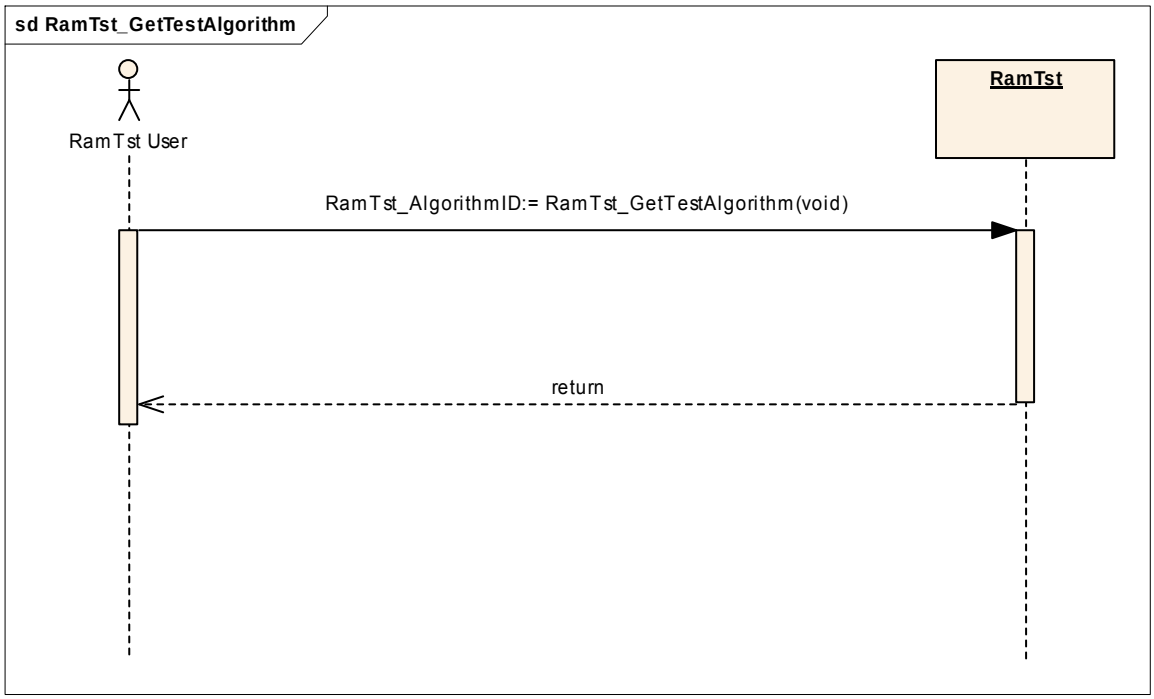


9.3 RamTst_GetTestResult

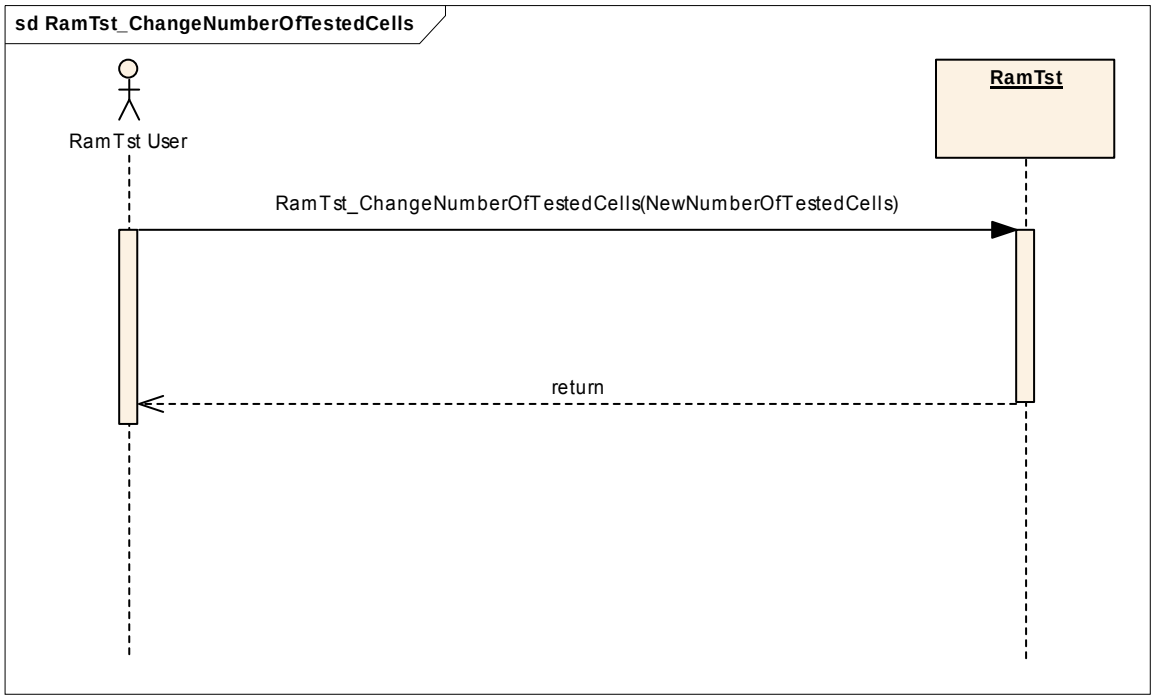


9.4 RamTst_GetTestResultPerBlock

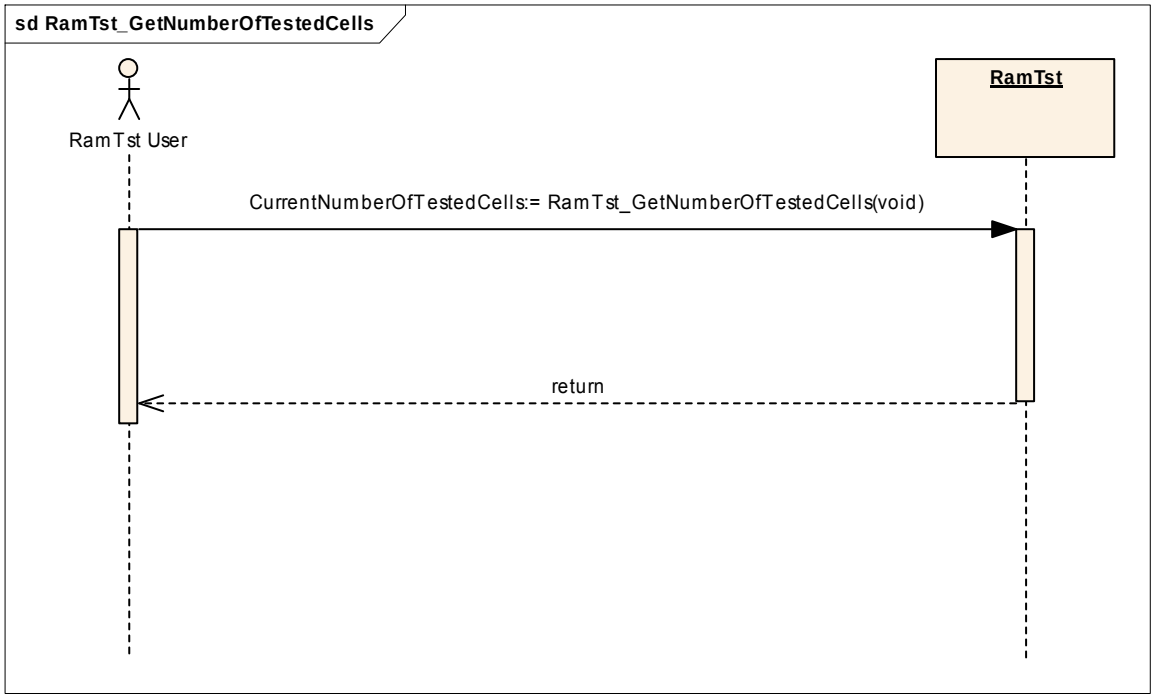
9.5 RamTst_GetTestAlgorithm



9.6 RamTst_ChangeNumberOfTestedCells



9.7 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
- 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 pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class *Post Build*.

10.2 Containers and configuration parameters

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

10.2.1 Variants

Variant PC: This variant is limited to pre-compile-configuration parameters only. The intention of this variant is to optimize the parameters configuration for a source code delivery.

Variant LT: This variant allows a mix of pre-compile time-, link time-configuration parameters. The intention of this variant is to optimize the parameters configuration for an object code delivery.

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 Container RamTst_Common

SWS Item	**RamTst070**: The following table specifies the common RAM Test parameters that shall be selected at pre-compile time in the module’s configuration file (RamTst_Cfg.h)
Container Name	RamTst_Common
Description	This container contains all configuration parameter for the RamTest selectable at pre-compile time.
Configuration Parameters	

Name	RAMTST_DEV_ERROR_DETECT		
Description	Preprocessor switch to select the development error tracer (DET) ON or OFF		
Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAMTST_VERSION_INFO_API
Description	Preprocessor switch to disable / enable the function “RamTst_GetVersionInfo”

Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	Disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAMTST_STOP_API		
Description	Preprocessor switch to disable / enable the function “RamTst_Stop”		
Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAMTST_CONTINUE_API		
Description	Preprocessor switch to disable / enable the function “RamTst_Continue”		
Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAMTST_GET_EXECUTION_STATUS_API		
Description	Preprocessor switch to disable / enable the function “RamTst_GetExecutionStatus”		
Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAMTST_GET_TEST_RESULT_API		
Description	Preprocessor switch to disable / enable the function “RamTst_GetTestResult”		
Type	#define		
Unit	--		
Range	STD_ON	enabled	

	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAMTST_GET_TEST_RESULT_PER_BLOCK_API		
Description	Preprocessor switch to disable / enable the function “RamTst_GetTestResultPerBlock”		
Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAMTST_CHANGE_TEST_ALGORITHM_API		
Description	Preprocessor switch to disable / enable the function “RamTst_ChangeTestAlgorithm”		
Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAMTST_GET_TEST_ALGORITHM_API		
Description	Preprocessor switch to disable / enable the function “RamTst_GetTestAlgorithm”		
Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAMTST_CHANGE_NUMBER_OF_TESTED_CELLS_API		
Description	Preprocessor switch to disable / enable the function “RamTst_ChangeNumberOfTestedCells”		
Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--

	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAMTST_GET_NUMBER_OF_TESTED_CELLS_API		
Description	Preprocessor switch to disable / enable the function “RamTst_GetNumberOfTestedCells”		
Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
RamTst_Algorithms	1	Selection of RAM Test algorithms.
RamTst_ConfigParams	1	This container is used for the “RamTst_Init” function. The multiplicity of the included container depends on the pre-compile configuration in the container “RamTst_Algorithms“

10.2.3 Container RamTst_Algorithms

SWS Item	**RamTst065**: The following table specifies the selectable algorithms
Container Name	RamTst_Algorithms
Description	A subset of supported RAM Test algorithms could be chosen in this container.
Configuration Parameters	

Name	RAMTST_CHECKERBOARD_TEST_SELECTED		
Description	Preprocessor switch to select the checkerboard test ON or OFF		
Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAMTST_MARCH_TEST_SELECTED
Description	Preprocessor switch to select the march test

	ON or OFF		
Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAMTST_WALK_PATH_TEST_SELECTED		
Description	Preprocessor switch to select the walking path test ON or OFF		
Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAMTST_GALPAT_TEST_SELECTED		
Description	Preprocessor switch to select the galpat test ON or OFF		
Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAMTST_TRANSP_GALPAT_TEST_SELECTED		
Description	Preprocessor switch to select the transparent galpat test ON or OFF		
Type	#define		
Unit	--		
Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAMTST_ABRAHAM_TEST_SELECTED		
Description	Preprocessor switch to select the abraham test ON or OFF		
Type	#define		
Unit	--		

Range	STD_ON	enabled	
	STD_OFF	disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	RAM_TEST_DATA_TYPE		
Description	RamTst049: Data type of tested RAM cells shall be configurable at pre-compile time.		
Type	#define		
Unit	--		
Range	uint8, uint16 or uint32		
	--	--	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

10.2.4 Container RamTst_ConfigParams

SWS Item	**RamTst066**: The following container specifies the configuration parameters that shall be available at link time.
Container Name	RamTst_ConfigParams
Description	The multiplicity of the included container depends on the pre-compile configuration in the container “RamTst_Algorithms“
Configuration Parameters	

Name	RAMTST_NUMBER_OF_ALGORITHMS		
Description	Number of configured algorithms in the container “RamTst_Algorithms”		
Type	RamTst_NumberOfAlgorithmsType		
Unit	--		
Range	RamTst_NumberOfAlgorithmsType	Follow link and see type definition	
	--	--	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	X	Variant LT
	Post Build	--	
Scope	Module		
Dependency	” RAMTST_NUMBER_OF_ALGORITHMS” is a derived parameter and is the count of the container “RamTst_AlgParams”.		

Name	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“		
Type	RamTst_AlgorithmType		
Unit	--		
Range	RamTst_AlgorithmType	Follow link and see type definition	
	--	--	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	X	Variant LT
	Post Build	--	
Scope	Module		
Dependency	None		

Name	RAMTST_TEST_COMPLETED_NOTIFICATION		
Description	This function will be called after finishing the ram test without an error.		
Type	Function pointer		
Unit	--		
Range	--	--	
	--	--	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	X	Variant LT
	Post Build	--	
Scope	Module		
Dependency	None		

Name	RAMTST_TEST_ERROR_NOTIFICATION		
Description	This function will be called if an error occurs during the ram test.		
Type	Function pointer		
Unit	--		
Range	--	--	
	--	--	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	X	Variant LT
	Post Build	--	
Scope	Module		
Dependency	None		

Included Containers			
Container Name	***Multiplicity***	***Scope***	***Dependency***
RamTst_AlgParams	1..*	Module	--

10.2.5 Container RamTst_AlgParams

SWS Item	**RamTst090**:
Container Name	RamTst_AlgParams
Description	This is the parameter container for an algorithm.
Configuration Parameters	

Name	RAMTST_ALGORITHM_ID		
Description	This is the algorithm ID		
Type	RamTst_AlgorithmType		
Unit	--		
Range	RamTst_AlgorithmType	Follow link and see type definition	
	--	--	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	X	Variant LT
	Post Build	--	
Scope	Module		
Dependency	None		

Name	RAMTST_NUMBER_OF_TESTED_CELLS		
Description	This is the initial value for a RAM variable which could be changed by the function “RamTst_ChangeNumberOfTestedCells“		
Type	RamTst_NumberOfTestedCellsType		
Unit	--		
Range	RamTst_NumberOfTestedCellsType	Follow link and see type definition	
	--	--	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	X	Variant LT
	Post Build	--	
Scope	Module		
Dependency	None		

Name	RAMTST_NUMBER_OF_BLOCKS		
Description	Number of ram blocks configured in the container “RamTst_BlockParams”		
Type	RamTst_NumberOfBlocksType		
Unit	--		
Range	RamTst_NumberOfBlocksType	Follow link and see type definition	
	--	--	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	X	Variant LT
	Post Build	--	
Scope	Module		
Dependency	” RAMTST_NUMBER_OF_BLOCKS” is a derived parameter and is the count of the container “RamTst_BlockParams”.		

Included Containers			
Container Name	**Multiplicity**	**Scope**	**Dependency**
RamTst_BlockParams	1..*	Module	--

10.2.6 Container RamTst_BlockParams

SWS Item	**RamTst091**:
Container Name	RamTst_BlockParams
Description	This container is used in the " RamTst_AlgParams" container.
Configuration Parameters	

Name	RAMTST_BLOCK_ID		
Description	ID of the RAM block		
Type	RamTst_NumberOfBlocksType		
Unit	--		
Range	RamTst_NumberOfBlocksType	Follow link and see type definition	
	--	--	
Configuration Class	**Pre-compile**	X	Variant PC
	Link time	X	Variant LT
	Post Build	--	
Scope	Module		
Dependency	None		

Name	RAMTST_START_ADDRESS		
Description	Start Address of the RAM block		
Type	RamTst_BlockAddressType		
Unit	--		
Range	RamTst_BlockAddressType	Follow link and see type definition	
	--	--	
Configuration Class	**Pre-compile**	X	Variant PC
	Link time	X	Variant LT
	Post Build	--	
Scope	Module		
Dependency	None		

Name	RAMTST_END_ADDRESS		
Description	End Address of the RAM block		
Type	RamTst_BlockAddressType		
Unit	--		
Range	RamTst_BlockAddressType	Follow link and see type definition	
	--	--	
Configuration Class	**Pre-compile**	X	Variant PC
	Link time	X	Variant LT
	Post Build	--	
Scope	Module		
Dependency	None		

Included Containers			
Container Name	***Multiplicity***	***Scope***	***Dependency***
--	--	--	--

10.3 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

SWS Item	**RamTst077**: Published information of this module	
Information elements		
Information element name	***Type / Range***	***Information element description***
`RAMTST_VENDOR_ID`	`#define/ uint16`	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
`RAMTST_MODULE_ID`	`#define/ uint8`	Module ID of this module from Module List
`RAMTST_AR_MAJOR_VERSION`	`#define/ uint8`	Major version number of AUTOSAR specification where the appropriate implementation is based on.
`RAMTST_AR_MINOR_VERSION`	`#define/ uint8`	Minor version number of AUTOSAR specification where the appropriate implementation is based on.
`RAMTST_AR_PATCH_VERSION`	`#define/ uint8`	Patch level version number of AUTOSAR specification where the appropriate implementation is based on.
`RAMTST_SW_MAJOR_VERSION`	`#define/ uint8`	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
`RAMTST_SW_MINOR_VERSION`	`#define/ uint8`	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
`RAMTST_SW_PATCH_VERSION`	`#define/ uint8`	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

RamTst080: The integration of incompatible files shall be avoided. Minimum implementation is the version check of the header file.

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: Provide a collection of practice-oriented measured / calculated runtime information for each algorithm implementation:

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

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

11 Changes to release 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: behaviour of RamTst_GetTestResult
RamTst095	Bug 15763: behavior of the RamTst_ChangeTestAlgorithm() function, in case of a wrong sequence call order described more in detail.