RAM Test user guide

TRAVEO™ T2G family

About this document

Scope and purpose

This guide describes the architecture, configuration and use of RAM Test. This guide also explains the functionality of the driver, and provides a reference to the driver API.

The installation, build process, and general information about the use of the EB tresos Studio are not within the scope of this document. See the *EB tresos Studio for ACG8 user's guide* **[7]** for detailed information on these topics.

Intended audience

This document is intended for anyone using the RAMTST software of the TRAVEO™ T2G family.

Document structure

Chapter **1 General overview** gives a brief introduction to RAM Test, explains how it is implemented in the AUTOSAR environment and describes the supported hardware and development environment.

Chapter **2 Using RAM Test** provides detailed steps required to use RAM Test in your application.

Chapter **3 Structure and dependencies** describes the file structure and the dependencies for RAM Test.

Chapter **4 EB tresos Studio configuration interface** describes the driver's configuration with EB tresos Studio software.

Chapter **5 Functional description** provides a functional description of all services offered by RAM Test.

Chapter **6 Hardware resources** describes the hardware resources used by RAM Test.

The **Appendix** provides a complete API reference and access register table.

Abbreviations and definitions

Table 1 Abbreviations

Abbreviations	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
ASIL	Automotive Safety Integrity Level
Basic Software (BSW)	Standardized part of software which does not fulfill a vehicle functional job.
DET	Default Error Tracer
DEM	Diagnostic Event Manager
uC	Microcontroller
OS	Operating System
MCAL	Microcontroller Abstraction Layer

Abbreviations	Description
MCU	Microcontroller Unit
CD	Clock Domain
PD	Power Domain
EB tresos ECU AUTOSAR Suite	A collection of AUTOSAR Basic Software modules and a Runtime Environment integrated in a common configuration and build environment.
EB tresos Studio	Elektrobit Automotive configuration framework.
Foreground Test	Foreground test is a synchronous test, are not to be an interrupt. It is called via a call to the user application.
Background Test	RAM Tests that are executed by the regular processing.
HAL	The code for the hardware operation of RAM Test of program.
ECC	Code to correct data errors in a computer (by checking the data at the time of reloading added code to each unit of data stored in memory).
SP	SP register of processor
The smallest unit	Basic test unit that make up RAM Test (foreground test, background test).
All RAM area	All RAM block area that is in the current configuration selected algorithm parameter set.
RAM cell	Basic unit of memory to be tested (1 cell = 32bits).
1 cycle	The call once to `RamTst_MainFunction` by regular treatment.
NMI	Non Maskable Interrupt

Related documents

AUTOSAR requirements and specifications

Bibliography

[1] General specification of basic software modules, AUTOSAR release 4.2.2.

[2] Specification of RAM Test, AUTOSAR release 4.2.2.

[3] Specification of standard types, AUTOSAR release 4.2.2.

[4] Specification of ECU configuration parameters, AUTOSAR release 4.2.2.

[5] Specification of default error Tracer, AUTOSAR release 4.2.2.

[6] Specification of diagnostics event manager, AUTOSAR release 4.2.2.

Elektrobit automotive documentation

Bibliography

[7] EB tresos Studio for ACG8 user's guide.

Hardware documentation

The hardware documents are listed in the delivery notes.

Related standards and norms

Bibliography

[8] Layered software architecture, AUTOSAR release 4.2.2.

Table of contents

Table of contents

1 General overview

1.1 Introduction to RAM Test

RAM Test is a self-diagnostic program intended to detect failure in the RAM of a microcontroller. RAM Test runs the diagnostic by setting the following parameters:

- Test algorithm
  - Checkerboard
  - March
  - Unique
  - Galpat
  - Walk path
  - Transparent galpat
  - Abraham
- Diagnostic start address
- Diagnostic end address
- BackupArea (stack data saving area/user data saving area)
- Temporary stack area (used for stack area diagnosis)
- Diagnostic implementation minimum size

By setting the above parameters, foreground and background tests are available. Foreground test will be called from the user application in a test to be carried out from `RamTst_RunFullTest` or `RamTst_RunPartialTest`. Also, background test executes from `RamTst_MainFunction` that is registered in the scheduler.

Test module conforms to AUTOSAR standard and is executed according to AUTOSAR *specification of RAM Test* [2].

1.2 User profile

This guide is intended for users with a basic knowledge of the following:

- Automotive embedded systems
- C programming language
- AUTOSAR standard
- Target hardware architecture

1.3 Embedding in the AUTOSAR environment

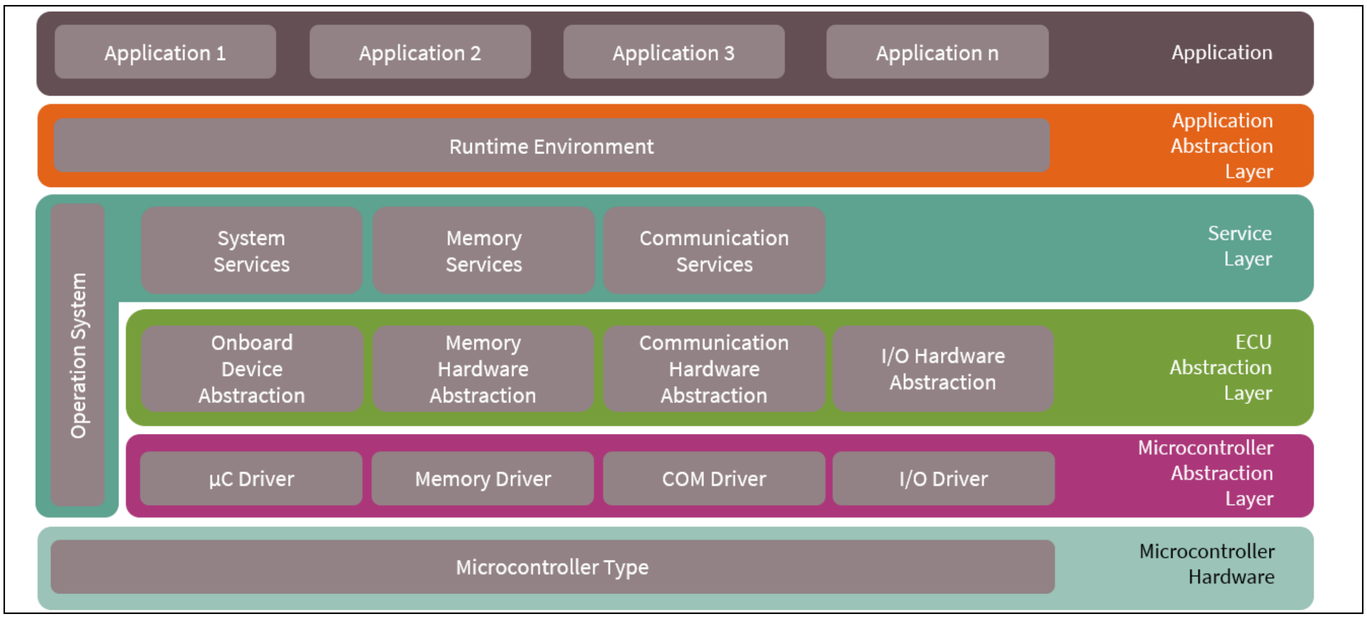

Figure 1 Overview of AUTOSAR software layers

Figure 1 shows the layered AUTOSAR software architecture. RAM Test (see **Figure 2**) is part of the microcontroller abstraction layer (MCAL), the lowest layer of Basic Software in the AUTOSAR environment.

For an overview of the AUTOSAR layered software architecture, see *layered software architecture* **[8]**.

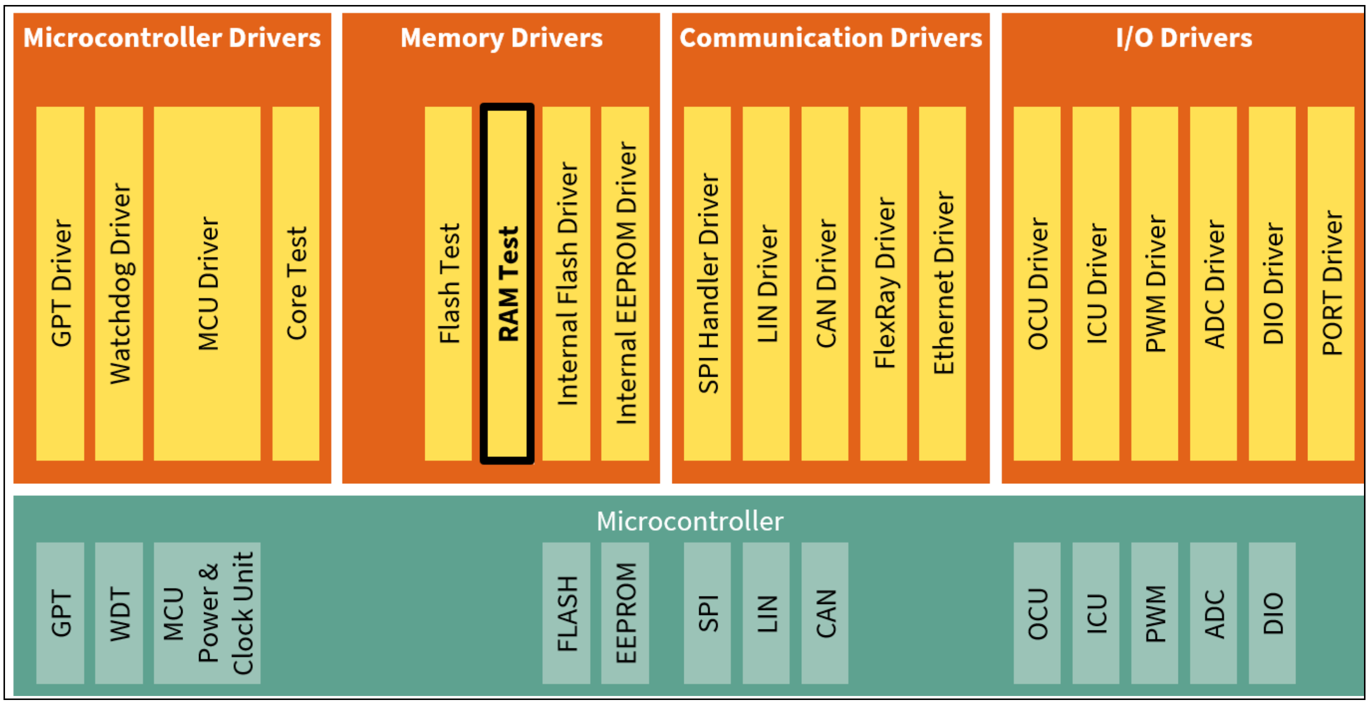

Figure 2 RAM Test in MCAL layer

1.4 Supported hardware

This RAM Test supports TRAVEO™ T2G SRAM area.

1.5 Development environment

The development environment corresponds to AUTOSAR release 4.2.2. The modules Base, Platforms, Make, and Resource are needed for proper functionality of RAM Test.

1.6 Character set and encoding

All source code files of RAM Test are restricted to the ASCII character set. The files are encoded in UTF-8 format, with only the 7-bit subset (values 0x00 # 0x7F) being used.

2 Using RAM Test

2.1 Installation and prerequisites

Note: *Before continuing with this chapter, see the EB tresos Studio for ACG8 user's guide* **[7]** *first. This provides required basic information about the installation procedure of EB tresos ECU AUTOSAR components and the usage of the EB tresos Studio and the EB tresos ECU AUTOSAR build environment. An explanation of how to set up and integrate your own application within the EB tresos ECU AUTOSAR build environment.*

The installation of RAM Test corresponds with the general installation procedure of EB tresos AUTOSAR components given in the documents mentioned above. If the driver has been installed successfully, the driver will appear in the module list of the EB tresos Studio (see *EB tresos Studio for ACG8 user's guide* **[7]**).

This document assumes that the project is properly set up and is using your application template as described in the *EB tresos Studio for ACG8 user's guide* **[7]**. This template provides the necessary folder structure, project, and makefiles needed to configure and compile an application within the build environment. You also must be familiar with the use of the command shell.

2.2 Configuring RAM Test

This section provides an overview of the settings required to use RAM Test.

For EB tresos, configure the following items:

1. Select diagnostic algorithm: Select the diagnostic method of the target area.
2. Configure diagnostic target area: Set the start address and end address of the diagnostic target area.
3. Manage destructive/non-destructive setting: Set whether to restore the diagnostic target data after diagnosis.
4. Set up data backup area: Set the area to back up the data of the diagnosis target area during diagnosis.
5. Set up RAMTST static area: Set the area to store the information necessary for implementing RAMTST.
6. Set up stack area at diagnosis: When the stack area is the diagnosis target area, set the stack area for temporary use.
7. Configure error report output: Select ON or OFF for the notification function, which is used for error or NMI handler.

Note: *If the Notification function is not filled in* `Config`*, the notification function is turned OFF.*

8. Configure WatchDog clear function: Set up the function to clear watchdog timer.
9. Configure ErrorCallOut function: When the default error and the production error occur, define the function of the notification of the error code.
10. Configure EnterExclusiveArea function: Define the function that starts the processing a critical section.
11. Configure ExitExclusiveArea function: Define the function that ends the processing of a critical section.

In addition, in the Linker definition file, set the data backup area, RamTst static area, temporary stack area, or all areas by ensuring consistency with the area set by EB tresos.

Note: *The correspondence between the item set by EB tresos and the section to be described in the Linker definition file (LD file) is as follows:*
EB tresos vs. Linker definition File

- `RamTstStaticAreaStartAddr` vs. either one of the top address of `.ramtstVStaticArea_data` or `.ramtstVStaticArea_bss (*)`
- `RamTstStackBackUpAreaStartAddr` vs. `.ramtstStackBackupArea`
- `RamTstDataBackUpAreaStartAddr` vs. `.ramtstDataBackupArea`

(*) These areas should be located in the continuous area.

You can execute diagnosis while using the RAM area.

For more information of the configuration, see chapter **4 EB tresos Studio configuration interface**.

2.2.1 Architecture specifics

- `RamTstErrorCalloutFunction`: Specifies an error callout handler, which is called when any errors are detected during runtime.
- `RamTstIncludeFile`: Specifies the filename used to include some definitions (e.g. declaration for error callout handler).

2.3 Adapting your application

To use the RAM Test in your application, include the *RamTst.h* by adding the following lines of code in your source file:

```
#include "RamTst.h" /* RamTst Header */
```

This publishes all needed functions or data prototypes and symbolic names of the configuration to your application.

Initialization of `RamTst` can be done as follows:

```
RamTst_Init();Initialize RAM Test module
```

`RamTst_Init` is called first before any other RAM Test functions are called, except `RamTst_GetVersionInfo`.

After initialization is completed, it is possible to execute the foreground test by calling one of the following functions:

```
RamTst_RunFullTest(); All the RAM area is diagnosed
RamTst_RunPartialTest(BlockId); Setting the area is diagnosed
```

To get the diagnosis results, call either of the following function:

```
RamTst_GetTestResult();Get the test results of all the RAM area
RamTst_GetTestResultPerBlock (BlockId);Get the test results of the RAM
block specified
```

The following functions access the backup area for the RAM data if NMI occurred during diagnosis. These functions are available `RamTst_NMIStop` returns `RAMTST_NMI_VARIABLE_ACCESS_FORBID`

```
RamTst_ReadGlobalVarInNMI(ReadAddress, AccessSize, ReadDataBuffer);
RamTst_WriteGlobalVarInNMI(WriteAddress, AccessSize, WriteDataBuffer);
```

The above functions are available only if the RAM data belongs to the target diagnosis area.

2.3.1 Defining sections

Green Hills Software:

Add a section to the Green Hills linker directives (*.ld*) file for the Green Hills Software (GHS) build environment.

- `.ramtstVStaticArea_data`: Data_section area used only by `RAMTST`
- `.ramtstVStaticArea_bss`: Bss_section area used only by `RAMTST`
- `.ramtstStackBackupArea`: Stack area used as temporary Stack area at diagnosis
- `.ramtstDataBackupArea`: Area to save data during diagnosis
- `.RamTst_SecAsmCode`: Section to place RAMTST assembler source

Code Listing 1 Configuration example

```
// The data segment
//
    .ramtstVStaticArea_data  0x08004000  MIN_SIZE(80)    : > SRAM
    .ramtstVStaticArea_bss   0x08004050  MIN_SIZE(48)    : > .      ← Settings
    .ramtstStackBackupArea   0x08004080  MIN_SIZE(600) : > .        required for
    .ramtstDataBackupArea    0x080042D8  MIN_SIZE(256) : > .        RAMTST
//
// The text segment
//
    .intvec                              ALIGN(256)    : > .
    .picbase                                           : > .
    .text                                              : > .
    .intercall                                         : > .
    .interfunc                                         : > .
    .fixaddr                                           : > .
    .fixtype                                           : > .
    .secinfo                                           : > .
    .robase                                            : > .
    .rodata                                            : > .
    .ROM.syscall                   ROM(.syscall)       : > .
    .ROM.data                       ROM(.data)         : > .     ← Settings
    .ROM.ramtstVStaticArea   ROM(.ramtstVStaticArea_data) : > .    required for
  .RamTst_SecAsmCode                                   : > .     RAMTST
//
```

Note: *`.ramtstVStaticArea_data`, `.ramtstVStaticArea_bss`, `.ramtstStackBackupArea`, and `.ramtstDataBackupArea` are necessary for the successful operation of the RAM Test. Exclude these areas from the diagnostic area.*

The sizes of `.ramtstVStaticArea_data` section, `.ramtstVStaticArea_bss` section, `.ramtstDataBackupArea`, and `.ramtstStackBackupArea` sections are larger than or equal to EB tresos configuration.

- The configuration of ".ramtstDataBackupArea" will be from `RamTstDataBackUpAreaStartAddr` to `RamTstDataBackUpAreaSize`.
- The configuration of ".ramtstStackBackupArea" will be from `RamTstStackBackUpAreaStartAddr` to `RamTstStackBackUpAreaSize` .
- The names of ".ramtstStackBackupArea" section and ".ramtstDataBackupArea" section can be changed as required

IAR Embedded Workbench:

Add a section to the IAR *icf* file for the IAR build environment.

- `ramtstVStaticArea_region`: Data_section area used only by `RAMTST`
- `ramtstStackBackupArea_region`: Stack area used as temporary stack area at diagnosis
- `ramtstDataBackupArea_region`: Area to save data during diagnosis
- `RamTst_SecAsmCode_region`: Section to place RAMTST assembler source

Note: *`ramtstVStaticArea_region`, `ramtstStackBackupArea_region`, and `ramtstDataBackupArea_region` are necessary for the successful operation of the RAM Test. Exclude these areas from the diagnostic area.*

The sizes of `ramtstVStaticArea_region` section, `ramtstDataBackupArea_region`, and `ramtstStackBackupArea_region` sections are larger than or equal to EB tresos configuration.

- The configuration of "ramtstDataBackupArea`_region` " will be from `RamTstDataBackUpAreaStartAddr` to `RamTstDataBackUpAreaSize`.
- The configuration of "ramtstStackBackupArea`_region` " will be from `RamTstStackBackUpAreaStartAddr` to `RamTstStackBackUpAreaSize` .

2.4 Starting the build process

Do the following to build your application:

Note: *For a clean build, use the build command with target `clean_all` before (`make clean_all`).*

1. On the command shell, type the following command to generate the necessary configuration-dependent files. See. **3.3 Generated files**.
   ```
   > make generate
   ```
2. Type the following command to resolve required file dependencies:
   ```
   > make depend
   ```
3. Type the following command to compile and link the application:
   ```
   > make (optional target: all)
   ```

The application is now built. All files are compiled and linked to a binary file which can be downloaded to the target hardware.

2.5 Memory mapping

The *RamTst_MemMap.h* file in the *$(TRESOS_BASE)/plugins/MemMap_TS_T40D13M0I0R0/include* directory is a sample. This file is replaced by the file generated by MEMMAP module. Input to MEMMAP module is generated as *RamTst_Bswmd.arxml* in the *$(PROJECT_ROOT)/ output/generated/swcd* directory of your project folder.

2.5.1 Memory allocation keyword

- `RAMTST_START_SEC_CODE_ASIL_B` / `RAMTST_STOP_SEC_CODE_ASIL_B`

  The memory section type is CODE. All executable code is allocated in this section.

- `RAMTST_START_SEC_CONST_ASIL_B_UNSPECIFIED` / `RAMTST_STOP_SEC_CONST_ASIL_B_UNSPECIFIED`

  The memory section type is CONST. All constants are allocated in this section.

- `RAMTST_START_SEC_VAR_INIT_ASIL_B_UNSPECIFIED` / `RAMTST_STOP_SEC_VAR_INIT_ASIL_B_UNSPECIFIED`

  The memory section type is VAR. All initialized data is allocated in this section.

- `RAMTST_START_SEC_VAR_STATIC_AREA_ASIL_B` / `RAMTST_STOP_SEC_VAR_STATIC_AREA_ASIL_B`

  The memory section type is VAR. All RamTst exclusive parameter is allocated in this section.

3 Structure and dependencies

RAM Test consists of static, configuration, and generated files.

3.1 Static files

- $(PLUGIN_PATH)=$(TRESOS_BASE)/plugins/RamTst_TS_* is the path to RAM Test module plugin.
- *$(PLUGIN_PATH)/lib_src* contains all static source files of RAM Test. These files represent the functionality of the driver. Therefore, they are independent of any configuration sets. The files are packed together into a static library.
- *$(PLUGIN_PATH)/src* contains configuration dependent source files or special derivative files. Each file will be rebuilt when the configuration set is changed.

All necessary source files will be automatically compiled and linked during the build process and all include paths will be set if RAM Test is enabled.

- *$(PLUGIN_PATH)/include* is the basic public include directory that is required and should include *RamTst.h*.
- *$(PLUGIN_PATH)/lib_include* contains the internal header files for the Ram Test.
- *$(PLUGIN_PATH)/autosar* directory contains the AUTOSAR ECU parameter definition with vendor, architecture, and derivative specific adaptations to create a correct matching parameter configuration for RAM Test module.

3.2 Configuration files

The configuration of RAM Test is done with the EB tresos Studio. When saving a project, the configuration description is written to the *RamTst.xdm* file and is in the *$(PROJECT_ROOT)/config* directory in your project folder. This file serves as input for the generation of the configuration dependent source and header files during the build process.

3.3 Generated files

During the build process, the following files are generated based on the current configuration description. They are all located in the *output/generated* sub folder of the project folder.

- *include/RamTst_Cfg.h* provides all symbolic names of the configuration and is included by *RamTst.h*.
- *include/RamTst_ExternalInclude.h* contains the external include file information.
- *src/RamTst_Cfg.c* contains the structure of the RamTst_ConfigParams/ RamTst_BlockParams/ RamTst_AlgParams.

Note: *Generated source files need not to be added to your application make file. They will be compiled and linked automatically during the build process.*

- *swcd/RamTst_Bswmd.arxml* is an implementation information for customer or RTE vendor.

Note: *Additional steps are required for the generation of BSW module description. In EB tresos Studio, follow the menu path **Project** > Build Project and click generate_swcd.*

3.4 Dependencies

3.4.1 DET

If the default error detection is enabled in RAM Test module configuration, the DET needs to be installed, configured, and integrated into your application.

3.4.2 DEM

If the RAM failure notification is enabled in RAM Test module configuration, the DEM needs to be installed, configured, and integrated into your application.

3.4.3 OS interrupts

The configured OS is needed for make ISR vector table entries of RAM Test. For details, see chapter 6.

3.4.4 BSW scheduler

RAM Test uses the following services of the BSW scheduler to enter and leave critical sections:

- `RAMTST_ENTER_EXCLUSIVE_AREA_FUNC` (This label is replaced with the function name set at configuration.)
- `RAMTST_EXIT_EXCLUSIVE_AREA_FUNC` (This label is replaced with the function name set at configuration.)

You must ensure that the BSW Scheduler is properly configured and initialized before using RAM Test.

3.4.5 Error callout handler

The error callout handler is called on every error that is detected, regardless of whether default error detection is enabled or disabled. The error callout handler is an ASIL safety extension that is not specified by AUTOSAR. It is configured via configuration parameter `RamTstErrorCallOutFunction`.

4 EB tresos Studio configuration interface

The GUI is not part of this delivery. For further information see *EB tresos Studio for ACG8 user's guide* [7].

4.1 General configuration

This section lists all configuration parameters required by AUTOSAR. Deviations or hardware specific restrictions from the specification are mentioned where applicable.

4.2 Containers and configuration parameters

4.2.1 RamTstCommon

This container holds a list of all available functions in the RAM Test module. Each function is turned ON or OFF before compiling, so that only the desired functions and test algorithms are in the compiled code.

4.2.1.1 RamTstDevErrorDetect

Description

Select the default error tracer (DET).

Remarks

None

Range

TRUE or FALSE

4.2.1.2 RamTstErrorCallOutFunction

Description

It will be called to report error to the callout handler when DET and DEM errors occur.

Remarks

When default and production errors occur, the function of the notification of the error code is defined.

Range

Correct function name. (For example, `error_callout_example`)

4.2.1.3 RamTstStopApi

Description

Adds or removes the service `RamTst_Stop` from the code.

Remarks

None

Range

TRUE or FALSE

4.2.1.4 RamTstAllowApi

Description

Adds or removes the service `RamTst_Allow` from the code.

Remarks

None

Range

TRUE or FALSE

4.2.1.5 RamTstSuspendApi

Description

Adds or removes the service `RamTst_Suspend` from the code.

Remarks

None

Range

TRUE or FALSE

4.2.1.6 RamTstResumeApi

Description

Adds or removes the service `RamTst_Resume` from the code.

Remarks

None

Range

TRUE or FALSE

4.2.1.7 RamTstGetExecutionStatusApi

Description

Adds or removes the service `RamTst_GetExecutionStatus` from the code.

Remarks

None

Range

TRUE or FALSE

4.2.1.8 RamTstGetTestResultApi

Description

Adds or removes the service `RamTst_GetTestResult` from the code.

Remarks

None

Range

TRUE or FALSE

4.2.1.9 RamTstGetTestResultPerBlockApi

Description

Adds or removes the service `RamTst_GetTestResultPerBlock` from the code.

Remarks

None

Range

TRUE or FALSE

4.2.1.10 RamTstGetVersionInfoApi

Description

Adds or removes the service `RamTst_GetVersionInfo` from the code.

Remarks

None

Range

TRUE or FALSE

4.2.1.11 RamTstGetAlgParamsApi

Description

Adds or removes the service `RamTst_GetAlgParams` from the code.

Remarks

None

Range

TRUE or FALSE

4.2.1.12 RamTstGetTestAlgorithmApi

Description

Adds or removes the service `RamTst_GetTestAlgorithm` from the code.

Remarks

None

Range

TRUE or FALSE

4.2.1.13 RamTstGetNumberOfTestedCellsApi

Description

Adds or removes the service `RamTst_GetNumberOfTestedCells` from the code.

Remarks

None

Range

TRUE or FALSE

4.2.1.14 RamTstSelectAlgParamsApi

Description

Adds or removes the service `RamTst_SelectAlgParams` from the code.

Remarks

None

Range

TRUE or FALSE

4.2.1.15 RamTstChangeNumOfTestedCellsApi

Description

Adds or removes the service `RamTst_ChangeNumOfTestedCells` from the code.

Remarks

None

Range

TRUE or FALSE

4.2.1.16 RamTstRunFullTestApi

Description

Adds or removes the service `RamTst_RunFullTest` from the code.

Remarks

None

Range

TRUE or FALSE

4.2.1.17 RamTstRunPartialTestApi

Description

Adds or removes the service `RamTst_RunPartialTest` from the code.

Remarks

None

Range

TRUE or FALSE

4.2.1.18 RamTstIncludeFile

Description

List of header files that are included in the configuration file.

Remarks

None

Range

Correct header file name. (For example, *abc_example.h*)

4.2.1.19 RamTstDebuggingSupport

Description

Enable or disable debugging support.

Remarks

Defines whether to inject a fault into RAM for debugging.

Range

TRUE or FALSE

4.2.1.20 RamTstEnterExclusiveAreaFunc

Description

Starts an exclusive area.

Remarks

Defines the function that starts the processing of a critical section

Range

Correct function name. (For example, `enter_exclusive_example`)

4.2.1.21 RamTstExitExclusiveAreaFunc

Description

Ends an exclusive area.

Remarks

Defines the function that ends the processing of a critical function.

Range

Correct function name. (For example, `exit_exclusive_example`)

4.2.1.22 RamTstWatchDogClearFunc

Description

Defines the function name for clear watchdog timer specified by the user.

Remarks

Defines the function that processes WatchDog clear.

Range

Correct function name. (For example, `watchdog_clear_example`)

4.2.1.23 BackupAreaDiagnosisMethod

Description

Defines how to check the backup data when testing the target RAM.

Remarks

Normally, select `Check_periodical`. Select `Check_at_once` only when the runtime request is strict.

Range

`Check_periodical` or `Check_at_once`

4.2.2 RamTstAlgorithms

The container sets the availability of each test algorithm.

4.2.2.1 RamTstAbrahamTestSelected

Description

Specifies whether Abraham test is available.

Remarks

None.

Range

TRUE or FALSE

4.2.2.2 RamTstCheckerboardTestSelected

Description

Specifies whether Checkerboard test is available.

Remarks

None.

Range

TRUE or FALSE

4.2.2.3 RamTstMarchTestSelected

Description

Specifies whether March test is available.

Remarks

None

Range

TRUE or FALSE

4.2.2.4 RamTstWalkPathTestSelected

Description

Specifies whether WalkPath test is available.

Remarks

None

Range

TRUE or FALSE

4.2.2.5 RamTstGalpatTestSelected

Description

Specifies whether Galpat test is available.

Remarks

None

Range

TRUE or FALSE

4.2.2.6 RamTstTranspGalpatTestSelected

Description

Specifies whether Transparent Galpat test is available.

Remarks

None

Range

TRUE or FALSE

4.2.2.7 RamTstUniqueTestSelected

Description

Specifies whether Unique test is available.

Remarks

None

Range

TRUE or FALSE

4.2.3 RamTstConfigParams

This container specifies the configuration parameters which are set during pre-compilation or at link time.

4.2.3.1 RamTstDefaultAlgParamsId

Description

This is the identifier for the default `RamTstAlgParams` that is valid after `RamTst_Init`. It is the initial value for a RAM variable that can be changed by `RamTst_SelectAlgParams`.

Remarks

None

Range

1…255

4.2.3.2 RamTstMinNumberOfTestedCells

Description

Minimum number of cells tested for one cycle of a background test.

Remarks

This parameter is read-only.

Range

Fixed to 1

4.2.3.3 RamTstNumberOfAlgParamSets

Description

Number of configured parameter sets for the available test algorithms. `calculationFormula` = count of the container `RamTst_AlgParams`.

Remarks

This parameter is read-only.

Range

Fixed to 255

4.2.3.4 RamTstTestCompletedNotification

Description

This function will be called from a background test after finishing the RAM Test without an error.

Remarks

If the notification function is not filled in config, the notification function to the user is turned OFF.

Range

Correct function name. (For example, `TestCompleted`)

4.2.3.5 RamTstTestErrorNotification

Description

This function will be called from a background test if an error occurs during the RAM Test.

Remarks

If the notification function is not filled in config, the notification function to the user is turned OFF.

Range

Correct function name. (For example, `TestError`)

4.2.3.6 RamTstStackBackUpAreaStartAddr

Description

This is the backup area start address of the stack.

Remarks

Start address must be aligned to multiples of four.

Range

The target is SRAM. However, the range varies depending on the type.

4.2.3.7 RamTstStackBackUpAreaSize

Description

This is the backup area size of the stack.

Remarks

This parameter is read-only.

Range

Fixed to 600 bytes.

4.2.3.8 RamTstStaticAreaStartAddr

Description

This is the start address of the static area used by RAM Test.

Remarks

Start address must be aligned to multiples of four.

Range

The target is SRAM. However, the range varies depending on the type.

4.2.3.9 RamTstStaticAreaSize

Description

This is the size of the static area used by RAM Test.

Remarks

This parameter is read-only.

Range

Fixed to 128 bytes.

4.2.4 RamTstDemEventParameterRefs

This container specifies values for each supported DEM error.

Description

There are 10 production errors supported by the RAM Test.

- `RAMTST_E_RAM_FAILURE`
- `RAMTST_MAIN_RAM_FAILURE`
- `RAMTST_RUNFL_RAM_FAILURE`
- `RAMTST_PART_RAM_FAILURE`
- `RAMTST_E_DATA_BACKUP_FAILURE`
- `RAMTST_E_STACK_BACKUP_FAILURE`
- `RAMTST_E_STATIC_AREA_FAILURE`
- `RAMTST_E_NMI_OCCURED`
- `RAMTST_E_SYSTEM_FAILURE`
- `RAMTST_E_SP_FAILURE`

Remarks

See section **7.1.2** for error details.

4.2.5 RamTstPublishedInformation

Container holding all `RamTst` specific published information parameter.

4.2.5.1 RamTstCellSize

Description

Size of the RAM cells (in bits) which can be tested individually by the given implementation.

Remarks

This parameter is read-only.

Range

Fixed to 32.

4.2.6 RamTstAlgParams

This container holds parameters for configuring an algorithm. For each algorithm selected in the `RamTstAlgorithms` container. There can be one or more `RamTstAlgParams` containers. The multiplicity of the included container `RamTstBlockParams` depends on the number of separate blocks of RAM, which are defined for the particular test configuration.

4.2.6.1 RamTstAlgParamsId

Description

Identifies the container that includes the algorithm and target area used for `RamTst`.

Remarks

None

Range

1…255

4.2.6.2 RamTstAlgorithm

Description

This is the algorithm for which the `RamTstAlgParams` set is defined.

Remarks

- The same algorithm can be used in more than one `RamTstAlgParams`.
- Only the algorithms selected by `RamTstCommon/RamTstAlgorithms` can be used.

Range

The algorithm can take the following values:

- `RAMTST_UNIQUE_TEST`
- `RAMTST_MARCH_TEST`
- `RAMTST_CHECKERBOARD_TEST`
- `RAMTST_WALK_PATH_TEST`
- `RAMTST_GALPAT_TEST`
- `RAMTST_TRANSP_GALPAT_TEST`
- `RAMTST_ABRAHAM_TEST`

4.2.6.3 RamTstExtNumberOfTestedCells

Description

This is the absolute maximum value for the number of cells that `RamTstNumberOfTestedCells` and `RamTstMaxNumberOfTestedCells` can be.

Remarks

This parameter is read-only.

Range

Fixed to 4294967295.

4.2.6.4 RamTstMaxNumberOfTestedCells

Description

This is the maximum value for the number of cells that can be tested in one cycle of a background test.

Remarks

None

Range

1…4294967295

4.2.6.5 RamTstNumberOfBlocks

Description

Number of RAM blocks configured using the container `RamTst_BlockParams` calculationFormula = Count of `RamTstBlockParams` contained in `RamTstAlgParams`.

Remarks

Set the same value as the number of blocks set in the `RamTst_BlockParams` container.

Range

1…65535

4.2.6.6 RamTstNumberOfTestedCells

Description

This is the initial value for a RAM variable, which can be changed by the `RamTst_ChangeNumberOfTestedCells`

Remarks

`RamTstNumberOfTestedCells` must be multiples of `RamTstNumberOfTestedCellsAtomic`.

Range

1…4294967295

4.2.6.7 RamTstNumberOfTestedCellsAtomic

Description

The smallest number of cells to be tested in each atomic test.

Remarks

For `Galpat`/`Transparent Galpat`/`Walk Path`, `RamTstNumberOfTestedCellsAtomic` must be a multiple of 4.

Range

1…16384

4.2.6.8 RamTstDataBackUpAreaStartAddr

Description

Start address of the save area of the algorithm parameter set.

Remarks

Start address must be aligned to multiples of four.

Range

The target is SRAM. However, the range varies depending on the type.

4.2.6.9 RamTstDataBackUpAreaSize

Description

Size of the save area of algorithm parameter set.

Remarks

This size must be aligned to multiples of four, and equal to or greater than `RamTstNumberOfTestedCellsAtomic`.

Range

The target is SRAM. However, the range varies depending on the type.

4.2.7 RamTstBlockParams

This container holds the description for one block of RAM. For each RAM block to be tested by a given algorithm, there is one container which describes the block. Multiple instances of this container are included in each `RamTstAlgParams` container.

4.2.7.1 RamTstBlockId

Description

ID of the RAM block.

Remarks

None

Range

1…65535

4.2.7.2 RamTstEndAddress

Description

End address of the RAM block.

Remarks

It must be larger than `RamTstStartAddress`.
The size of the diagnostic area should be a multiple of `RamTstNumberOfTestedCells` and a multiple of 4.

Range

The target is SRAM. However, the range varies depending on the type.

4.2.7.3 RamTstFillPattern

Description

Pattern to be filled into each memory cell after a destructive test of this block.

Remarks

None

Range

0x0…0xFFFFFFFF

4.2.7.4 RamTstStartAddress

Description

Start address of the RAM block.

Remarks

It must be smaller than `RamTstEndAddress`.
The size of the diagnostic area should be a multiple of `RamTstNumberOfTestedCells` and a multiple of 4.
For `Unique`/`March`/`Checkerboard`/`Abraham`, `RamTstStartAddress` must be 4-byte aligned.
For `Galpat`/`Transparent Galpat`/`Walk Path`, `RamTstStartAddress` must be 8-byte aligned.

Range

The target is SRAM. However, the range varies depending on the type.

4.2.7.5 RamTstTestPolicy

Description

Policy regarding destruction or non-destruction of memory content.

Remarks

None

Range

`RAMTEST_NON_DESTRUCTIVE` or

`RAMTEST_DESTRUCTIVE`

5 Functional description

5.1 Initialization

The following are the features of the initialization function (`RamTst_Init()`) of RAM Test:

- The function execution state is not equal to `RAMTST_EXECUTION_UNINIT`, to notify the error value `RAMTST_E_STATUS_FAILURE` to the user.
- The function initializes all data structures, global variables, and registers with the default values.
- The function changes the execution state to `RAMTST_EXECUTION_STOPPED`.
- The function clears the test results of the foreground and background.

5.2 Deinitialization

RAM Test provides the following functions as a diagnostic end processing:

- The function execution state is equal to `RAMTST_EXECUTION_UNINIT`, to notify the error value `RAMTST_E_UNINIT` to the user.
- The function initializes all data structures, global variables, and registers with the default values.
- The function changes the execution state to `RAMTST_EXECUTION_UNINIT`.
- The function clears the test results of the foreground and background.

5.3 Background test

The background test of RAMTST is executed by calling the `RamTst_MainFunction` from BSW.

The `RamTst_MainFunction` provides the following functions:

- The `RamTst_MainFunction` diagnosis is based on RAM domain and algorithm.
- If an error is detected, it notifies the user, and then continues with the diagnostic process.

5.4 Foreground test

The foreground test of RAMTST is executed by calling `RamTst_RunFullTest` or `RamTst_RunPartialTest`. The Foreground test provides the following functions:

- The foreground test is diagnosed based on the RAM domain and algorithm.
- If an error is detected, it notifies the user, and then discontinues the diagnosis process.

5.5 Stop/Suspend

`RamTst_Stop` stops background test. `RamTst_Suspend` suspends background test. If the background test is suspended, resume the test using `RamTst_Resume`. See **Figure 3** for the state transition.

5.6 Allow

`RamTst_Allow` changes the available status to run background test. See **Figure 3** for the state transitions. The following are the behavior of `RamTst_Allow`:

- The function initializes all data structures, global variables, and registers with the default values.
- The function changes the execution state to `RAMTST_EXECUTION_RUNNING`.
- The function clears the test results of the foreground and background.

5.7 State transition

The state of RAMTST and the functions related to the transition are as follows. You can see the state by `RamTst_GetExecutionStatus`.

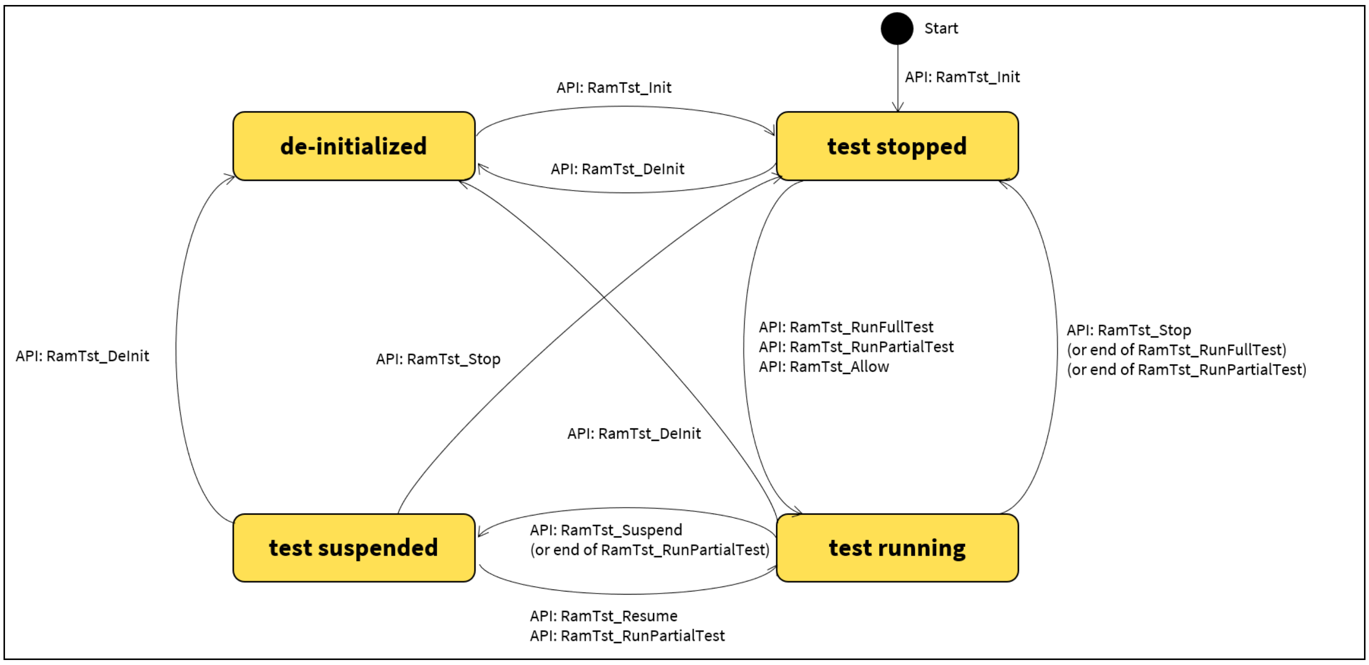

Figure 3 The state chart

5.8 API parameter checking

If default error detection is enabled, the driver's services execute default error checks. A default error is an error which will be detected and fixed during the development phase. A default error will not occur in production code.

All default errors are reported to the DET, a central error hook function within the AUTOSAR environment.

If an error occurs, the error hook routine will be called and the error code, and the service and the module ID will be passed on as parameters. For more information about the DET and its API see *specification of default error tracer* [5].

5.9 Diagnostic algorithm

The following algorithms are by `RamTst_RunFullTest`/`RamTst_RunPartialTest` / `RamTst_MainFunction` to execute detection of fault:

- Checkerboard
- March
- Unique
- Galpat/Transparent galpat
- Walk path
- Abraham

5.10 Production error detection

If failure occurs, `RAMTST_E_RAM_FAILURE` / `RAMTST_MAIN_RAM_FAILURE` / `RAMTST_RUNFL_RAM_FAILURE` / `RAMTST_PART_RAM_FAILURE` / `RAMTST_E_DATA_BACKUP_FAILURE` / `RAMTST_E_STACK_BACKUP_FAILURE` / `RAMTST_E_STATIC_AREA_FAILURE` / `RAMTST_E_NMI_OCCURED` / `RAMTST_E_SYSTEM_FAILURE` / `RAMTST_E_SP_FAILURE` is reported to the DEM.

When an error occurs, the error hook routine (configured via `RamTstErrorCalloutFunction`) is also called and the service ID, module ID, and instance ID are passed as parameters.

5.11 Reentrancy

The following functions are reentrant to each other and itself:

- `RamTst_GetExecutionStatus`
- `RamTst_GetTestResult`
- `RamTst_GetTestResultPerBlock`
- `RamTst_GetVersionInfo`

All other API functions of Ram Test are not reentrant.

5.12 Debugging support

If `RamTstDebuggingSupport` is selected in the configuration, it can cause a pseudo error for debug when testing the specified address.

The address causing the pseudo error is set in the `RamTst_DebuggingAddress` variable included in *RamTst_Cfg.h* header file. The initial value is 0xFFFFFFFF.

6 Hardware resources

6.1 RAM

RAM Test supports system ram (SRAM) of the TRAVEO™ T2G series.

6.2 TCM

RAM Test supports tightly-coupled memories (TCM) of the TRAVEO™ T2G series (Arm® Cortex®-M7 CPU).

6.3 Timer

RAM Test does not use any hardware timers.

6.4 Interrupts

RAM Test does not use any interrupt.

6.5 Data cache

RAM Test performs a clean and invalidate for data cache of the test target TRAVEO™ T2G series (Arm® Cortex®-M7 CPU) SRAM.

7 Appendix

7.1 API reference

7.1.1 Data types

7.1.1.1 RamTst_ExecutionStatusType

Type

```
typedef enum
{
   RAMTST_EXECUTION_UNINIT=0,
   RAMTST_EXECUTION_STOPPED_READY,
   RAMTST_EXECUTION_STOPPED,
   RAMTST_EXECUTION_SUSPENDED_READY,
   RAMTST_EXECUTION_SUSPENDED,
   RAMTST_EXECUTION_RUNNING
} RamTst_ExecutionStatusType
```

emphasis

This is a status value returned by `RamTst_GetExecutionStatus`.

7.1.1.2 RamTst_TestResultType

Type

```
typedef enum
{
   RAMTST_RESULT_NOT_TESTED=0,
   RAMTST_RESULT_OK,
   RAMTST_RESULT_NOT_OK,
   RAMTST_RESULT_UNDEFINED
} RamTst_TestResultType
```

emphasis

This is a status value returned by `RamTst_GetTestResult`/ `RamTst_GetTestResultPerBlock`.

7.1.1.3 RamTst_AlgParamsIdType

Type

```
uint8
```

emphasis

Data type used to identify a set of configuration parameters for a test algorithm.

7.1.1.4 RamTst_AlgorithmType

Type

```
typedef enum
{
   RAMTST_ALGORITHM_UNDEFINED=0,
   RAMTST_WALK_PATH_TEST,
   RAMTST_GALPAT_TEST,
   RAMTST_TRANSP_GALPAT_TEST,
   RAMTST_CHECKERBOARD_TEST,
   RAMTST_MARCH_TEST,
   RAMTST_UNIQUE_TEST,
   RAMTST_ABRAHAM_TEST
} RamTst_AlgorithmType
```

emphasis

This is a value returned by `RamTst_GetTestAlgorithm`.

7.1.1.5 RamTst_NumberOfTestedCellsType

Type

```
uint32
```

emphasis

Data type of the number of tested RAM cells.

7.1.1.6 RamTst_NumberOfBlocksType

Type

```
uint16
```

emphasis

Data type used to identify or count RAM blocks given in the test configuration parameters.

7.1.1.7 RamTst_NMIAccessSizeType

Type

```
typedef enum
{
   SIZE_1_BYTE = 1,
   SIZE_2_BYTE = 2,
   SIZE_4_BYTE = 4,
   SIZE_8_BYTE = 8
}RamTst_NMIAccessSizeType
```

emphasis

It is an enumerated type that represents the size of data to be accessed in the NMI handler.

7.1.1.8 RamTst_NMIStopReturnType

Type

```
typedef enum
{
   RAMTST_NMI_VARIABLE_ACCESS_PERMIT,
   RAMTST_NMI_VARIABLE_ACCESS_FORBID
}RamTst_NMIStopReturnType
```

emphasis

It is an enumerated type that represents the data access permission is the return value of `RamTst_NMIStop`.

7.1.2 Constants

7.1.2.1 Error codes

The service might return the error codes, listed in **Table 2**, if default error detection is enabled:

Table 2 Error codes

Name	Value	Description
`RAMTST_E_STATUS_FAILURE`	0x01	API is called under unexpected state.
`RAMTST_E_OUT_OF_RANGE`	0x02	API is called with the values of the arguments of illegal range.
`RAMTST_E_UNINIT`	0x03	API is called without any initialization of RamTest.
`RAMTST_E_PARAM_POINTER`	0x04	The `RamTst_GetVersionInfo` is called with a NULL pointer as an argument.
`RAMTST_CALLOUT_E_RAM_FAILURE`	0x31	Fault is detected in the diagnostic target area. This error id is used to call error callout handler.
`RAMTST_CALLOUT_DATA_BACKUP_FAILURE`	0x32	Fault is detected in RAM save area. This error id is used to call error callout handler.
`RAMTST_CALLOUT_STACK_BACKUP_FAILURE`	0x33	Fault is detected in the stack save area. This error id is used to call error callout handler.
`RAMTST_CALLOUT_STATIC_AREA_FAILURE`	0x34	Fault is detected in the static area. This error id is used to call error callout handler.
`RAMTST_CALLOUT_NMI_OCCURED`	0x35	The diagnosis canceled due to diagnostic running NMI occurs. This error id is used to call error callout handler.
`RAMTST_CALLOUT_SYSTEM_FAILURE`	0x36	Any system abnormalities are detected. This error id is used to call error callout handler.
`RAMTST_CALLOUT_SP_FAILURE`	0x37	It detects a mismatch of SP. This error id is used to call error callout handler.
`RAMTST_CALLOUT_MAIN_RAM_FAILURE`	0x38	Fault is detected in the diagnostic target area. This error id is used to call error callout handler.

Name	Value	Description
		(Diagnosis with the `RamTst_MainFunction` function)
`RAMTST_CALLOUT_RUNFL_RAM_FAILURE`	0x39	Fault is detected in the diagnostic target area. This error id is used to call error callout handler. (Diagnosis with the `RamTst_FullTest` function)
`RAMTST_CALLOUT_PART_RAM_FAILURE`	0x40	Fault is detected in the diagnostic target area. This error id is used to call error callout handler. (Diagnosis with the `RamTst_PartialTest` function)

Table 3 **Error codes**

Name	Value	Description
`RAMTST_E_RAM_FAILURE`	External allocation by DEM	Fault is detected in the diagnostic target area. It becomes effective in one of the following cases: · When `RAMTST_MAIN_RAM_FAILURE` is invalid · When `RAMTST_RUNFL_RAM_FAILURE` is invalid · When `RAMTST_PART_RAM_FAILURE` is invalid
`RAMTST_MAIN_RAM_FAILURE`	External allocation by DEM	Fault is detected in the diagnostic target area. (Diagnosis with the `RamTst_MainFunction` function)
`RAMTST_RUNFL_RAM_FAILURE`	External allocation by DEM	Fault is detected in the diagnostic target area. (Diagnosis with the `RamTst_FullTest` function)
`RAMTST_PART_RAM_FAILURE`	External allocation by DEM	Fault is detected in the diagnostic target area. (Diagnosis with the `RamTst_PartialTest` function)
`RAMTST_E_DATA_BACKUP_FAILURE`	External allocation by DEM	Fault is detected in RAM save area.
`RAMTST_E_STACK_BACKUP_FAILURE`	External allocation by DEM	Fault is detected in the stack save area.
`RAMTST_E_STATIC_AREA_FAILURE`	External allocation by DEM	Fault is detected in the static area.

Name	Value	Description
`RAMTST_E_NMI_OCCURED`	External allocation by DEM	The diagnosis canceled due to diagnostic running NMI occurs.
`RAMTST_E_SYSTEM_FAILURE`	External allocation by DEM	Failed to enter or exit from critical section.
`RAMTST_E_SP_FAILURE`	External allocation by DEM	SP register has changed during atomic sequence test.

7.1.2.2 Version information

The version information listed in **Table 4** is published in the driver's header file.

Table 4 Version information

Name	Value	Description
`RAMTST_SW_MAJOR_VERSION`	See release notes	Major version number
`RAMTST_SW_MINOR_VERSION`	See release notes	Minor version number
`RAMTST_SW_PATCH_VERSION`	See release notes	Patch version number

7.1.2.3 Module information

Table 5 Module information

Name	Value	Description
`RAMTST_MODULE_ID`	93	Module ID
`RAMTST_VENDOR_ID`	66	Vendor ID

7.1.2.4 API service IDs

The API service IDs listed in **Table 6** are published in the driver's header file.

Table 6 API service IDs

Name	Value	Description
`RamTst_Init()`	0x00	Service ID of `RamTst_Init`
`RamTst_MainFunction()`	0x01	Service ID of `RamTst_MainFunction`
`RamTst_Stop()`	0x02	Service ID of `RamTst_Stop`
`RamTst_Allow()`	0x03	Service ID of `RamTst_Allow`
`RamTst_GetExecutionStatus()`	0x04	Service ID of `RamTst_GetExecutionStatus`
`RamTst_GetTestResult ()`	0x05	Service ID of `RamTst_GetTestResult`
`RamTst_GetTestResultPerBlock()`	0x06	Service ID of `RamTst_GetTestResultPerBlock`
`RamTst_GetTestAlgorithm()`	0x07	Service ID of `RamTst_GetTestAlgorithm`
`RamTst_ChangeNumberOfTestedCells()`	0x08	Service ID of `RamTst_ChangeNumberOfTestedCells`

Name	Value	Description
`RamTst_GetNumberOfTestedCells()`	0x09	Service ID of `RamTst_GetNumberOfTestedCells`
`RamTst_GetVersionInfo()`	0x0a	Service ID of `RamTst_GetVersionInfo`
`RamTst_SelectAlgParams ()`	0x0b	Service ID of `RamTst_SelectAlgParams`
`RamTst_DeInit()`	0x0c	Service ID of `RamTst_DeInit`
`RamTst_Suspend()`	0x0d	Service ID of `RamTst_Suspend`
`RamTst_Resume()`	0x0e	Service ID of `RamTst_Resume`
`RamTst_RunFullTest()`	0x10	Service ID of `RamTst_RunFullTest`
`RamTst_RunPartialTest()`	0x11	Service ID of `RamTst_RunPartialTest`
`RamTst_GetAlgParams()`	0x12	Service ID of `RamTst_GetAlgParams`
`RamTst_NMIStop()`	0x13	Service ID of `RamTst_NMIStop`
`RamTst_ReadGlobalVarInNMI()`	0x14	Service ID of `RamTst_ReadGlobalVarInNMI`
`RamTst_WriteGlobalVarInNMI()`	0x15	Service ID of `RamTst_WriteGlobalVarInNMI`

7.1.3 Variables

7.1.3.1 RamTst_DebuggingAddress

Type

`uint32`

Description

RAM Test can cause a pseudo error at the address of this variable. The initial value is 0xffffffff; so, the RAM Test cannot cause a pseudo error as this address cannot be tested. If a value of test area or backup area. is set to this variable, RAM Test will cause an error at the address. This variable can be used only when `RamTstDebuggingSupport` in the configuration is selected.

7.1.4 Functions

7.1.4.1 RamTst_Init

Syntax

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

Service ID

0x00

Sync/Async

Synchronous

Reentrancy

Non-reentrant

Parameters (in)

`ConfigPtr`: Do not use it internally. Therefore, set to `NULL_PTR`.

Parameters (inout)

None

Parameters (out)

None

Return value

None

DET errors

- `RAMTST_E_STATUS_FAILURE` – The function is called in an unexpected state.

DEM errors

- `RAMTST_E_SYSTEM_FAILURE` – Failed to enter or exit from a critical section.

Description

Service for RAM Test initialization.

7.1.4.2 RamTst_MainFunction

Syntax

```
void RamTst_MainFunction
(
 void
)
```

Service ID

0x01

Timing

`VARIABLE_CYCLIC_OR_ON_PRECONDITION` (see *Specification of RAM Test* [2])

Parameters (in)

None

Parameters (inout)

None

Parameters (out)

None

Return value

None

DET errors

- `RAMTST_E_UNINIT` – RAM Test has not been initialized.

DEM errors

- `RAMTST_E_RAM_FAILURE` – Fault is detected in the diagnostic target area.
- `RAMTST_MAIN_RAM_FAILURE` – Fault is detected in the diagnostic target area.
- `RAMTST_E_DATA_BACKUP_FAILURE` – Fault is detected in RAM save area.
- `RAMTST_E_STACK_BACKUP_FAILURE` – Fault is detected in the stack save area.
- `RAMTST_E_STATIC_AREA_FAILURE` – Fault is detected in the static area.
- `RAMTST_E_NMI_OCCURED` – The diagnosis is canceled if NMI occurs during diagnosis.
- `RAMTST_E_SYSTEM_FAILURE` – Failed to enter or exit from a critical section.
- `RAMTST_E_SP_FAILURE` – SP register changed during atomic sequence test.

Description

Scheduled function for executing RAM Test in the background mode.

If `RAMTST_PART_RAM_FAILURE` is invalid, `RAMTST_E_RAM_FAILURE` is enabled.

7.1.4.3 RamTst_Stop

Syntax

```
void RamTst_Stop
(
 void
)
```

Service ID

0x02

Sync/Async

Asynchronous

Reentrancy

Non-reentrant

Parameters (in)

None

Parameters (inout)

None

Parameters (out)

None

Return value

None

DET errors

- `RAMTST_E_STATUS_FAILURE` – The function is called in an unexpected state.
- `RAMTST_E_UNINIT` – RAM Test has not been initialized.

DEM errors

- `RAMTST_E_SYSTEM_FAILURE` – Failed to enter or exit from a critical section.

Description

Service to stop a background mode test that is in progress.

7.1.4.4 RamTst_Allow

Syntax

```
void RamTst_Allow
(
 void
)
```

Service ID

0x03

Sync/Async

Asynchronous

Reentrancy

Non-reentrant

Parameters (in)

None

Parameters (inout)

None

Parameters (out)

None

Return value

None

DET errors

- `RAMTST_E_STATUS_FAILURE` – The function is called in an unexpected state.
- `RAMTST_E_UNINIT` – RAM Test has not been initialized.

DEM errors

- `RAMTST_E_SYSTEM_FAILURE` – Failed to enter or exit from a critical section.

Description

Service for starting a background mode test.

7.1.4.5 RamTst_GetExecutionStatus

Syntax

```
RamTst_ExecutionStatusType RamTst_GetExecutionStatus
(
 void
)
```

Service ID

0x04

Sync/Async

Synchronous

Reentrancy

Reentrant

Parameters (in)

None

Parameters (inout)

None

Parameters (out)

None

Return value

- `RAMTST_EXECUTION_UNINIT` – Uninitialized.
- `RAMTST_EXECUTION_STOPPED` – The diagnosis is pending and is waiting to be restarted
- `RAMTST_EXECUTION_STOPPED_READY` – The diagnosis stopped.
- `RAMTST_EXECUTION_RUNNING` – The diagnosis is running.
- `RAMTST_EXECUTION_SUSPENDED` – The diagnosis is suspended and is waiting to resume.
- `RAMTST_EXECUTION_SUSPENNDED_READY` – The diagnosis is suspended.

DET errors

- `RAMTST_E_UNINIT` – RAM Test has not been initialized.

DEM errors

None

Description

Service returns the current RAM Test execution status.

7.1.4.6 RamTst_GetTestResult

Syntax

```
RamTst_TestResultType RamTst_GetTestResult
(
 void
)
```

Service ID

0x05

Sync/Async

Synchronous

Reentrancy

Reentrant

Parameters (in)

None

Parameters (inout)

None

Parameters (out)

None

Return value

- `RAMTST_RESULT_NOT_TESTED` – Diagnosis is not yet executed.
- `RAMTST_RESULT_OK` – Diagnosis result is OK.
- `RAMTST_RESULT_NOT_OK` – Diagnosis result is not OK.
- `RAMTST_RESULT_UNDEFINED` – The result is not fixed since the diagnosis is not finished yet.

DET errors

- `RAMTST_E_UNINIT` – RAM Test has not been initialized.

DEM errors

- `RAMTST_E_SYSTEM_FAILURE` – Failed to enter or exit a critical section.

Description

Service returns the current RAM Test result.

7.1.4.7 RamTst_GetTestResultPerBlock

Syntax

```
RamTst_TestResultType RamTst_GetTestResultPerBlock
(
 RamTst_NumberOfBlocksType BlockId
)
```

Service ID

0x06

Sync/Async

Synchronous

Reentrancy

Reentrant

Parameters (in)

`BlockId`: ID of the RAM blocks defined in the `RamTstAlgParams`.

Parameters (inout)

None

Parameters (out)

None

Return value

- `RAMTST_RESULT_NOT_TESTED` – Diagnosis is not yet executed.
- `RAMTST_RESULT_OK` – Diagnosis result is OK.
- `RAMTST_RESULT_NOT_OK` – Diagnosis result is not OK.
- `RAMTST_RESULT_UNDEFINED` – The result is not fixed since the diagnosis is not finished yet.

DET errors

- `RAMTST_E_OUT_OF_RANGE` – The argument of the function is out of range.
- `RAMTST_E_UNINIT` – RAM Test has not been initialized.

DEM errors

- `RAMTST_E_SYSTEM_FAILURE` – Failed to enter or exit a critical section.

Description

Service returns the current RAM Test result for the specified block.

7.1.4.8 RamTst_GetTestAlgorithm

Syntax

```
RamTst_AlgorithmType RamTst_GetTestAlgorithm
(
void
)
```

Service ID

0x07

Sync/Async

Synchronous

Reentrancy

Non-reentrant

Parameters (in)

None

Parameters (inout)

None

Parameters (out)

None

Return value

- `RAMTST_ALGORITHM_UNDEFINED` – Test algorithm is undefined.
- `RAMTST_CHECKERBOARD_TEST` – Checkerboard test algorithm.
- `RAMTST_MARCH_TEST` – March test algorithm.
- `RAMTST_WALK_PATH_TEST` – Walking path test algorithm.
- `RAMTST_GALPAT_TEST` – Galpat test algorithm.
- `RAMTST_TRANSP_GALPAT_TEST` – Transparent galpat test algorithm.
- `RAMTST_ABRAHAM _TEST` – Abraham test algorithm.
- `RAMTST_UNIQUE_TEST` – Unique test algorithm.

DET errors

- `RAMTST_E_UNINIT` – RAM Test has not been initialized.

DEM errors

None.

Description

Service returns the current RAM Test algorithm.

7.1.4.9 RamTst_ChangeNumberOfTestedCells

Syntax

```
void RamTst_ChangeNumberOfTestedCells
(
 RamTst_NumberOfTestedCellsType NewNumberOfTestedCells
)
```

Service ID

0x08

Sync/Async

Synchronous

Reentrancy

Non-reentrant

Parameters (in)

`NewNumberOfTestedCells`: The number of cells to be tested per cycle. The value of `RamTstConfigParams/ RamTstMinNumberOfTestedCells` to `RamTstAlgParams/ RamTstMaxNumberOfTestedCells`

Parameters (inout)

None

Parameters (out)

None

Return value

None

DET errors

- `RAMTST_E_STATUS_FAILURE` – The function is called in an unexpected state.
- `RAMTST_E_OUT_OF_RANGE` – The argument of the function is not a multiple of `RamTstNumberOfTestedCellsAtomic` or out of range.
- `RAMTST_E_UNINIT` – RAM Test has not been initialized.

DEM errors

- `RAMTST_E_SYSTEM_FAILURE` – Failed to enter or exit a critical section.

Description

Service changes the current number of tested cells.

7.1.4.10 RamTst_GetNumberOfTestedCells

Syntax

```
RamTst_NumberOfTestedCellsType RamTst_GetNumberOfTestedCells
(
void
)
```

Service ID

0x09

Sync/Async

Synchronous

Reentrancy

Non-reentrant

Parameters (in)

None

Parameters (inout)

None

Parameters (out)

None

Return value

- `RamTst_NumberOfTestedCellsType` – Number of cells per cycle currently being tested.

DET errors

- `RAMTST_E_UNINIT` – RAM Test has not been initialized.

DEM errors

None.

Description

Service returns the current number of tested cells per main-function cycle.

7.1.4.11 RamTst_GetVersionInfo

Syntax

```
void RamTst_GetVersionInfo
(
 Std_VersionInfoType* versioninfo
)
```

Service ID

0x0a

Sync/Async

Synchronous

Reentrancy

Reentrant

Parameters (in)

None.

Parameters (inout)

None.

Parameters (out)

`versioninfo`: Pointer to the location or address where the version information of this module is stored.

- Vendor ID – Number assigned to the vendor.
- Module ID – Number assigned to the module.
- Vendor specific version numbers – Module major version, module minor version, module patch version.

Return value

None.

DET errors

- `RAMTST_E_PARAM_POINTER` – The function is called with a NULL pointer.

DEM errors

None.

Description

Service returns the version information of this module.

7.1.4.12 RamTst_SelectAlgParams

Syntax

```
void RamTst_SelectAlgParams
(
 RamTst_AlgParamsIdType NewAlgParamsId
)
```

Service ID

0x0b

Sync/Async

Synchronous

Reentrancy

Non-reentrant

Parameters (in)

`NewAlgParamsId`: One of the configured values for `RamTstAlgParams/RamTstAlgParamsId`. The value range is 1-255.

Parameters (inout)

None

Parameters (out)

None

Return value

None

DET errors

- `RAMTST_E_STATUS_FAILURE` – The function is called in an unexpected state.
- `RAMTST_E_OUT_OF_RANGE` – The argument of the function is out of range.
- `RAMTST_E_UNINIT` – RAM Test has not been initialized.

DEM errors

- `RAMTST_E_SYSTEM_FAILURE` – Failed to enter or exit a critical section.

Description

Service used to set the test algorithm and its parameter set.

7.1.4.13 RamTst_DeInit

Syntax

```
void RamTst_DeInit
(
 void
)
```

Service ID

0x0c

Sync/Async

Synchronous

Reentrancy

Non-reentrant

Parameters (in)

None

Parameters (inout)

None

Parameters (out)

None

Return value

None

DET errors

- `RAMTST_E_UNINIT` – RAM Test has not been initialized.

DEM errors

- `RAMTST_E_SYSTEM_FAILURE` – Failed to enter or exit a critical section.

Description

Service for RAM Test deinitialization.

7.1.4.14 RamTst_Suspend

Syntax

```
void RamTst_Suspend
(
 void
)
```

Service ID

0x0d

Sync/Async

Asynchronous

Reentrancy

Non-reentrant

Parameters (in)

None

Parameters (inout)

None

Parameters (out)

None

Return value

None

DET errors

- `RAMTST_E_STATUS_FAILURE` – The function is called in an unexpected state.
- `RAMTST_E_UNINIT` – RAM Test has not been initialized.

DEM errors

- `RAMTST_E_SYSTEM_FAILURE` – Failed to enter or exit a critical section.

Description

Service for suspending current operation of background RAM Test, until `RamTst_Resume` is called.

7.1.4.15 RamTst_Resume

Syntax

```
void RamTst_Resume
(
 void
)
```

Service ID

0x0e

Sync/Async

Asynchronous

Reentrancy

Non-reentrant

Parameters (in)

None

Parameters (inout)

None

Parameters (out)

None

Return value

None

DET errors

- `RAMTST_E_STATUS_FAILURE` – The function is called in an unexpected state.
- `RAMTST_E_UNINIT` – RAM Test has not been initialized.

DEM errors

- `RAMTST_E_SYSTEM_FAILURE` – Failed to enter or exit a critical section.

Description

Service that allows the background RAM Test at the point to continue is suspended.

7.1.4.16 RamTst_RunFullTest

Syntax

```
void RamTst_RunFullTest
(
void
)
```

Service ID

0x10

Sync/Async

Synchronous

Reentrancy

Non-reentrant

Parameters (in)

None

Parameters (inout)

None

Parameters (out)

None

Return value

None

DET errors

- `RAMTST_E_STATUS_FAILURE` – The function is called in an unexpected state.
- `RAMTST_E_UNINIT` – RAM Test has not been initialized.

DEM errors

- `RAMTST_E_RAM_FAILURE` – Fault is detected in the diagnostic target area.
- `RAMTST_RUNFL_RAM_FAILURE` – Fault is detected in the diagnostic target area.
- `RAMTST_E_DATA_BACKUP_FAILURE` – Fault is detected in RAM save area.
- `RAMTST_E_STACK_BACKUP_FAILURE` – Fault is detected in the stack save area.
- `RAMTST_E_STATIC_AREA_FAILURE` – Fault is detected in the static area.
- `RAMTST_E_NMI_OCCURED` – The diagnosis is canceled if NMI occurs during diagnosis.
- `RAMTST_E_SYSTEM_FAILURE` – Failed to enter or exit a critical section.
- `RAMTST_E_SP_FAILURE` – SP register has changed during atomic sequence test.

Description

Service for executing the full RAM Test in the foreground mode.

If `RAMTST_RUNFL_RAM_FAILURE` is invalid, `RAMTST_E_RAM_FAILURE` is enabled.

7.1.4.17 RamTst_RunPartialTest

Syntax

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

Service ID

0x11

Sync/Async

Synchronous

Reentrancy

Non-reentrant

Parameters (in)

`BlockId`: ID of the RAM blocks defined in the `RamTstAlgParams`.

Parameters (inout)

None

Parameters (out)

None

Return value

None

DET errors

- `RAMTST_E_STATUS_FAILURE` – The function is called in an unexpected state.
- `RAMTST_E_OUT_OF_RANGE` – The argument of the function is out of range.
- `RAMTST_E_UNINIT` – RAM Test has not been initialized.

DEM errors

- `RAMTST_E_RAM_FAILURE` – Fault is detected in the diagnostic target area.
- `RAMTST_PART_RAM_FAILURE` – Fault is detected in the diagnostic target area.
- `RAMTST_E_DATA_BACKUP_FAILURE` – Fault is detected in RAM save area.
- `RAMTST_E_STACK_BACKUP_FAILURE` – Fault is detected in the stack save area.
- `RAMTST_E_STATIC_AREA_FAILURE` – Fault is detected in the static area.
- `RAMTST_E_NMI_OCCURED` – The diagnosis is canceled if NMI occurs during diagnosis.
- `RAMTST_E_SYSTEM_FAILURE` – Failed to enter or exit a critical section.
- `RAMTST_E_SP_FAILURE` – SP register mode has changed during atomic sequence test.

Description

Service for testing one RAM block in the foreground mode.

If `RAMTST_PART_RAM_FAILURE` is invalid, `RAMTST_E_RAM_FAILURE` is enabled.

7.1.4.18 RamTst_GetAlgParams

Syntax

```
RamTst_AlgParamsIdType RamTst_GetAlgParams
(
 void
 )
```

Service ID

0x12

Sync/Async

Synchronous

Reentrancy

Non-reentrant

Parameters (in)

None

Parameters (inout)

None

Parameters (out)

None

Return value

`RamTst_AlgParamsIdType` – ID for the current target container based on the algorithm and target area.

DET errors

- `RAMTST_E_UNINIT` - RAM Test has not been initialized.

DEM errors

None

Description

Service returns the ID of the current RAM Test algorithm parameter set.

7.1.4.19 RamTst_NMIStop

Syntax

```
RamTst_NMIStopReturnType RamTst_NMIStop
(
 void
)
```

Service ID

0x13

Sync/Async

Synchronous

Reentrancy

Non-reentrant

Parameters (in)

None

Parameters (inout)

None

Parameters (out)

None

Return value

`RamTst_NMIStopReturnType` – It shows data accessibility.

DET errors

None

DEM errors

None

Description

If the NMI occurs during diagnosis, call this API at the head of NMI handler, to interrupt the running diagnosis and to know whether accessing global variable is permitted.

7.1.4.20 RamTst_ReadGlobalVarInNMI

Syntax

```
void RamTst_ReadGlobalVarInNMI
(
 const void *ReadAddress, RamTst_NMIAccessSizeType AccessSize, void *ReadDataBuffer
)
```

Service ID

0x14

Sync/Async

Synchronous

Reentrancy

Non-reentrant

Parameters (in)

- `ReadAddress`: Address of the user global variable to be read in the NMI handler.
- `AccessSize`: Size of the user global variable that NMI handler reads out.

Parameters (inout)

None

Parameters (out)

- `ReadDataBuffer`: Buffer for storing data to be read.

Return value

None

DET errors

- `RAMTST_E_PARAM_POINTER` – The function called with a NULL pointer.

DEM errors

None

Description

A NMI can occur during a diagnosis. In the users' NMI handler, if the user reads the user global variable while the variable is being diagnosed, the value read out is not the expected value. In this case, use this service to read the user global variable safely.

7.1.4.21 RamTst_WriteGlobalVarInNMI

Syntax

```
void RamTst_WriteGlobalVarInNMI
(
 void *WriteAddress, RamTst_NMIAccessSizeType AccessSize, const void *WriteDataBuffer
)
```

Service ID

0x15

Sync/Async

Synchronous

Reentrancy

Non-reentrant

Parameters (in)

- `WriteAddress`: Address of the user global variable to be written in the NMI handler.
- `AccessSize`: Size of the user global variable that the NMI handler must write.

Parameters (inout)

None

Parameters (out)

- `WriteDataBuffer`: Buffer for data to write.

Return value

None

DET errors

- `RAMTST_E_PARAM_POINTER` – The function called with a NULL pointer.

DEM errors

None

Description

A NMI can occur during a diagnosis. In the users' NMI handler, if the user writes the user global variable while the variable is being diagnosed, the written value is not valid. In this case, use this service to write the user global variable safely.

7.1.4.22 RamTst_TestCompletedNotification

The name of the function to be called can be configured by the RamTstTestCompletedNotification parameter.

Syntax

```
void RamTst_TestCompletedNotification
(
 void
)
```

Sync/Async

-

Reentrancy

Don't care

Parameters (in)

None

Parameters (inout)

None

Parameters (out)

None

Return value

None

DET errors

None

DEM errors

None

Description

The `RamTst_TestCompletedNotification` function will be called every time all RAM blocks of the current test configuration are being tested in the background.

If the notification function is not filled in config, the notification function to the user is turned OFF.

7.1.4.23 RamTst_ErrorNotification

The name of the function to be called can be configured by parameter RamTstErrorNotification.

Syntax

```
void RamTst_ErrorNotification
(
 void
)
```

Sync/Async

-

Reentrancy

Don’t care

Parameters (in)

None

Parameters (inout)

None

Parameters (out)

None

Return value

None

DET errors

None

DEM errors

None

Description

The `RamTst_ErrorNotification` function will be called every time a RAM failure is detected by the selected test algorithm in the background.

If the notification function is not filled in config, the notification function to the user is turned OFF.

7.1.4.24 Error callout API

The AUTOSAR RamTst module requires an error callout handler. Each error is reported to this handler, error checking cannot be switched OFF. The name of the function to be called can be configured by parameter RamTstErrorCalloutFunction.

Syntax

```
void Error_Handler_Name
(
 uint16 ModuleId, uint8 InstanceId,
 uint8 ApiId, uint8 ErrorId
)
```

Reentrancy

Reentrant

Parameters (in)

- `ModuleId`: Module ID of calling module.
- `InstanceId`: Instance ID of calling module.
- `ApiId`: ID of the API that calls this function.
- `ErrorId`: ID of the detected error.

Parameters (inout)

None

Parameters (out)

None

Return value

None

DET errors

None

DEM errors

None

Description

Service for reporting errors. If default error or product error happens, this service will be called regardless of whether the error report function is switched ON. This service is executed by the user.

8 Appendix

8.1 Access register table

8.1.1 CM7_SCS

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
`CCR` (Configuration and control register)	31:0	Word (32 bits)	– (Read only.)	Previous data cache enabled.	`SRAM Test`	0x00000000 (monitoring is not needed.)	0x00000000 (monitoring is not needed.)
`DCCIMVAC` (D-cache clean and invalidate by MVA to PoC)	31:0	Word (32 bits)	– (The address of RamTst ram area)	Cleans and invalidates D-cache.	`SRAM Test`	0x00000000 (monitoring is not needed.)	0x00000000 (monitoring is not needed.)

Revision history

Revision	Issue date	Description of change
**	2018-01-23	New spec.
*A	2018-06-29	Updated the documentation name and the number in hardware documentation (section:Related documentation). Changed generated file name (from RamTst_BswImplementation.bmd to RamTst_Bswmd.axml) and related description(section:2.6 Memory mapping/ 3.1 Static files/ 3.3 Generated files).
*B	2018-11-30	- Related documentation Updated for Cortex®-M7 device. - 4.2.1 RamTstCommon Added the new configuration. - 6 Hardware resources Added the resources for Arm® Cortex®-M7 CPU. - Appendix B Added the chapter.
*C	2019-06-18	- Related documentation Updated hardware documentation information. - 2.2 Configuration RAM Test Deleted the description of the stack area. - 2.3.1.Setting of LD file Deleted the description of the stack area. Changed configuration name: RamTstStackBackUpAreaSize RamTstDataBackUpAreaSize - 2.5 Measuring stack consumption Deleted this section. - 4.2 Containers and configuration parameters Added the new configuration: RamTstDebuggingSupport Changed configuration names: RamTstStackBackUpAreaSize RamTstStaticAreaSize RamTstDataBackUpAreaSize Deleted the configurations: RamTstStackAreaStartAddr_PSP RamTstStackAreaEndAddr_PSP RamTstStackAreaStartAddr_MSP RamTstStackAreaEndAddr_MSP - 5.12 Debugging support Updated the description. - A.1.3 Variables

Revision history

Revision	Issue date	Description of change
		Added this section.
*D	2019-08-02	- 6.5 Data cache Changed description. - Appendix B Changed access registers.
*E	2019-11-18	- 4.2.6 RamTstAlgParams Updated remarks of RamTstNumberOfTestedCellsAtomic - 4.2.7 RamTstBlockParams Updated remarks of RamTstStartAddress
*F	2020-09-05	- 2.5 Memory mapping Changed a memmap file include folder in section "Memory mapping".
*G	2020-11-20	MOVED TO INFINEON TEMPLATE.
*H	2021-12-23	Updated to Infineon style.
*I	2023-12-08	Web release. No content updates.

Edition 2023-12-08

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email:

erratum@infineon.com

Document reference

002-22288 Rev. *I