Document Title	Requirements on Flash Test
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	260

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R23-11

Document Change History

Date	Release	Changed by	Description
2023-11-23	R23-11	AUTOSAR Release Management	• References to "AUTOSAR SWS RAM Test.pdf" removed from the document
2022-11-24	R22-11	AUTOSAR Release Management	• No content changes
2021-11-25	R21-11	AUTOSAR Release Management	• No content changes
2020-11-30	R20-11	AUTOSAR Release Management	• No content changes
2019-11-28	R19-11	AUTOSAR Release Management	• No content changes • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Editorial changes
2017-12-08	4.3.1	AUTOSAR Release Management	• Removed references to HIS • Renamed "default error" to "development error" • minor corrections / clarifications / editorial changes; for details please refer to the ChangeDocumentation
2016-11-30	4.3.0	AUTOSAR Release Management	• Editorial changes

▽

△

2015-07-31	4.2.2	AUTOSAR Release Management	• Editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	• Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Editorial changes
2013-03-15	4.1.1	AUTOSAR Release Management	• Link Requirement with BSW Feature Document • Updating format of requirements according to TPS StandardizationTemplate
2010-02-02	3.1.4	AUTOSAR Release Management	• Initial release

Contents

1 Scope of Document

This document specifies requirements on the module Flash Test.

2 Conventions to be used

2.1 Document Conventions

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template, chapter Support for Traceability ([1]).

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see Standardization Template, chapter Support for Traceability ([1]).

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as follows.

Note that the requirement level of the document in which they are used modifies the force of these words.

- MUST: This word, or the adjective "LEGALLY REQUIRED", means that the definition is an absolute requirement of the specification due to legal issues.
- MUST NOT: This phrase, or the phrase "MUST NOT", means that the definition is an absolute prohibition of the specification due to legal issues.
- SHALL: This phrase, or the adjective "REQUIRED", means that the definition is an absolute requirement of the specification.
- SHALL NOT: This phrase means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED", means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY: This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item.

An implementation, which does not include a particular option, SHALL be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, SHALL be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides.)

3 Acronyms and abbreviations

The glossary below includes acronyms and abbreviations relevant to Flash Test that are not included in the AUTOSAR Glossary [2].

Acronym:	Description:
ECU	Electric Control Unit
EOL	End Of Line Often used in the term 'EOL Programming' or 'EOL Configuration'
CRC	Cyclic Redundancy Check
MCAL	Microcontroller Abstraction Layer
MCU	Microcontroller Unit
NMI	Non maskable interrupt
OS	Operating System
DEM	Diagnostic Event Manager
SFR	Special Function Register
RTE	Runtime environment
WP	Work Package
ECC	Error Correction Code

Table 3.1: Acronyms used in the scope of this Document

Abbreviation:	Description:
STD	Standard
REQ	Requirement
UNINIT	Uninitialized (= not initialized)

Table 3.2: Abbreviations used in the scope of this Document

Term:	Description:
signature	Unique calculation result of the content of a specific memory block
Memory scrubbing	Automatic sequential data reading to trigger detection/verification mechanisms typical ECC.
Invariable memory	Invariable memory can be program flash, program SRAM, locked cache and ROM
Background test	Background test is called periodically by a scheduler, and is interruptible. The test is split up over many scheduled tasks.
Foreground test	Foreground test is called via users call.
Test interval	Interval of a complete Flash test in background mode

Table 3.3: Terms used in the scope of this Document

As this is a document from professionals for professionals, all other terms are expected to be known.

4 Requirements Specification

This chapter describes all requirements driving the work to define the Flash Test.

4.1 Functional Overview

This SW module provides algorithm to test invariable memory. Invariable memory can be data/program flash, program SRAM, locked cache and is either embedded in the microcontroller or memory mapped connected to the microcontroller. For simplification the SW module is called Flash Test Driver.

The test service can be executed at any time after MCU initialization and it is up to the user of the Flash Test Driver to choose the suitable test algorithm and the right execution place to fulfill the safety requirements of the system. The test service itself is dependant on the storage concept of the system. Therefore the availability of different test algorithms is configurable.

The Flash Test driver is intended to be integrated in the overall safety concept and will not provide the required diagnostic coverage on its own.

4.2 Functional Requirements

The Flash Test module is using the Diagnostic Event Manager (DEM) for error reporting. Errors are reported though the DEM API (BSW, Dem_SetEventStatus()).

Development errors are reported to the Default Error Tracer (DET).

4.2.1 Configuration

In this chapter requirements on configurability of the module are listed.

[SRS_FlsTst_14222] Memory block test shall be configured ⌈

Description:	Configure memory blocks to be tested. Block sizes shall be configurable. It shall be possible to configure multiple Flash block areas (for example by configuring their start and end address). Additionally the link to the stored signature or checksum locations shall be configured for each memory block, if applicable.
Rationale:	Define memory blocks to be tested.
Use Case:	–
Dependencies:	–

▽

△

Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129)*

[SRS_FlsTst_14200] Flash test service shall be configured ⌈

Description:	It shall be configurable which test algorithms can be used on each memory block.
Rationale:	Adapt flash test service to system storage concept and optimize driver code
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129)*

[SRS_FlsTst_14201] Post build configuration shall be supported ⌈

Description:	Post build configuration shall be supported
Rationale:	1. For object code deliveries. 2. Flash test configuration is dependent on addresses and can vary with SW versions
Use Case:	load configuration for different ECU variants
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129)*

4.2.2 Initialization

No special initialization requirements collected.

4.2.3 Normal Operation

In this chapter requirements on the "normal" functionality of the module are listed.

[SRS_FlsTst_14202] Data integrity shall be checked using ECC ⌈

Description:	Actively test data integrity using ECC. An algorithm shall be provided to check the correctness of data and redundant bits extended to a word of memory when reading the word. This algorithm is applicable if invariable memory supports this mechanism. The test shall report a DEM error in case of any failure.
Rationale:	Detect single bit failures, two-bit failures, three-bit failures in a 16-bit word
Use Case:	SW ECC or HW ECC. It can be used for "memory scrubbing".
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_00129)*

[SRS_FlsTst_14203] Data integrity shall be checked using checksum ⌈

Description:	Check data integrity using checksum An algorithm shall be provided to re-calculate a checksum of a memory block and compare it with stored checksum located on a defined address in the invariable memory. This algorithm is applicable if flash memory contains a checksum. The link to the location of the stored checksum shall be configurable. The test shall report a DEM error in case of checksum mismatch.
Rationale:	Detect bit failures
Use Case:	–
Dependencies:	SRS_FlsTst_14222

⌋*(RS_BRF_00129)*

[SRS_FlsTst_14204] Data integrity shall be checked using CRC signature (8-bit length) ⌈

Description:	An algorithm shall be provided to re-calculate a signature of a memory block using CRC algorithm and compare it with a stored signature located on a defined address in the invariable memory. The CRC checksum shall have the length of a 8 bit word. This test is applicable if flash memory contains a block signature. The link to the location of the stored signature shall be configurable. The test shall report a DEM error in case of signature mismatch.
Rationale:	Detect bit failures within a word as well as 99.6% of all possible bit failures
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_00129)*

[SRS_FlsTst_14205] Data integrity shall be checked using CRC signature (16-bit length) ⌈

Description:	An algorithm shall be provided to re-calculate a signature of a memory block using CRC algorithm and compare it with a stored signature located on a defined address in the invariable memory. The CRC checksum shall have the length of a 16 bit word. This algorithm is applicable if flash memory contains a block signature. The link to the location of the stored signature shall be configurable. The test shall report a DEM error in case of signature mismatch.
Rationale:	Detect bit failures within a word as well as 99.998% of all possible bit failures
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_00129)*

[SRS_FlsTst_14206] Data integrity shall be checked using CRC signature (32-bit length) ⌈

Description:	An algorithm shall be provided to re-calculate a signature of a memory block using CRC algorithm and compare it with a stored signature located on a defined address in the invariable memory. The CRC checksum shall have the length of a 32bit word. This algorithm is applicable if flash memory contains a block signature. The link to the location of the stored signature shall be configurable. The test shall report a DEM error in case of signature mismatch.
Rationale:	Detect bit failures within a word as well as 99.99999995% of all possible bit failures.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_00129)*

[SRS_FlsTst_14207] Data integrity shall be checked by comparing duplicated memory blocks ⌈

Description:	A test shall be provided to compare two identical memory blocks. The test case is applicable when memory is duplicated. The test shall report a DEM error in case of a mismatch.
Rationale:	Detect all bit failures according to block replication technique
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_00129)*

[SRS_FlsTst_14208] Background Flash test shall be interruptible ⌈

Description:	–
Rationale:	Test is intended to be used in a scheduled background task. Therefore it should not block other operations.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129)*

[SRS_FlsTst_14209] The memory to be tested shall be split into individual smaller pieces ⌈

Description:	The memory to be tested shall be split into individual smaller pieces, which can be scheduled according to the needs of the system.
Rationale:	Share CPU resources with concurrent tasks
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129)*

[SRS_FlsTst_14211] Flash test execution status shall be available ⌈

Description:	Current status of Flash test execution shall be available through a get status interface. This function shall be optional.
Rationale:	Ability to monitor the current running test for test flow control.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129)*

[SRS_FlsTst_14212] Flash test execution completion shall be provided by a notification mechanism ⌈

Description:	Information about test has been finished shall be provided to the user by a notification mechanism. This function shall be optional.
Rationale:	Ability to indicate the completion of the current running test for test flow control.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129)*

[SRS_FlsTst_14213] Calculation signature/checksum of a finalized test shall be provided ⌈

Description:	Calculation signature/checksum of a finalized test shall be provided to the user. This function shall be optional.
Rationale:	Pass/fail decision of a finalized test can be made outside of this SW module.
Use Case:	Pass or fail decision could be done from an external safety unit
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129)*

[SRS_FlsTst_14214] Service for Flash test execution result shall be provided. ⌈

Description:	Information about the test that has been finished shall be provided upon SW request. This function shall be optional.
Rationale:	Ability to monitor the completion of the finalized test for diagnostic purpose.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129)*

[SRS_FlsTst_14215] Suspend Flash test execution shall be possible ⌈

Description:	A service shall be provided to suspend a running Flash test. Suspend will stop the test execution at the next atomic boundary and store the intermediate state. This function shall be optional.
Rationale:	Suspend background flash test task in case of higher priority tasks.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129)*

[SRS_FlsTst_14216] Flash test execution shall be resumed when suspended ⌈

Description:	A service shall be provided to resume a Flash test which was suspended. This function shall be optional.
Rationale:	Complete a suspended flash test.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129)*

[SRS_FlsTst_14217] Flash test execution shall be stopped when wanted ⌈

Description:	A service shall be provided to stop a Flash test. The test shall be cancelled and a re-start of the test shall start from the beginning.
Rationale:	–
Use Case:	In case of DEM error the user can cancel a running flash test execution
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129)*

[SRS_FlsTst_14219] Foreground Flash test shall be available ⌈

Description:	A service shall be provided to test memory blocks in foreground mode. This function shall be optional.
Rationale:	–
Use Case:	• test complete program flash during startup phase • test special memory block before critical operations
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129)*

[SRS_FlsTst_14223] Flash Test Error details shall be reported ⌈

Description:	A service shall be provided to report specific error data detected from ECC tests. The service shall be optional and is applicable in case of: • hardware is equipped with ECC for invariable memories • hardware is able to report hardware specific error details • caller requires these data It is the responsibility of the caller to interpret the data provided from this service.
Rationale:	These detailed error data are used for diagnostic purpose.
Use Case:	During ECU start-up invariable memory shall be tested for ECC failure. In case of failure, this function shall be used to collect hardware specific fault data like the fault address of an ECC failure
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129, RS_BRF_02168)*

[SRS_FlsTst_14224] ECC Circuitry shall be tested ⌈

Description:	A service shall be provided to test the ECC circuitry and report the test result. The service shall be optional and is applicable in case of: • hardware is equipped with ECC for invariable memories • mechanism is provided from the hardware to test ECC circuitry • caller requires this test
Rationale:	In a safety related application, it may be necessary to establish that hardware memory test mechanism is functioning properly. This can be achieved by verifying that the available circuitry (ECC) is functioning without errors.
Use Case:	Verify ECC hardware logic during ECU start-up.
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129)*

[SRS_FlsTst_14225] Each Flash test Interval shall have an Identifier ⌈

Description:	Each Flash test Interval shall have an Identifier, which shall be incremented by each start of a validtest interval in background mode. The value of the Flash test interval shall be provided to upper layer. The end value of the Identifier shall be configurable.
Rationale:	Assign test result or test signature to a test interval in order to monitor ECU test flow by upper layer.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129, RS_BRF_01024)*

4.2.4 Shutdown Operation

There are no dedicated requirements for shutdown operation collected.

4.2.5 Fault Operation

There are no dedicated requirements for fault/recovery operation operations collected for this SW module.

4.3 Non-Functional Requirements (Qualities)

There are no dedicated non-functional requirements collected.

4.3.1 Timing Requirements

There are no dedicated timing requirements collected.

4.3.2 Resource Usage

[SRS_FlsTst_14221] Memory Content to Be Tested Should Not be Valid During the Test ⌈

Description:	The tests are content based and require no change of the content during the test period. This has to be ensured by the caller.
Rationale:	Content of memory blocks under test shall not be valid during the test period
Use Case:	Test of data flash during run time
Dependencies:	–
Supporting Material:	–

⌋*(RS_BRF_02224, RS_BRF_00129)*

5 Requirements Tracing

The following table references the features specified in [3] and links to the fulfillments of these.

Requirement	Description	Satisfied by
[RS_BRF_00129]	AUTOSAR shall support data corruption detection and protection	[SRS_FlsTst_14200] [SRS_FlsTst_14201] [SRS_FlsTst_14202] [SRS_FlsTst_14203] [SRS_FlsTst_14204] [SRS_FlsTst_14205] [SRS_FlsTst_14206] [SRS_FlsTst_14207] [SRS_FlsTst_14208] [SRS_FlsTst_14209] [SRS_FlsTst_14211] [SRS_FlsTst_14212] [SRS_FlsTst_14213] [SRS_FlsTst_14214] [SRS_FlsTst_14215] [SRS_FlsTst_14216] [SRS_FlsTst_14217] [SRS_FlsTst_14219] [SRS_FlsTst_14221] [SRS_FlsTst_14222] [SRS_FlsTst_14223] [SRS_FlsTst_14224] [SRS_FlsTst_14225]
[RS_BRF_01024]	AUTOSAR shall provide naming rules for public symbols	[SRS_FlsTst_14225]
[RS_BRF_02168]	AUTOSAR diagnostics shall provide a central classification and handling of abnormal operative conditions	[SRS_FlsTst_14223]
[RS_BRF_02224]	AUTOSAR shall support run-time hardware tests	[SRS_FlsTst_14200] [SRS_FlsTst_14201] [SRS_FlsTst_14208] [SRS_FlsTst_14209] [SRS_FlsTst_14211] [SRS_FlsTst_14212] [SRS_FlsTst_14213] [SRS_FlsTst_14214] [SRS_FlsTst_14215] [SRS_FlsTst_14216] [SRS_FlsTst_14217] [SRS_FlsTst_14219] [SRS_FlsTst_14221] [SRS_FlsTst_14222] [SRS_FlsTst_14223] [SRS_FlsTst_14224] [SRS_FlsTst_14225]

Table 5.1: RequirementsTracing

6 References

[1] Standardization Template
AUTOSAR_FO_TPS_StandardizationTemplate

[2] Glossary
AUTOSAR_FO_TR_Glossary

[3] Requirements on AUTOSAR Features
AUTOSAR_CP_RS_Features

A Change history of AUTOSAR traceable items

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

A.1 Traceable item history of this document according to AUTOSAR Release R23-11

A.1.1 Added Requirements in R23-11

none

A.1.2 Changed Requirements in R23-11

none

A.1.3 Deleted Requirements in R23-11

none