| Document Title | Requirements on Flash Driver |
|---|---|
| **Document Owner** | AUTOSAR |
| **Document Responsibility** | AUTOSAR |
| **Document Identification No** | 194 |

| 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 | • Editorial changes<br><br>• Replaced references to BS BRF with RS BRF |
| 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<br><br>• 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 |
| 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 Administration | • Formal rework for requirements tracing<br>• Requirements reworked according to TPS_STDT_00078<br>• Requirements linked to BSW & RTE features |
| 2010-09-30 | 3.1.5 | AUTOSAR Administration | • Legal disclaimer revised |
| 2008-08-13 | 3.1.1 | AUTOSAR Administration | • Legal disclaimer revised |
| 2007-12-21 | 3.0.1 | AUTOSAR Administration | • Document meta information extended<br>• Small layout adaptations made |
| 2007-01-24 | 2.1.15 | AUTOSAR Administration | • "Advice for users" revised<br>• "Revision Information" added |
| 2006-11-28 | 2.1 | AUTOSAR Administration | • Legal disclaimer revised |
| 2006-05-16 | 2.0 | AUTOSAR Administration | • Release as a separate document. The SRS SPAL V1.0.0 has been splint into 15 independent documents for Release 2.0<br>• Requirement SRS_Fls_13301, SRS_Fls_13302 and SRS_Fls_13304 added<br>• Requirement SRS_Fls_12132 changed |
| 2005-05-31 | 1.0 | AUTOSAR Administration | • Initial release as a part of the SRS SPAL V1.0.0 |

**Disclaimer**

# Contents

# 1 Scope of Document

This document specifies requirements on the module Flash Driver.

**Constraints**

First scope for specification of requirements on basic software modules are systems which are not safety relevant. For this reason safety requirements are assigned to medium priority.

# 2 How to read this document

Each requirement has its unique identifier starting with prefix "BSW" (for "Basic Software"). For any review annotations, remarks or questions, please refer to this unique ID rather than chapter or page numbers!

## 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]).

## 2.2 Conventions to be used

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.)

## 2.3   Requirements structure

Each module specific chapter contains a short functional description of the Basic Software Module. Requirements of the same kind within each chapter are grouped under the following headlines (where applicable):

**Functional Requirements**

- Configuration (which elements of the module need to be configurable)
- Initialisation
- Normal Operation
- Shutdown Operation
- Fault Operation
- ...

**Non-Functional Requirements**

- Timing Requirements
- Resource Usage
- Usability
- Output for other WPs (e.g. Description Templates, Tooling,...)
- ...

# 3 Acronyms and abbreviations

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

| Acronym: | Description: |
|---|---|
| CS | Chip select |
| DIO | Digital Input Output |
| ECU | Electric Control Unit |
| EOL | End Of Line |
| | Often used in the term 'EOL Programming' or 'EOL Configuration' |
| ICU | Interrupt Capture Unit |
| MAL | Old name of Microconroller Abstraction Layer (replaced by MCAL because 'MAL' is a french term meaning 'bad') |
| MCAL | Microconroller Abstraction Layer |
| MCU | Microcontroller Unit |
| MMU | Memory Management Unit |
| Master | A device controlling other devices (slaves, see below) |
| Slave | A device beeing completely controlled by a master device |
| NMI | Non maskable interrupt |
| OS | Operating System |
| PLL | Phase Locked Loop |
| PWM | Pulse Width Modulation |
| RX | Reception (in the context of bus communication) |
| SPAL | The name of this working group |
| SFR | Special Function Register |
| RTE | Runtime environment |
| WP | Work Package |

**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**

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 Driver.

## 4.1 Functional Overview

### 4.1.1 Internal Flash Driver

The internal Flash driver provides services for initialization and reading, writing, erasing the internal Flash memory. The Flash driver provides a built-in loader capability that allows loading the flash access code to RAM and execute the write/erase operations from there if this is required.

In application mode of the ECU, the flash driver is only to be used by the Flash EEP-ROM emulation module for writing data. It is not intended to write program code to flash memory in application mode. This shall be done in boot mode which is out of scope of AUTOSAR.

### 4.1.2 External Flash Driver

The external Flash driver provides services for initialization and reading, writing, erasing an external Flash memory. It has the same functional scope as an internal flash driver.

## 4.2 Functional Requirements

### 4.2.1 Internal Flash Driver

#### 4.2.1.1 Configuration

**[SRS_Fls_12132] Flash driver shall be statically configurable** ⌈

| | |
|---|---|
| **Description:** | The following constants of the Flash driver shall be statically configurable: <br><br> 1. Flash memory base address <br><br> 2. Flash memory size <br><br> 3. Maximum block sizes for read (compare), write and erase operations processed within the job processing function in normal mode <br><br> 4. Maximum block sizes for read (compare), write and erase operations processed within the job processing function in fast mode <br><br> 5. Job processing triggered by interrupt or cyclic job processing (polling) function for write and erase <br><br> 6. Call cycle of cyclic job processing function for writeand erase, protect (in case the flash hardware does not provide this timing) <br><br> 7. Flash write protection |
| **Rationale:** | Basic configuration |
| **Use Case:** | 1+2: can also be used for restricting the accessible flash memory area (protect program code from being overwritten) <br><br> 4: Some microcontrollers provide flash memory interrupts <br><br> 5: Needed if the flash memory hardware does not provide this timing and/or deadline checks are necessary |
| **Dependencies:** | – |
| **Supporting Material:** | – |

⌋ (*RS_BRF_01136*)

**[SRS_Fls_12133] Flash memory properties shall be published** ⌈

| | |
|---|---|
| **Description:** | The flash driver description shall publish the following flash memory properties: <br><br> 1. value of erased flash cell <br><br> 2. size of one flash cell (e.g. 8bit, 16bit, ...) <br><br> 3. flash memory size in bytes <br><br> 4. flash memory base address <br><br> 5. physical memory segmentation (minimum writable / readable / erasable / protectable units) |
| **Rationale:** | For configuration of higher layers |

▽

△

| Use Case: | 1: The NVRAM manager wants to perform an flash blank check. For that he needs the value of an erased flash cell. |
| | 5: During NVRAM layout configuration data blocks shall not be misaligned to segmentation borders. |
| Dependencies: | – |
| Supporting Material: | – |

⌋*()*

### 4.2.1.2  Normal Operation

### [SRS_Fls_12134] The flash driver shall provide an asynchronous read function ⌈

| Description: | The flash driver shall provide an asynchronous read function that reads a data block starting from the requested flash address with the passed length from the internal flash memory. |
| Rationale: | Basic functionality |
| Use Case: | Flash EEPROM Emulation; access of flash that is not memory mapped |
| Dependencies: | – |
| Supporting Material: | – |

⌋*()*

### [SRS_Fls_12135] The flash driver shall provide an asynchronous write function ⌈

| Description: | The flash driver shall provide an asynchronous write function that writes a data block starting from the requested flash address with the passed length to the internal flash memory. |
| | The flash address and the length shall be aligned to the physical memory segmentation of the flash memory. Unaligned write requests shall be rejected by the flash driver with an error code. |
| Rationale: | Basic functionality |
| Use Case: | – |
| Dependencies: | – |
| Supporting Material: | – |

⌋*()*

### [SRS_Fls_12136] The flash driver shall provide an asynchronous erase function ⌈

| Description: | The flash driver shall provide an asynchronous erase function that erases one or multiple flash segments starting from the requested flash address with the passed length. |
|---|---|
| | The flash address and the length shall be aligned to the physical memory segmentation of the flash memory. Unaligned erase requests shall be rejected by the flash driver with an error code. |
| | The flash driver shall choose the optimal erase strategy internally. E.g. use block erase commands if supported by flash hardware. |
| Rationale: | Basic functionality |
| Use Case: | – |
| Dependencies: | – |
| Supporting Material: | – |

⌋*()*

### [SRS_Fls_13301] The flash driver shall provide an asynchronous compare function ⌈

| Description: | The flash driver shall provide an asynchronous compare function that compares a section in memory with a section in flash memory with the passed length. |
|---|---|
| Rationale: | The flash driver shall provide the same functionality as the EEPROM driver to allow for transparency towards the NVRAM manager. |
| Use Case: | Internal mechanisms in the Flash EEPROM Emulation can use this function to determine, whether erasing / writing a sector / page is needed or not. |
| Dependencies: | – |
| Supporting Material: | – |

⌋*()*

### [SRS_Fls_12137] The flash driver shall provide a synchronous cancel function ⌈

| Description: | The flash driver shall provide a synchronous cancel function that stops the currently processed job. The states and data of the affected flash cells are undefined! |
|---|---|
| | The flash driver and controller itself is ready for new jobs. |
| | Note: |
| | In most cases, ongoing hardware write/erase processes cannot be stopped, but the writing/erasing of further data blocks is aborted. |
| Rationale: | Needed for EEPROM emulation only (urgent write commands can be performed without any delay). |
| Use Case: | Writing crash relevant data in case of detected vehicle crash without any delay. |

▽

△

| Dependencies: | – |
|---|---|
| Supporting Material: | – |

⌋ *()*

## [SRS_Fls_12138] The flash driver shall provide a synchronous status function ⌈

| Description: | The flash driver shall provide a synchronous function which returns the job processing status. |
|---|---|
| Rationale: | Check if flash driver is busy |
| Use Case: | Only example (will be specified within API definition):<br>• After Reset and before a successful initialization the driver state is UNINIT.<br>• After a successful initialization the driver state is IDLE.<br>• During job processing the driver state is BUSY.<br>• After canceling a job the driver state is IDLE. |
| Dependencies: | – |
| Supporting Material: | – |

⌋ *()*

## [SRS_Fls_13302] The flash driver shall provide a synchronous selection function ⌈

| Description: | The flash driver shall provide a synchronous function that allows to switch the operation mode between normal and fast flash memory access.<br>Comment:<br>For specification of these two modes see the links below. |
|---|---|
| Rationale: | The flash driver shall provide the same functionality as the EEPROM driver to allow for transparency towards the NVRAM manager. |
| Use Case: | – |
| Dependencies: | [SRS_Fls_12132], [SRS_Fls_13304], [SRS_Fls_13303] |
| Supporting Material: | – |

⌋ *(RS_BRF_01144)*

## [SRS_Fls_12159] The write and erase functions of the Flash driver shall check the passed address parameters ⌈

| Description: | The write and erase functions of the Flash driver shall check the passed address parameters for being within the valid configured address borders.<br>Write/erase accesses beyond the allowed borders shall be rejected with an error code. |
|---|---|

▽

△

| Rationale: | Avoid write attempts to not allowed flash areas (e.g. program code). |
|---|---|
| Use Case: | – |
| Dependencies: | [SRS_Fls_12132] |
| Supporting Material: | – |

⌋(RS_BRF_02232)

### [SRS_Fls_12158] Before writing, the flash driver shall verify if the addressed memory area has been erased ⌈

| Description: | Before writing data to flash memory, the flash driver shall verify if the addressed memory area has been erased.<br><br>If the memory is not erased, the processing of the write function shall be aborted with an error notification.<br><br>This feature shall be statically configurable (on/off). |
|---|---|
| Rationale: | Avoid write attempts to not erased flash memory. |
| Use Case: | – |
| Dependencies: | – |
| Supporting Material: | – |

⌋(RS_BRF_02232, RS_BRF_00129)

### [SRS_Fls_12141] The flash driver shall verify written data ⌈

| Description: | The flash driver shall verify written data by reading back from flash and comparing with the source data after each write access.<br><br>Differences shall be notified as error.<br><br>The checking shall be done within the processing of the write function.<br><br>This feature shall be statically configurable (on/off). |
|---|---|
| Rationale: | Detecting data corruption. |
| Use Case: | – |
| Dependencies: | – |
| Supporting Material: | – |

⌋(RS_BRF_02232, RS_BRF_00129)

### [SRS_Fls_12160] After execution of an erase job, the flash driver shall verify that the addressed block has been erased completely ⌈

| Description: | After execution of an erase job, the flash driver shall verify that the addressed block has been erased completely.<br><br>This feature shall be statically configurable (on/off). |
|---|---|

▽

△

| Rationale: | – |
|---|---|
| Use Case: | – |
| Dependencies: | – |
| Supporting Material: | – |

⌋(*RS_BRF_02232*, *RS_BRF_00129*)

### [SRS_Fls_12143] The flash driver shall handle only one job at one time ⌈

| Description: | The flash driver shall handle only one job (write or erase) at one time. Job requests during a running job shall be rejected and handled as error. This error detection shall be statically configurable (on/off). Further explanation: The calling function is responsible for buffering and queueing of jobs, not the flash driver. |
|---|---|
| Rationale: | Different operations like write and erase can't be handled at the same time and the results are dependent of the execution order. |
| Use Case: | During development, the error detection is enabled. For production code, the error detection is disabled for efficiency reasons. |
| Dependencies: | – |
| Supporting Material: | – |

⌋*()*

### [SRS_Fls_12144] The flash driver shall provide a function that has to be called for job processing ⌈

| Description: | The flash driver shall provide a function that has to be called for job processing. All job processing shall be done within this function. If supported by hardware, this function can be called from an interrupt. Otherwise, this function can be called with a fixed cycle time. Further comments for better understanding: The job processing function usually contains a big state machine which processes the write and erase jobs and sets the driver status variable. |
|---|---|
| Rationale: | Allow flexible possibilities of job processing. |
| Use Case: | Example: The job processing function is called every 10ms. |
| Dependencies: | – |
| Supporting Material: | – |

⌋*()*

**[SRS_Fls_13303] In normal mode, one cycle of the job processing function of the flash driver shall limit the block size to the default block size** ⌈

| Description: | In normal mode, one cycle of the job processing function of the flash driver shall limit the block size that is read from flash memory to the configured default block size. |
| :--- | :--- |
| | Simplified comment: |
| | Only read a few bytes during one call of the job processing function. |
| Rationale: | The flash driver shall provide the same functionality as the EEPROM driver to allow for transparency towards the NVRAM manager. |
| Use Case: | Example: |
| | In normal mode, the maximum block size of read data is 16. |
| | In fast mode, the maximum block size of read data is 128. |
| Dependencies: | [SRS_Fls_12132], [SRS_Fls_13302] |
| Supporting Material: | – |

⌋*(RS_BRF_01144)*

**[SRS_Fls_13304] In fast mode, one cycle of the job processing function of the flash driver shall limit the block size to the maximum block size** ⌈

| Description: | In fast mode, one cycle of the job processing function of the flash driver shall limit the block size that is read from flash memory to the configured maximum block size. |
| :--- | :--- |
| | Simplified comment: |
| | Read a big block of data during one call of the job processing function. |
| Rationale: | The flash driver shall provide the same functionality as the EEPROM driver to allow for transparency towards the NVRAM manager. |
| Use Case: | Example: |
| | In normal mode, the maximum block size of read data is 16. |
| | In fast mode, the maximum block size of read data is 128. |
| Dependencies: | [SRS_Fls_12132], [SRS_Fls_13302] |
| Supporting Material: | – |

⌋*(RS_BRF_01144)*

**[SRS_Fls_12193] The flash driver shall load the code that accesses the flash hardware to RAM whenever an erase or write job is started** ⌈

| Description: | The flash driver shall load the code that accesses the flash hardware (internal erase / write routines) to RAM whenever an erase or write job is started. |
| :--- | :--- |
| | This feature shall be statically configurable on/off (pre-compile configuration). |

▽

△

| | |
|---|---|
| *Rationale:* | During an erase / write operation on a flash bank, read access to this bank (and therefore execution of code located in this bank) is not possible. |
| *Use Case:* | The flash bank containing the flash access routines is also the bank currently addressed for an erase / write operation. |
| *Dependencies:* | – |
| *Supporting Material:* | This is only necessary if the erase / write routines are located in the same bank that shall be erased or reprogrammed. |

⌋*()*

## [SRS_Fls_12194] The flash driver shall execute the code that accesses the flash hardware from RAM ⌈

| | |
|---|---|
| *Description:* | The flash driver shall execute the code that accesses the flash hardware (internal erase / write routines) from RAM. This requirement is only applicable if the flash access code has been loaded to RAM. <br><br> The flash driver has to ensure, that this code execution is not interrupted. Therefore the runtime of this routine shall be kept as short as possible. |
| *Rationale:* | During an erase / write operation on a flash bank, read access to this bank (and therefore execution of code located in this bank) is not possible. |
| *Use Case:* | The flash bank containing the flash driver code is also the bank currently addressed for an erase / write operation. |
| *Dependencies:* | [SRS_Fls_12193] |
| *Supporting Material:* | – |

⌋*()*

## [SRS_Fls_13300] The flash driver shall remove the code that accesses the flash hardware from RAM after the current job has been finished or canceled ⌈

| | |
|---|---|
| *Description:* | The flash driver shall remove the code that accesses the flash hardware (internal erase / write routines) from RAM after the current erase or write job has been finished or canceled. <br><br> Removing the flash access code from RAM is only necessary if the flash driver has loaded that code to RAM during start of an erase / write job. If the FAC has been loaded to RAM during initialization the flash driver shall not remove the code from RAM. <br><br> This feature shall be statically configurable on/off (pre-compile configuration). |
| *Rationale:* | The flash access code shall be removed from RAM to avoid possibly harmful operations (flash erase / write) outside of the flash driver's operation. |
| *Use Case:* | The flash access code for erasing the flash memory is loaded at the beginning of an erase job and unloaded after the erase job has finished to prevent further (unwanted) erasure of flash memory. |
| *Dependencies:* | [SRS_Fls_12193] |

▽

△

| Supporting Material: | – |
|---|---|

⌋ *()*

### 4.2.2 External Flash Driver

#### 4.2.2.1 General

**[SRS_Fls_12147] The same requirements shall apply for an external and internal flash driver** ⌈

| Description: | For an external flash driver the same requirements shall apply like for an internal flash driver. |
|---|---|
| Rationale: | Make no functional differences between internal and external flash memory. Keep the functional scope the same. |
| Use Case: | The STAR12 has internal flash memory. Other microcontrollers are using only external flash memory. On both types of microcontrollers the same NVRAM Manager shall be used. |
| Dependencies: | – |
| Supporting Material: | – |

⌋*(RS_BRF_01080, RS_BRF_01008, RS_BRF_01800)*

#### 4.2.2.2 Configuration

**[SRS_Fls_12182] The external flash driver shall allow the static configuration of the hardware flash ID and the suspend time** ⌈

| Description: | In addition to the basic configuration parameters the external flash driver shall allow the static configuration of the following parameters: <br> 1. Expected hardware flash ID <br> 2. Maximum read access blocking time ("suspend time") |
|---|---|
| Rationale: | Basic configuration |
| Use Case: | 1: [SRS_Fls_12107] Check Flash type <br> 2: [SRS_Fls_12184] Limit read access blocking times |
| Dependencies: | – |
| Supporting Material: | – |

⌋*(RS_BRF_01136)*

### 4.2.2.3   Fault operation

**[SRS_Fls_12107] The external flash driver shall check if the configured flash type matches with the hardware flash ID** ⌈

| | |
|---|---|
| ***Description:*** | The external flash driver shall check within it's initialization function if the configured flash type matches with the hardware flash ID. A detected mismatch shall be reported to the Error Manager.<br><br>This check is only to be provided if the flash hardware provides a flash ID. |
| ***Rationale:*** | Avoid use of wrong configuration for programming |
| ***Use Case:*** | – |
| ***Dependencies:*** | [SRS_Fls_12182] |
| ***Supporting Material:*** | Requirements Specification CAS LLD - Configuration Tool: RS_LLD_CONFIG/ 7.11 |

⌋*(RS_BRF_02232)*

## 4.3   Non-Functional Requirements (Qualities)

## 4.4   Non-Functional Requirements

### 4.4.1   Internal Flash Driver

**[SRS_Fls_12145]  The job processing function of the flash driver shall process only as much data as the flash hardware can handle** ⌈

| | |
|---|---|
| ***Description:*** | The job processing function of the flash driver shall process only as much data as the flash hardware can handle in one step (particularly write operation) or as much as a defined user limit (particularly read operation). |
| ***Rationale:*** | Minimize processor load, reduce blocking times. |
| ***Use Case:*** | E.g. the job processing function performs the writing of one byte and the reading of max. 8 bytes during one call. |
| ***Dependencies:*** | – |
| ***Supporting Material:*** | – |

⌋*()*

### 4.4.2 External Flash Driver

**[SRS_Fls_12184] The flash driver shall limit the read access blocking times to the configured time** ⌈

| Description: | The flash driver shall limit the read access blocking times to the configured time (Maximum read access blocking time). |
|---|---|
| Rationale: | Avoid blocking the scheduling and the interrupts of the whole system. |
| Use Case: | Bosch EDC16: blocking time shall be maximum $40\mu$s. |
| Dependencies: | [SRS_Fls_12194], [SRS_Fls_12182] |
| Supporting Material: | – |

⌋ *()*

**[SRS_Fls_12148] The external flash driver shall have a semantically identical API as an internal flash driver** ⌈

| Description: | The external flash driver shall have a semantically identical API as an internal flash driver. |
|---|---|
| Rationale: | Ease Memory Abstraction. Keep handling of internal and external flash memory similar. |
| Use Case: | One ECU uses the STAR12 with internal flash memory. Another ECU uses a controller with only external flash memory. On both microcontrollers the same upper layer (NVRAM Manager, Flash/ EEPROM emulation) shall be used. |
| Dependencies: | Requirements on internal flash driver. |
| Supporting Material: | – |

⌋ *(RS_BRF_01080)*

**[SRS_Fls_12149] The source code of the external flash driver shall be independent from the underlying microcontroller** ⌈

| Description: | The source code of the external flash driver shall be independent from the underlying microcontroller. |
|---|---|
| Rationale: | Reuse of external flash driver across multiple microcontrollers |
| Use Case: | The same external flash driver for a flash device can be used on a NEC V850 and on a MPC563 without any modification. |
| Dependencies: | – |
| Supporting Material: | – |

⌋ *(RS_BRF_01008, RS_BRF_01800)*

# 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_Fls_12141] [SRS_Fls_12158] [SRS_Fls_12160] |
| **[RS_BRF_01008]** | AUTOSAR shall organize the hardware dependent layer in a microcontroller independent and a microcontroller dependent layer | [SRS_Fls_12147] [SRS_Fls_12149] |
| **[RS_BRF_01080]** | AUTOSAR shall allow access to internal and external peripheral devices | [SRS_Fls_12147] [SRS_Fls_12148] |
| **[RS_BRF_01136]** | AUTOSAR shall support variants of configured BSW data resolved after system start-up | [SRS_Fls_12132] [SRS_Fls_12182] |
| **[RS_BRF_01144]** | AUTOSAR shall support configuration parameters which allow to trade interrupt response time against runtime | [SRS_Fls_13302] [SRS_Fls_13303] [SRS_Fls_13304] |
| **[RS_BRF_01800]** | AUTOSAR non-volatile memory functionality shall be divided into a hardware dependent and independent layer | [SRS_Fls_12147] [SRS_Fls_12149] |
| **[RS_BRF_02232]** | AUTOSAR shall support development with run-time assertion checks | [SRS_Fls_12107] [SRS_Fls_12141] [SRS_Fls_12158] [SRS_Fls_12159] [SRS_Fls_12160] |

**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

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

### A.1.1 Added Requirements in R23-11

### A.1.2 Changed Requirements in R23-11

| Number | Heading |
|---|---|
| [SRS_Fls_12107] | The external flash driver shall check if the configured flash type matches with the hardware flash ID |
| [SRS_Fls_12132] | Flash driver shall be statically configurable |
| [SRS_Fls_12133] | Flash memory properties shall be published |
| [SRS_Fls_12134] | The flash driver shall provide an asynchronous read function |
| [SRS_Fls_12135] | The flash driver shall provide an asynchronous write function |
| [SRS_Fls_12136] | The flash driver shall provide an asynchronous erase function |
| [SRS_Fls_12137] | The flash driver shall provide a synchronous cancel function |
| [SRS_Fls_12138] | The flash driver shall provide a synchronous status function |
| [SRS_Fls_12141] | The flash driver shall verify written data |
| [SRS_Fls_12143] | The flash driver shall handle only one job at one time |
| [SRS_Fls_12144] | The flash driver shall provide a function that has to be called for job processing |
| [SRS_Fls_12145] | The job processing function of the flash driver shall process only as much data as the flash hardware can handle |
| [SRS_Fls_12147] | The same requirements shall apply for an external and internal flash driver |
| [SRS_Fls_12148] | The external flash driver shall have a semantically identical API as an internal flash driver |
| [SRS_Fls_12149] | The source code of the external flash driver shall be independent from the underlying microcontroller |
| [SRS_Fls_12158] | Before writing, the flash driver shall verify if the addressed memory area has been erased |
| [SRS_Fls_12159] | The write and erase functions of the Flash driver shall check the passed address parameters |
| [SRS_Fls_12160] | After execution of an erase job, the flash driver shall verify that the addressed block has been erased completely |
| [SRS_Fls_12182] | The external flash driver shall allow the static configuration of the hardware flash ID and the suspend time |
| [SRS_Fls_12184] | The flash driver shall limit the read access blocking times to the configured time |

▽

△

| Number | Heading |
|--------|---------|
| [SRS_Fls_12193] | The flash driver shall load the code that accesses the flash hardware to RAM whenever an erase or write job is started |
| [SRS_Fls_12194] | The flash driver shall execute the code that accesses the flash hardware from RAM |
| [SRS_Fls_13300] | The flash driver shall remove the code that accesses the flash hardware from RAM after the current job has been finished or canceled |
| [SRS_Fls_13301] | The flash driver shall provide an asynchronous compare function |
| [SRS_Fls_13302] | The flash driver shall provide a synchronous selection function |
| [SRS_Fls_13303] | In normal mode, one cycle of the job processing function of the flash driver shall limit the block size to the default block size |
| [SRS_Fls_13304] | In fast mode, one cycle of the job processing function of the flash driver shall limit the block size to the maximum block size |

**Table A.1: Changed Requirements in R23-11**

## A.1.3 Deleted Requirements in R23-11