Document Title	Requirements on Flash Driver
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Version	2.0.2
Document Status	Final
Part of Release	2.1
Revision	0014

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
24.01.2007	2.0.2	AUTOSAR Administration	• "Advice for users" revised • "Revision Information" added
15.12.2006	2.0.1	AUTOSAR Administration	Legal disclaimer revised
21.02.2006	2.0.0	AUTOSAR Administration	Release as a separate document. The SRS SPAL V1.0.0 has been split into 15 independent documents for Release 2.0 • Requirement BSW13301, BSW13302, BSW13303 and BSW13304 added • Requirement BSW12132 changed
11.07.2005	1.0.0	AUTOSAR Administration	Initial release as a part of the SRS SPAL V1.0.0

Page left intentionally blank

Disclaimer

Advice to users of AUTOSAR Specification Documents:

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

Table of Content

1 Scope of this document 6

2 How to read this document 7

2.1 Conventions used 7
2.2 Requirements structure 8

3 Acronyms and abbreviations 9

4 Requirement Specification 10

4.1 Internal Flash Driver 10
4.1.1 Functional overview 10
4.1.2 Functional requirements 10
4.1.2.1 Configuration 10
4.1.2.1.1 [BSW12132] Flash driver static configuration 10
4.1.2.1.2 [BSW12133] Publication of flash properties 11
4.1.2.2 Normal Operation 11
4.1.2.2.1 [BSW12134] Flash read function 11
4.1.2.2.2 [BSW12135] Flash write function 11
4.1.2.2.3 [BSW12136] Flash erase function 12
4.1.2.2.4 [BSW13301] Flash compare function 12
4.1.2.2.5 [BSW12137] Flash cancel function 13
4.1.2.2.6 [BSW12138] Flash driver status function 13
4.1.2.2.7 [BSW13302] Flash driver mode selection function 13
4.1.2.2.8 [BSW12159] Flash address check 14
4.1.2.2.9 [BSW12158] Flash blank check 14
4.1.2.2.10 [BSW12141] Flash write verification 15
4.1.2.2.11 [BSW12160] Flash erase verification 15
4.1.2.2.12 [BSW12143] Flash driver job management 15
4.1.2.2.13 [BSW12144] Flash driver job processing function 16
4.1.2.2.14 [BSW13303] Job processing – normal mode 16
4.1.2.2.15 [BSW13304] Job processing – fast mode 17
4.1.2.2.16 [BSW12193] Load flash access code to RAM on job start 17
4.1.2.2.17 [BSW12194] Execute flash access code from RAM 17
4.1.2.2.18 [BSW13300] Remove flash access code from RAM 18
4.1.3 Non-functional requirements 18
4.1.3.1 [BSW12145] Flash driver job processing execution time 18
4.1.3.2 [BSW12083] Use HIS specification as basis 19
4.2 External Flash Driver 20
4.2.1 Functional Overview 20
4.2.2 Functional Requirements 20
4.2.2.1 General 20
4.2.2.1.1 [BSW12147] Functional scope 20
4.2.2.2 Configuration 20
4.2.2.2.1 [BSW12182] External flash driver static configuration 20
4.2.2.3 Fault operation 21
4.2.2.3.1 [BSW12107] Check Flash type 21

4.2.3 Non-Functional Requirements (Qualities) 21
4.2.3.1 [BSW12184] Limit read access blocking times 21
4.2.3.2 [BSW12148] Common Flash API 21
4.2.3.3 [BSW12149] Microcontroller independency 22
5 References 23
5.1 Deliverables of AUTOSAR 23
5.2 Related standards and norms 23

1 Scope of this 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 the 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 Conventions used

In requirements, the following specific semantics are used (taken from Request for Comment RFC 2119 from the Internet Engineering Task Force IETF)

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 described in RFC 2119. Note that the requirement level of the document in which they are used modifies the force of these words.

- MUST: This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
- MUST NOT: This phrase, or the phrase „SHALL NOT“, means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective "RECOMMENDED", mean 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" mean 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, MUST 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, MUST be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides.)

2.2 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

Acronyms and abbreviations that have a local scope are not contained in the AUTOSAR glossary. These must appear in a local glossary.

Acronym / Abbreviation	***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'
HIS	Herstellerinitiative Software
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
STD	Standard
REQ	Requirement
UNINIT	Uninitialized (= not initialized)

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

4 Requirement Specification

4.1 Internal Flash Driver

4.1.1 Functional overview

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 EEPROM 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 Functional requirements

4.1.2.1 Configuration

4.1.2.1.1 [BSW12132] Flash driver static configuration

Initiator:	DC
Date:	18.11.2005
Short Description:	Flash driver static configuration
Type:	Changed (maximum block sizes for read (and compare) jobs added, duplicated for normal and fast mode configuration)
Importance:	High
Description:	The following constants of the Flash driver shall be statically configurable: 1. Flash memory base address 2. Flash memory size 3. Maximum block sizes for read (compare), write and erase operations processed within the job processing function in normal mode 4. Maximum block sizes for read (compare), write and erase operations processed within the job processing function in fast mode 5. Job processing triggered by interrupt or cyclic job processing (polling) function for write and erase 6. Call cycle of cyclic job processing function for writeand erase, protect (in case the flash hardware does not provide this timing) 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) → 4: Some microcontrollers provide flash memory interrupts → 5: Needed if the flash memory hardware does not provide this timing and/or deadline checks are necessary
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.1.2.1.2 [BSW12133] Publication of flash properties

Initiator:	DC
Date:	14.06.2004
Short Description:	Publication of flash properties
Type:	New
Importance:	High
Description:	The flash driver description shall publish the following flash memory properties: 1. value of erased flash cell 2. size of one flash cell (e.g. 8bit, 16bit, …) 3. flash memory size in bytes 4. flash memory base address 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:	--
Conflicts:	--
Supporting Material:	--

4.1.2.2 Normal Operation

4.1.2.2.1 [BSW12134] Flash read function

Initiator:	DC
Date:	14.06.2004
Short Description:	Flash read function
Type:	Changed (reactivated)
Importance:	High
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:	--
Conflicts:	--
Supporting Material:	--

4.1.2.2.2 [BSW12135] Flash write function

Initiator:	DC
Date:	14.06.2004
Short Description:	Flash write function
Type:	New
Importance:	High
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:	The flash memory shall be erased.
Conflicts:	--
Supporting Material:	--

4.1.2.2.3 [BSW12136] Flash erase function

Initiator:	DC
Date:	14.06.2004
Short Description:	Flash erase function
Type:	New
Importance:	High
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:	--
Conflicts:	--
Supporting Material:	--

4.1.2.2.4 [BSW13301] Flash compare function

Initiator:	DC
Date:	18.11.2005
Short Description:	Flash compare function
Type:	New (copied and adapted from SRS EEPROM because of RfC #6793)
Importance:	High
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:	RfC #6793: 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:	RfC #6793: Provide the same functionality as the EEPROM driver
Conflicts:	--
Supporting Material:	--

4.1.2.2.5 [BSW12137] Flash cancel function

Initiator:	DC
Date:	14.06.2004
Short Description:	Flash cancel function
Type:	New
Importance:	High
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:	The NVRAM manager shall pay attention to the priority of the jobs. Only he is authorized to use this function!
Conflicts:	--
Supporting Material:	--

4.1.2.2.6 [BSW12138] Flash driver status function

Initiator:	DC
Date:	14.06.2004
Short Description:	Flash driver status function
Type:	New
Importance:	High
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): • After Reset and before a successful initialization the driver state is UNINIT. • After a successful initialization the driver state is IDLE. • During job processing the driver state is BUSY. • After canceling a job the driver state is IDLE.
Dependencies:	--
Conflicts:	HIS specification
Supporting Material:	--

4.1.2.2.7 [BSW13302] Flash driver mode selection function

Initiator:	DC
Date:	18.11.2005
Short Description:	Flash driver mode selection function
Type:	New (copied and adapted from SRS EEPROM because of RfC #6793)
Importance:	High
Description:	The flash driver shall provide a synchronous function that allows to switch the operation mode between normal and fast flash memory access.

	Comment: For specification of these two modes see the links below.
Rationale:	RfC #6793: The flash driver shall provide the same functionality as the EEPROM driver to allow for transparency towards the NVRAM manager.
Use Case:	--
Dependencies:	[BSW12132] Flash driver static configuration [BSW13304] Job processing – fast mode [BSW13303] Job processing – normal mode
Conflicts:	--
Supporting Material:	--

4.1.2.2.8 [BSW12159] Flash address check

Initiator:	WP4.2.2.1.12
Date:	06.07.2004
Short Description:	Flash address check
Type:	New
Importance:	High
Description:	The write and erase functions of the Flash driver shall check the passed address parameters for being within the valid configured address borders. 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:	[BSW12132] Flash driver static configuration, items 1 + 2
Conflicts:	--
Supporting Material:	--

4.1.2.2.9 [BSW12158] Flash blank check

Initiator:	WP4.2.2.1.12
Date:	06.07.2004
Short Description:	Flash blank check
Type:	New
Importance:	High
Description:	Before writing data to flash memory, the flash driver shall verify if the addressed memory area has been erased. If the memory is not erased, the processing of the write function shall be aborted with an error notification. This feature shall be statically configurable (on/off).
Rationale:	Avoid write attempts to not erased flash memory.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.1.2.2.10 [BSW12141] Flash write verification

Initiator:	DC
Date:	14.06.2004
Short Description:	Flash write verification
Type:	New
Importance:	High
Description:	The flash driver shall verify written data by reading back from flash and comparing with the source data after each write access. Differences shall be notified as error. The checking shall be done within the processing of the write function. This feature shall be statically configurable (on/off).
Rationale:	Detecting data corruption.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.1.2.2.11 [BSW12160] Flash erase verification

Initiator:	DC
Date:	07.07.2004
Short Description:	Flash erase verification
Type:	New
Importance:	High
Description:	After execution of an erase job, the flash driver shall verify that the addressed block has been erased completely. This feature shall be statically configurable (on/off).
Rationale:	--
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.1.2.2.12 [BSW12143] Flash driver job management

Initiator:	DC
Date:	14.06.2004
Short Description:	Flash driver job management
Type:	New
Importance:	High
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:	--
Conflicts:	--
Supporting Material:	--

4.1.2.2.13 [BSW12144] Flash driver job processing function

Initiator:	DC
Date:	14.06.2004
Short Description:	Flash driver job processing function
Type:	New
Importance:	High
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:	--
Conflicts:	--
Supporting Material:	--

4.1.2.2.14 [BSW13303] Job processing – normal mode

Initiator:	DC
Date:	18.11.2005
Short Description:	Job processing – normal mode
Type:	New (copied and adapted from SRS EEPROM because of RfC #6793)
Importance:	High
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:	RfC #6793: 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:	[BSW12132] Flash driver static configuration [BSW13302] Flash driver mode selection function
Conflicts:	--
Supporting Material:	--

4.1.2.2.15 [BSW13304] Job processing – fast mode

Initiator:	DC
Date:	18.11.2005
Short Description:	Job processing – fast mode
Type:	New (copied and adapted from SRS EEPROM because of RfC #6793)
Importance:	High
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:	RfC #6793: 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:	[BSW12132] Flash driver static configuration [BSW13302] Flash driver mode selection function
Conflicts:	--
Supporting Material:	--

4.1.2.2.16 [BSW12193] Load flash access code to RAM on job start

Initiator:	DC
Date:	25.05.2005
Short Description:	Load flash access code to RAM on job start
Type:	Changed
Importance:	High
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:	--
Conflicts:	--
Supporting Material:	This is only necessary if the erase / write routines are located in the same bank that shall be erased or reprogrammed.

4.1.2.2.17 [BSW12194] Execute flash access code from RAM

Initiator:	DC
Date:	25.05.2005
Short Description:	Execute flash access code from RAM
Type:	Changed (after discussion in 26th SPAL meeting)
Importance:	High
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. 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:	Code execution from RAM must be possible on the target platform. [BSW12193] Load flash access code to RAM on job start
Conflicts:	--
Supporting Material:	--

4.1.2.2.18 [BSW13300] Remove flash access code from RAM

Initiator:	DC
Date:	25.05.2005
Short Description:	Remove flash access code from RAM
Type:	Changed
Importance:	High
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. 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. 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:	[BSW12193] Load flash access code to RAM on job start
Conflicts:	--
Supporting Material:	--

4.1.3 Non-functional requirements

4.1.3.1 [BSW12145] Flash driver job processing execution time

Initiator:	DC
Date:	14.06.2004
Short Description:	Flash driver job processing execution time
Type:	New
Importance:	High
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:	--
Conflicts:	--
Supporting Material:	--

4.1.3.2 [BSW12083] Use HIS specification as basis

Initiator:	CAS
Date:	27.05.2004
Short Description:	Use HIS specification as basis
Type:	New
Importance:	High
Description:	Use HIS specification as basis for specifying the Flash driver
Rationale:	Reuse of specification
Use Case:	--
Dependencies:	--
Conflicts:	Different interface because of different use case (application vs. flash loader) and supply mode (source code vs. binary), same (comparable) functionality.
Supporting Material:	--

4.2 External Flash Driver

4.2.1 Functional Overview

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

4.2.2 Functional Requirements

4.2.2.1 General

4.2.2.1.1 [BSW12147] Functional scope

Initiator:	DC
Date:	14.06.2004
Short Description:	Functional scope
Type:	New
Importance:	High
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:	Requirements on internal flash driver
Conflicts:	--
Supporting Material:	--

4.2.2.2 Configuration

4.2.2.2.1 [BSW12182] External flash driver static configuration

Initiator:	WP4.2.2.1.12
Date:	20.07.2004
Short Description:	External flash driver static configuration
Type:	New
Importance:	High
Description:	In addition to the basic configuration parameters the external flash driver shall allow the static configuration of the following parameters: 1. Expected hardware flash ID 2. Maximum read access blocking time (“suspend time”)
Rationale:	Basic configuration
Use Case:	→ 1: [BSW12107] Check Flash type → 2: [BSW12184] Limit read access blocking times
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.2.2.3 Fault operation

4.2.2.3.1 [BSW12107] Check Flash type

Initiator:	WP4.2.2.1.12
Date:	07.07.2004
Short Description:	Check Flash type
Type:	Changed (reformulated CAS requirement)
Importance:	High
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. 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:	Expected flash type: see [BSW12182] External flash driver static configuration
Conflicts:	--
Supporting Material:	Requirements Specification CAS LLD – Configuration Tool: RS_LLD_CONFIG/ 7.11

4.2.3 Non-Functional Requirements (Qualities)

4.2.3.1 [BSW12184] Limit read access blocking times

Initiator:	Bosch
Date:	20.07.2004
Short Description:	Limit read access blocking times
Type:	New
Importance:	High
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 40µs.
Dependencies:	[BSW12194] Execute flash access code from RAM [BSW12182] External flash driver static configuration
Conflicts:	--
Supporting Material:	--

4.2.3.2 [BSW12148] Common Flash API

Initiator:	DC
Date:	14.06.2004
Short Description:	Common Flash API
Type:	New
Importance:	High
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.
Conflicts:	--
Supporting Material:	--

4.2.3.3 [BSW12149] Microcontroller independency

Initiator:	DC
Date:	14.06.2004
Short Description:	Microcontroller independency
Type:	New
Importance:	High
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:	--
Conflicts:	--
Supporting Material:	--

5 References

5.1 Deliverables of AUTOSAR

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

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

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

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

5.2 Related standards and norms

[5] HIS Flash Driver Specification
www.automotive-his.de/results/flash_programming/
HIS_FlashDriver_v130.pdf