Document Title	Requirements on EEPROM Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	192
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R19-11

Document Change History

Date	Release	Changed by	Change Description
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 obsolete references • Editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	• Editorial changes
2013-03-15	4.1.1	AUTOSAR Administration	• Formal updates to TPS_standardizationTemplate (TPS_STDT_00078) • Traceability of BSWAndRTE_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 • Small layout adaptations made
2007-01-24	2.1.15	AUTOSAR Administration	• “Advice for users” revised • “Revision Information” added
2006-11-28	2.1	AUTOSAR Administration	• Legal disclaimer revised

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2006-05-16	2.0	AUTOSAR Administration	• Release as a separate document. The SRS SPAL V1.0.0 has been split into 12 independent documents for Release 2.0 • minor changes regarding formal issues
2005-05-31	1.0	AUTOSAR Administration	• Initial release as a part of the SRS SPAL V1.0.0

Disclaimer

Table of Contents

1 Scope of document

This document specifies requirements on the module EEPROM Driver.

Constraints

First scope for specification of requirements on basic software modules is 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

- The representation of requirements in AUTOSAR documents follows the table specified in [5].

- In requirements, the following specific semantics are 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 . 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:	***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	Microcontroller 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

Abbreviation:	***Description:***
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 Functional Overview

4.1 Internal EEPROM Driver

The internal EEPROM driver provides services for initialization and reading, writing, erasing to/from internal EEPROM.

4.2 External EEPROM Driver

The external EEPROM driver provides services for initialization and reading, writing, erasing to/from external EEPROM.

5 Requirements Tracing

Requirement	Description	Satisfied by
RS_BRF_01000	AUTOSAR architecture shall organize the BSW in a hardware independent and a hardware dependent layer	SRS_Eep_12053
RS_BRF_01008	AUTOSAR shall organize the hardware dependent layer in a microcontroller independent and a microcontroller dependent layer	SRS_Eep_12053
RS_BRF_01080	AUTOSAR shall allow access to internal and external peripheral devices	SRS_Eep_12124, SRS_Eep_12164
RS_BRF_01136	AUTOSAR shall support variants of configured BSW data resolved after system start-up	SRS_Eep_00096, SRS_Eep_12164
RS_BRF_01808	AUTOSAR non-volatile memory handling shall support different kinds of memory hardware	SRS_Eep_12051, SRS_Eep_12052
RS_BRF_01928	AUTOSAR microcontroller abstraction shall provide access to non-volatile memory hardware	SRS_Eep_00087, SRS_Eep_00088, SRS_Eep_00089, SRS_Eep_00090, SRS_Eep_00091, SRS_Eep_00092, SRS_Eep_00094, SRS_Eep_00095, SRS_Eep_12047, SRS_Eep_12050, SRS_Eep_12072, SRS_Eep_12091, SRS_Eep_12156, SRS_Eep_12157

6 Requirement Specification

6.1 Functional requirements

6.1.1 Internal EEPROM Driver

6.1.1.1 Configuration

6.1.1.1.1 [SRS_Eep_00096] EEPROM driver static shall be configured

⌈

Type:	Valid
Description:	The following constants of the EEPROM driver shall be statically configurable: 1. EEPROM base address 2. EEPROM size (can be equal or smaller than physical EEPROM size) 3. maximum block sizes (write, erase) processed within the job processing function 4. maximum read block sizes for normal and fast EEPROM mode processed within the job processing function 5. call cycle of cyclic job processing function for read, write, erase
Rationale:	Basic configuration
Use Case:	→ 5: needed if the EEPROM hardware does not provide this timing and/or deadline checks are necessary
Dependencies:	[SRS_Eep_12072] Job processing – fast mode [SRS_Eep_12157] Job processing – normal mode
Supporting Material:	--

⌋(RS_BRF_01136)

6.1.1.1.2 [SRS_Eep_12071] The EEPROM properties shall be published

⌈

Type:	Valid
Description:	The EEPROM driver description shall publish the following EEPROM properties: • total physical EEPROM size • value of erased EEPROM cell • size of one EEPROM cell (e.g. 8bit, 16bit, …) • Physical memory segmentation (minimum writable/readable/erasable units)
Rationale:	For configuration of higher layers
Use Case:	--
Dependencies:	--
Supporting Material:	--

⌋()

6.1.1.2 Normal Operation

6.1.1.2.1 [SRS_Eep_00087] The EEPROM driver shall provide an asynchronous read function

⌈

Type:	Valid
Description:	The EEPROM driver shall provide an asynchronous read function that reads a data block starting from the requested EEPROM address with the passed length from the internal EEPROM.

	EEPROM driver shall provide byte-wise data read access.
Rationale:	Basic functionality
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_01928)

6.1.1.2.2 [SRS_Eep_00088] The EEPROM driver shall provide an asynchronous write function

[

Type:	Valid
Description:	The EEPROM driver shall provide an asynchronous write function that writes a data block starting from the requested EEPROM address with the passed length to the internal EEPROM. If an addressed EEPROM cell is not empty, an erase operation shall be done automatically before the write command is executed. If the erasable block doesn't align to the write request, the driver shall buffer and rewrite the additional affected data in the block. EEPROM driver shall provide byte-wise data write access.
Rationale:	Basic functionality
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_01928)

6.1.1.2.3 [SRS_Eep_00089] The EEPROM driver shall provide an asynchronous erase function

[

Type:	Valid
Description:	The EEPROM driver shall provide an asynchronous erase function that erases a data block starting from the requested EEPROM address with the passed length from the internal EEPROM. If the erasable block doesn't align to the erase request, the driver shall buffer and rewrite the additional affected data in the block. The EEPROM driver shall choose the optimal erase strategy internally. E.g. use block erase commands if supported by EEPROM hardware and the passed borders of the data block to be erased fit to a physically erasable block.
Rationale:	For some EEPROM types an erased EEPROM can be programmed much faster.
Use Case:	• ECU Production and fast EOL programming • Fast saving of crash data • Get sure that data is really erased
Dependencies:	--
Supporting Material:	--

](RS_BRF_01928)

6.1.1.2.4 [SRS_Eep_12091] The EEPROM driver shall provide an asynchronous compare function

⌈

Type:	Valid
Description:	The EEPROM driver shall provide an asynchronous compare function that compares a section in memory with a section in EEPROM with the passed length. EEPROM driver shall provide byte-wise data compare access.
Rationale:	Basic functionality
Use Case:	Compare a complete block after a hot reset could speed up the recovery of the system. This function can also be used for verifying complete data blocks after writing to EEPROM.
Dependencies:	--
Supporting Material:	--

⌋(RS_BRF_01928)

6.1.1.2.5 [SRS_Eep_00090] The EEPROM driver shall provide a synchronous cancel function

⌈

Type:	Valid
Description:	The EEPROM driver shall provide a synchronous cancel function that stops the currently processed job. The states and data of the affected EEPROM cells are undefined! The EEPROM driver and controller itself shall be ready for valid jobs.
Rationale:	Urgent write commands can be performed without any delay
Use Case:	Writing crash relevant data in case of detected vehicle crash.
Dependencies:	--
Supporting Material:	--

⌋(RS_BRF_01928)

6.1.1.2.6 [SRS_Eep_00091] The EEPROM driver shall provide a synchronous function which returns the job processing status

⌈

Type:	Valid
Description:	The EEPROM driver shall provide a synchronous function which returns the job processing status.
Rationale:	Check if EEPROM driver is busy
Use Case:	Only example (will be specified within API definition): • After Reset and before a successful initialization the RW state is UNINIT. • After a successful initialization the RW state is READY. • During job processing the RW state is BUSY. • After canceling a job the RW state is READY. • After an detected error the RW state is the associated ERROR_xxx status.
Dependencies:	--
Supporting Material:	--

⌋(RS_BRF_01928)

6.1.1.2.7 [SRS_Eep_12156] The EEPROM driver shall provide a synchronous selection function

[

Type:	Valid
Description:	The EEPROM driver shall provide a synchronous function which allows to switch the operation mode between normal and fast EEPROM access. Comment: For specification of these two modes see the links below.
Rationale:	Fast read operation during ECU start-up, fast write operation during ECU shut down, “cooperative” operation during normal ECU mode.
Use Case:	--
Dependencies:	[SRS_Eep_12072] ,[SRS_Eep_12157],[SRS_Eep_12124]
Supporting Material:	--

](RS_BRF_01928)

6.1.1.2.8 [SRS_Eep_00092] The EEPROM driver shall only write data if at least one data value of the affected erasable block is different from the data value to be written

[

Type:	Valid
Description:	The EEPROM driver shall only write data if at least one data value of the affected erasable block in the EEPROM is different from the data value to be written. This feature shall be statically configurable (on/off).
Rationale:	An erase and write cycle is only done if necessary. Thereby the lifetime of EEPROM is elongated.
Use Case:	The value ‘0x45’ shall be written to an EEPROM cell. This value is already contained in this EEPROM cell.
Dependencies:	--
Supporting Material:	--

](RS_BRF_01928)

6.1.1.2.9 [SRS_Eep_00094] The EEPROM driver shall handle the EEPROM memory segmentation

[

Type:	Valid
Description:	The EEPROM driver shall handle the EEPROM memory segmentation. The read, write, erase and compare functions of the EEPROM driver shall resolve the physical EEPROM block sizes and segment borders by using read-modify-write operations if necessary.
Rationale:	Identical API behavior for different EEPROM types. The EEPROM driver knows the most efficient way to handle and resolve the segmentation.
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_01928)

6.1.1.2.10 [SRS_Eep_00095] The EEPROM driver shall handle only one job at the same time

[

Type:	Valid
Description:	The EEPROM driver shall handle only one job (read or write or erase or

	compare) at the same time. Requested jobs 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 EEPROM driver.
Rationale:	Different operations like read, write, erase, compare can't be handled at the same time and the results are dependent of the execution order.
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_01928)

6.1.1.2.11 [SRS_Eep_12047] The EEPROM driver shall provide a function that has to be called for job processin

[

Type:	Valid
Description:	The EEPROM 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 should be handled by a dedicated module. Further comments for better understanding: The job processing function usually contains a big state machine which processes the read/write/erase jobs and sets the driver status variable.
Rationale:	Allow flexible possibilities of job processing. Fulfill requirements of OS independency.
Use Case:	Example: The job processing function is called every 10ms.
Dependencies:	--
Supporting Material:	--

](RS_BRF_01928)

6.1.1.2.12 [SRS_Eep_12157] In normal mode, one cycle of the job processing function of the EEPROM driver shall limit the block size that is read from EEPROM to the configured default block size

[

Type:	Valid
Description:	In normal mode, one cycle of the job processing function of the EEPROM driver shall limit the block size that is read from EEPROM to the configured default block size. Simplified comment: Only read a few bytes during one call of the job processing function.
Rationale:	Cooperative, non-blocking scheduling of EEPROM driver in normal operation mode.
Use Case:	Example: In normal EEPROM mode, the maximum block size of read data is 16. In fast EEPROM mode, the maximum block size of read data is 128.
Dependencies:	[SRS_Eep_00096] ,[SRS_Eep_12156]
Supporting Material:	--

](RS_BRF_01928)

6.1.1.2.13 [SRS_Eep_12072] In fast mode, one cycle of the job processing function of the EEPROM driver shall limit the block size that is read from EEPROM to the configured maximum block size

[

Type:	Valid
Description:	In fast mode, one cycle of the job processing function of the EEPROM driver shall limit the block size that is read from EEPROM to the configured maximum block size. Simplified comment: Read a big block of data during one call of the job processing function.
Rationale:	Allow fast reading and checking of EEPROM during ECU start-up
Use Case:	Example: In normal EEPROM mode, the maximum block size of read data is 16. In fast EEPROM mode, the maximum block size of read data is 128.
Dependencies:	[SRS_Eep_00096] ,[SRS_Eep_12156]
Supporting Material:	--

](RS_BRF_01928)

6.1.2 External EEPROM Driver

6.1.2.1 General

6.1.2.1.1 [SRS_Eep_12051] The same requirements shall apply for an external and internal EEPROM driver

[

Type:	Valid
Description:	For an external EEPROM driver the same requirements shall apply like for an internal EEPROM driver.
Rationale:	Make no functional differences between internal and external EEPROM. Keep the functional scope the same.
Use Case:	The STAR12 has internal EEPROM. The NEC V850 needs external EEPROM. On both microcontrollers the same NVRAM Manager shall be used.
Dependencies:	--
Supporting Material:	--

](RS_BRF_01808)

6.1.2.2 Configuration

6.1.2.2.1 [SRS_Eep_12164] A driver for an external SPI EEPROM shall allow the static configuration of the required SPI parameters

[

Type:	Valid
Description:	A driver for an external SPI EEPROM shall allow the static configuration of the required SPI parameters. Those parameters are specified by the SPI Handler specification.
Rationale:	Basic configuration of SPI access
Use Case:	Use the SPI EEPROM driver together with other SPI device drivers on the same SPI bus.
Dependencies:	[SRS_Eep_00096] EEPROM driver static configuration.

Supporting Material:	AUTOSAR SWS SPI Handler

](RS_BRF_01080,RS_BRF_01136)

6.1.2.3 Normal Operation

6.1.2.3.1 [SRS_Eep_12124] The EEPROM driver for an external SPI EEPROM device shall access the SPI depending on the current EEPROM mode

[

Type:	Valid
Description:	The EEPROM driver for an external SPI EEPROM device shall access the SPI depending on the current EEPROM mode: • normal EEPROM mode: SPI channel with single byte/word mode • fast EEPROM mode: SPI channel with burst mode
Rationale:	Fast read operation during ECU start-up, non-blocking SPI usage during normal operation.
Use Case:	External SPI EEPROM, SPI baud rate 500kbaud. During ECU start-up the EEPROM is operated in burst mode in order to reduce start-up time. An EEPROM access (32bytes+header) blocks the SPI bus for approx. 560µs. During normal operation, the EEPROM is operated in normal mode so that the SPI bus access of the EEPROM driver does not block a communication request with higher priority.
Dependencies:	--
Supporting Material:	--

](RS_BRF_01080)

6.2 Non-Functional Requirements

6.2.1 Internal EEPROM Driver

6.2.1.1 [SRS_Eep_12050] The job processing function of the EEPROM driver shall process only as much data as the EEPROM hardware can handle

[

Type:	Valid
Description:	The job processing function of the EEPROM driver shall process only as much data as the EEPROM 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:	The job processing function performs the writing of one byte and the reading of max. 8 bytes during one call.
Dependencies:	[SRS_Eep_12157] Job processing – normal mode
Supporting Material:	--

](RS_BRF_01928)

6.2.2 External EEPROM Driver

6.2.2.1 [SRS_Eep_12052] The external EEPROM driver shall have a semantically identical API as an internal EEPROM driver

[

Type:	Valid
Description:	The external EEPROM driver shall have a semantically identical API as an internal EEPROM driver.
Rationale:	Ease Memory Abstraction. Keep handling of internal and external EEPROM similar.
Use Case:	The STAR12 has internal EEPROM. The NEC V850 needs external EEPROM. On both microcontrollers the same NVRAM Manager shall be used.
Dependencies:	--
Supporting Material:	--

](RS_BRF_01808)

6.2.2.2 [SRS_Eep_12053] The source code of the external EEPROM driver shall be independent from the underlying microcontroller

[

Type:	Valid
Description:	The source code of the external EEPROM driver shall be independent from the underlying microcontroller.
Rationale:	Reuse of external EEPROM driver across multiple microcontrollers
Use Case:	The same external EEPROM driver for an SPI EEPROM device can be used on a NEC V850 and on a MPC563 without any modification using the standardized SPI Handler interface.
Dependencies:	--
Supporting Material:	--

](RS_BRF_01008,RS_BRF_01000)

7 References

7.1 Deliverables of AUTOSAR

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[4] General Requirements on SPAL
AUTOSAR_SRS_SPALGeneral.pdf

[5] Software Standardization Template
AUTOSAR_TPS_StandardizationTemplate.pdf