Document Title	Requirements on Memory Services
Document Category	Requirement specification (RS)
Document UID	7
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• No content changes
2025-11-27	R25-11	AUTOSAR Release Management	• No content changes
2024-11-27	R24-11	AUTOSAR Release Management	• No content changes
2023-11-23	R23-11	AUTOSAR Release Management	• Editorial changes
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	• Added [SRS_Mem_00139] • 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	• Editorial changes

△

2016-11-30	4.3.0	AUTOSAR Release Management	• Added Requirements Tracing chapter
2015-07-31	4.2.2	AUTOSAR Release Management	• Requirements linked to BSW features
2014-10-31	4.2.1	AUTOSAR Release Management	• Requirements linked to BSW features
2013-10-31	4.1.2	AUTOSAR Release Management	• Editorial changes
2013-03-15	4.1.1	AUTOSAR Administration	• Formal rework for requirements tracing • Requirements reworked according to [TPS_STDT_00078] • Requirements linked to BSW and RTE features
2010-09-30	3.1.5	AUTOSAR Administration	• Added NVM safety mechanism • Added support for debugging and diagnosis • Added shared use of blocks • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2007-21-21	3.0.1	AUTOSAR Administration	• Removal of requirements belonging to the CRC library • Document meta information extended • Small layout adaptations made
2007-01-24	2.1.15	AUTOSAR Administration	• Removed statement about conformance and mandatory requirements • Legal disclaimer revised • “Advice for users” revised • “Revision Information” added

▽

△

2006-05-16	2.0	AUTOSAR Administration	• Requirements on Block Management Types changed • New requirements, related to RAM Block Types, error detection, reduction of write load, configuration. • Requirements on “Spreading of write accesses” an “detection of incomplete writes” removed
2005-05-31	1.0	AUTOSAR Administration	• Initial release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Scope of Document 6

2 How to read this document 7
2.1 Conventions used . . . 7
2.1.1 Requirements structure . . . 8

3 Acronyms and abbreviations 9

4 Requirements Specification 11
4.1 Functional Overview . . . 11
4.2 Functional Requirements . . . 11
4.2.1 Configuration . . . 11
4.2.2 Initialization . . . 18
4.2.3 Normal Operation . . . 18
4.2.4 Shutdown Operation . . . 28
4.2.5 Fault Operation . . . 30
4.3 Non-Functional Requirements (Qualities) . . . 31
4.3.1 Hardware independence . . . 31

5 References 32

A Change history of AUTOSAR traceable items 33
A.1 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 33
A.1.1 Added Requirements in R26-11 . . . 33
A.1.2 Changed Requirements in R26-11 . . . 33
A.1.3 Deleted Requirements in R26-11 . . . 33
A.2 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 33
A.2.1 Added Requirements in R25-11 . . . 33
A.2.2 Changed Requirements in R25-11 . . . 33
A.2.3 Deleted Requirements in R25-11 . . . 33
A.3 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 34
A.3.1 Added Requirements in R24-11 . . . 34
A.3.2 Changed Requirements in R24-11 . . . 34
A.3.3 Deleted Requirements in R24-11 . . . 34

1 Scope of Document

This document specifies requirements on Basic Software Modules of the following software layers:

- Service Layer

Those modules are of the following type:

- NVRAM Management
- Interfaces

The selection of modules is derived from the Basic Software Module List and the AUTOSAR Layered Software Architecture. The following modules are in scope:

- NVRAM Manager
- Bulk NvData Manager

The requirements are structured in the following way:

- General requirements on Basic Software Modules (other document)
- General requirements which apply to all modules of the NVRAM Management
- Module specific requirements

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

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).

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078].

2.1.1 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)
- Initialization
- 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 Requirements on Memory Services that are not included in the AUTOSAR Glossary [1].

Acronym:	Description:
Basic Storage Object	A "Basic Storage Object" is the smallest entity of a NVRAM Block. Several "Basic Storage Objects" can be used to build a NVRAM Block. A "Basic Storage Object" can reside in different memory locations (RAM/ROM/NV memory).
NVRAM Block	The "NVRAM Block" is the entire structure, which is needed to administrate and to store a block of NV data.
NV data	The data to be stored in the Non-Volatile memory.
Block Management Type	Type of the NVRAM Block. It depends on the (configurable) individual composition of a NVRAM Block in chunks of different mandatory/optional Basic Storage Objects and the subsequent handling of this NVRAM block.
NV Block Header	Additional information included in the NV Block if the mechanism "Static Block ID" is enabled.
RAM Block	The "RAM Block" is a Basic Storage Object. It represents the part of a NVRAM Block, which resides in the RAM. See [SRS_LIBS_08534]
ROM Block	The "ROM Block" is a Basic Storage Object. It represents the part of a NVRAM Block, which resides in the ROM. The "ROM Block" is an optional part of a NVRAM Block.
NV Block	The "NV Block" is a Basic Storage Object. It represents the part of a NVRAM Block, which resides in the NV memory. The "NV Block" is a mandatory part of a NVRAM Block.
Administrative Block	The "Administrative Block" is a Basic Storage Object. It resides in RAM. The Administrative Block contains any RAM data, that are necessary to manage the NVRAM block, for being able to perform processing on it and to deliver status information. The "Administrative Block" is a mandatory part of a NVRAM Block.

Table 3.1: Acronyms used in the scope of this Document

The following UML diagram illustrates the relationship between the acronyms defined in the table above:

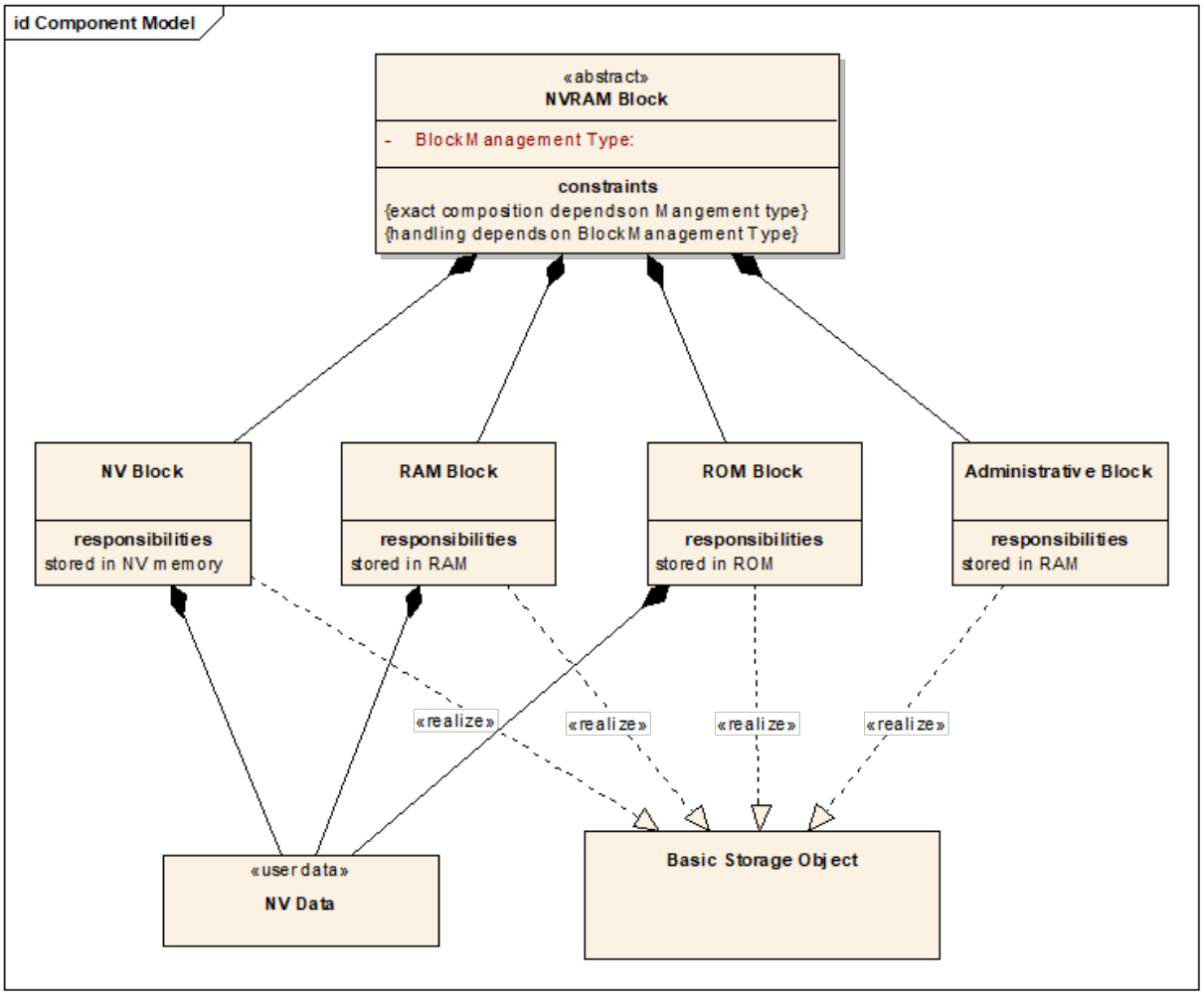


Figure 3.1: Relationship between the acronyms defined in the table in chapter 3

Abbreviation:	Description:
MemHwA	Memory Hardware Abstraction see also [AUTOSAR_SRS_MEMHW]

Table 3.2: Abbreviations used in the scope of this Document

4 Requirements Specification

4.1 Functional Overview

The Non-Volatile RAM Manager (NVRAM Manager) manages the storage of data in all kinds of non-volatile memory.

The NVRAM manager itself shall be hardware independent, all functionality which is directly accessing hardware, e.g. internal or external EEPROM, emulated EEPROM in internal or external flash, etc. is encapsulated in lower layers of the Basic SW. The NVRAM manager handles concurrent accesses to the non-volatile data and provides reliability mechanisms like checksum protection for single data elements. To be usable in all domains of an automotive system, the NVRAM manager needs to be highly scalable (e.g. define the number and size of request queues, support different block management types, EEPROM Emulation, ...).

4.2 Functional Requirements

4.2.1 Configuration

[SRS_Mem_00041] Each application shall be enabled to declare the memory requirements at configuration time ⌈

Description:	By use of configuration techniques each application shall be enabled to declare the memory requirements at configuration time. This information shall be useable to assign memory areas and to generate the appropriate interfaces. Wrong memory assignments and conflicts in requirements (sufficient memory not available) shall be detected at configuration time.
Rationale:	Realization of higher reliability, interoperability
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_08534] The NVRAM manager shall support two classes of RAM data blocks ⌈

Description:	The NVRAM manager shall support two classes of RAM data blocks. These RAM data blocks are mandatory for data exchange between the NVRAM Manager and SW-Components. They have to be provided by the SW-Components or BSW modules. The assignment to a special NVRAM block can be: • permanent, i.e. the RAM data block is assigned to exactly one NVRAM block during configuration time • temporary, i.e. the RAM data block can be assigned to any NVRAM block during runtime
Rationale:	Reduce RAM consumption.
Use Case:	1. Some NVRAM blocks are not frequently used and their data need not to be permanently available in RAM. For example, an application wants to use one RAM block shared for all its NVRAM blocks, which are used mutual exclusive. 2. Diagnostic service wants to read out data from NV that is used by some application, without endangering application's RAM block. It uses one RAM block (must be large enough) to read out any requested block of NV data.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_08528] The NVRAM manager shall allow the configuration of native Block management types ⌈

Description:	The NVRAM manager shall allow the configuration of native Block management types. • A native NVRAM block shall provide a basic storage of data and ROM configurable ROM defaults.
Rationale:	Basic block type
Use Case:	Regular NVRAM data without special requirements.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_08529] The NVRAM manager shall allow the configuration of redundant Block management types ⌈

Description:	The NVRAM manager shall allow the configuration of redundant Block management types. A redundant NVRAM Block shall be transparent to application, and it shall be able to provide data in case of inconsistencies (e.g. incomplete write). A redundant NVRAM Block shall be configurable to include default values. • Note: Redundancy does not necessarily mean double (or multiple) storage.
Rationale:	Safety, enhanced data availability, integrity
Use Case:	For saving safety relevant data (e.g. immobilization data)
Dependencies:	[SRS_Mem_00038] Treatable errors shall not affect the application [SRS_Mem_00129] Automatic data repair
Supporting Material:	–

⌋

[SRS_Mem_08531] The NVRAM manager shall allow the configuration of dataset Block management type ⌈

Description:	The NVRAM manager shall allow the configuration of dataset Block management type. Such NVRAM block shall provide a configurable number of selectable elements in NV. • A dataset NVRAM Block shall be configurable to include default values.
Rationale:	Make runtime selection of different datasets in ROM/NVRAM possible.
Use Case:	One ECU is used within several models/country variants of a car. Depending on the car variant, the corresponding data set in NVRAM or ROM is selected.
Dependencies:	[SRS_Mem_08007] Selection of datasets
Supporting Material:	–

⌋

[SRS_Mem_08543] The priority of each NVRAM block shall be statically configurable in different levels ⌈

Description:	The priority of each NVRAM block shall be statically configurable (pre-compile time) in different levels. One of these levels shall be "immediate", for those blocks [SRS_Mem_08542] shall apply.
Rationale:	Flexibility
Use Case:	Writing of crash data and writing of regular data
Dependencies:	[SRS_Mem_08542] Job order prioritization

▽

△

Supporting Material:	–

⌋

[SRS_Mem_08009] The NVRAM Manager shall allow a static configuration of a default write protection (on/off) for each NVRAM block ⌈

Description:	The NVRAM Manager shall allow a static configuration of a default write protection (on/off) for each NVRAM block.
Rationale:	Some data are read-only
Use Case:	Write protection off: data generated by the application itself (e.g. adaptive data, error memory) Write protection on: data given to the ECU from outside (e.g. EOL data, variant coding, parameters)
Dependencies:	[SRS_Mem_00127] Write protect/unprotect function
Supporting Material:	–

⌋

[SRS_Mem_00135] The NVRAM manager shall have an unique configuration identifier ⌈

Description:	The NVRAM manager shall have a configuration identifier that is a unique property of the non-volatile memory configuration. The ID can be either statically assigned to the configuration or it can be calculated from the configuration properties. The ID must be changed if the block configuration changes, i.e. if a block is added or removed, or if its size or type is changed. The ID shall be stored separately and shall be used to determine the validity of the NVRAM contents. The mechanism shall be robust during production (e.g. power-off).
Rationale:	The NVRAM data configuration may change between different versions of the application software. There must be a way to know if the contents of the NVRAM match the configuration expected by the application.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_08549] The NVRAM manager shall provide functionality to automatically initialize RAM data blocks after a software update ⌈

Description:	The NVRAM manager shall provide functionality to automatically initialize RAM data blocks with ROM defaults, after a software update. This shall be configurable per NVRAM block. The whole functionality shall be statically configurable.
Rationale:	Flexibility. So there is no need to update the NV memory during ECU reprogramming.
Use Case:	After ECU reprogramming (mismatch of Configuration IDs) NVRAM blocks are initialized with (new) ROM defaults, except immobilizer data, which shall not be updated.
Dependencies:	[SRS_Mem_00135] NVRAM configuration ID
Supporting Material:	–

⌋

[SRS_Mem_00125] For each block a notification shall be configurable ⌈

Description:	For each block a notification shall be configurable (which is initiated at completion of read/write jobs).
Rationale:	Flexibility, integration with application software
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_08000] The NVRAM manager shall be able to access multiple non-volatile memory devices ⌈

Description:	The NVRAM manager shall be able to access multiple non-volatile memory devices. The non-volatile memory devices can be of different type.
Rationale:	Flexibility, integration with application software
Use Case:	Multiple Non-volatile memories in one ECU, e.g. external EEPROM and EEPROM Emulation in internal flash
Dependencies:	[SRS_Mem_00011] (Memory) hardware independence
Supporting Material:	–

⌋

[SRS_Mem_08001] The NVRAM manager shall have a configurable consistency check for each block ⌈

Description:	The NVRAM manager shall have a configurable consistency check for each block.
Rationale:	Consistency check might not be used for all blocks
Use Case:	Data that are read or written can be checked immediately by checksum recalculation and comparison.
Dependencies:	[SRS_Mem_00030] Consistency/integrity check of data [BSW023] Detection of incomplete write operations
Supporting Material:	–

⌋

[SRS_Mem_08551] The maximum number of retries in case of an error shall be separately configurable ⌈

Description:	For read and write operations of the NVRAM manager, the maximum number of retries in case of an error shall be separately configurable.
Rationale:	Recover from temporary error situation
Use Case:	Reading and writing of data under error conditions.
Dependencies:	[SRS_Mem_08554] Retrying of read and write operations
Supporting Material:	–

⌋

[SRS_Mem_08552] The NVRAM manager shall provide a configurable verification of the unique block identifier ⌈

Description:	The NVRAM manager shall provide a configurable verification of the unique block identifier when reading an NVRAM block.
Rationale:	Protection against misaddressing of blocks if required
Use Case:	Hardware addressing error can be detected if required
Dependencies:	[SRS_Mem_08555] Verification of unique block identifier
Supporting Material:	–

⌋

[SRS_Mem_08553] The NVRAM manager shall provide a configurable write verification of data ⌈

Description:	The NVRAM manager shall provide a configurable write verification of data by reading again the previously written data and comparing it. Write verification shall be configurable per NVRAM block.

▽

△

Rationale:	Verification of correctly stored data if required
Use Case:	Hardware write errors can be detected if required
Dependencies:	[SRS_Mem_08556] Write verification for data
Supporting Material:	–

⌋

[SRS_Mem_08538] It shall be statically configurable which blocks are loaded automatically during startup of the NVRAM manager ⌈

Description:	It shall be statically configurable which blocks are loaded automatically during startup of the NVRAM manager.
Rationale:	Allow flexibility - what is loaded during start-up
Use Case:	Load only those blocks needed to start communication via CAN within strict timing constraints.
Supporting Material:	Only applicable for NVRAM Blocks configured with permanent RAM Blocks

⌋

[SRS_Mem_08546] It shall be possible to protect permanent RAM data blocks against data loss due to reset ⌈

Description:	For each NVRAM block with permanent RAM data block it shall be a configurable option to enforce mechanisms increasing integrity of RAM data in case of resets.
Rationale:	In case of resets due to voltage drops the RAM content might remain valid. This cannot be detected (or guaranteed) by all platforms. It increases robustness, if the most recent data (from RAM) can be delivered at startup.
Use Case:	A currently moving sun-roof control can neither be fed with outdated position data after a reset occurred (see [SRS_Mem_08011]) nor it is acceptable to loose the position on every voltage drop.
Dependencies:	[SRS_Mem_08545] Validation of RAM data and update of integrity information
Supporting Material:	–

⌋

[SRS_Mem_08560] Each NVRAM block shall be configurable for shared access ⌈

Description:	Each NVRAM block shall be configurable for shared access.
Rationale:	This is on a block-by-block basis to keep memory consumption low.
Use Case:	–

▽

△

Supporting Material:	–

⌋

4.2.2 Initialization

[SRS_Mem_08533] The NVRAM manager shall provide a service to check and load those NVRAM blocks, configured to have a permanent RAM data block to RAM ⌈

Description:	The NVRAM manager shall provide a service to check and load those NVRAM blocks, configured to have a permanent RAM data block to RAM. This service has to be called by the ECU state manager once during start-up before the application is running.
Rationale:	The application needs RAM blocks with valid data after ECU start-up.
Use Case:	ECU Start-up: provide NV data for application.
Dependencies:	[SRS_Mem_08538] Static configuration of NVRAM blocks being loaded during start-up
Supporting Material:	–

⌋

4.2.3 Normal Operation

Note: Only NVRAM manager shall access non-volatile memory. All other software shall exclusively use NVRAM manager to access data in non-volatile memory. However, this is not a requirement on Memory Services but on the usage. Therefore, the former requirement [BSW176] ("Only access non-volatile memory via NVRAM manager") has been removed.

[SRS_Mem_00027] The NVRAM manager shall provide an implicit way of accessing blocks in the NVRAM and in the shared memory (RAM). ⌈

Description:	The NVRAM manager only provides an implicit way of accessing blocks in the NVRAM and in the shared memory (RAM). This means, the NVRAM manager copies one or more blocks from NV memory to RAM block(s) and other way round. Explicit read and write access of variables inside a RAM block has to be provided by application (access macro ...).
Rationale:	Basic functionality of NVRAM manager

▽

△

Use Case:	Accessing of parameters
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_08014] The NVRAM manager shall allow a non-continuous RAM block allocation in the global RAM area ⌈

Description:	RAM block allocation can be defined as not continuous in the global RAM area
Rationale:	–
Use Case:	Each RAM block can be allocated without address constraint in the global RAM area
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_00013] The NVRAM manager shall provide a mechanism to handle multiple, concurrent read / write requests ⌈

Description:	The NVRAM manager shall provide a mechanism to handle multiple, concurrent read / write orders, e.g. queuing. Implementation requirement: if queuing is used, only the read / write orders are buffered, not the data to be written!
Rationale:	The NVRAM manager handles all accesses to NVRAM in the entire SW system. Therefore concurrent accesses are most likely to occur. The NVRAM manager must process these parallel accesses in a serial manner due to resource restrictions.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_00016] The NVRAM manager shall provide functionality to read out data associated with an NVRAM block from the non-volatile memory ⌈

Description:	The NVRAM manager shall provide functionality to read out data associated with an NVRAM block from the non-volatile memory. The NVRAM block is referenced by a unique identifier. The NVRAM data is copied into the corresponding RAM block. The availability of this service shall be configurable.

▽

△

Rationale:	Basic functionality of NVRAM manager
Use Case:	Reading of application data from NV.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_00017] The NVRAM manager shall provide functionality to store data associated with an NVRAM block in the non-volatile memory ⌈

Description:	The NVRAM manager shall provide functionality to store data associated with an NVRAM block in the non-volatile memory. The NVRAM block is referenced by a unique identifier. The data is copied from RAM block into the corresponding NV block. The availability of this service shall be configurable.
Rationale:	Basic functionality of NVRAM manager
Use Case:	Non volatile storage of data before ECU power supply is switched off
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_08554] The NVRAM manager shall retry read and write operations on NVRAM blocks if they have not succeeded up to a configurable number of times ⌈

Description:	The NVRAM manager shall retry read and write operations on NVRAM blocks if they have not succeeded up to a configurable number of times.
Rationale:	Recover from temporary error situation
Use Case:	An electromagnetic pulse has altered the data just read and checksum check has therefore failed. Reading again the data recovers from this error situation.
Dependencies:	[SRS_Mem_08551] Configuration of maximum read and write retries
Supporting Material:	–

⌋

[SRS_Mem_08541] The NVRAM manager shall guarantee that an accepted write request will be processed ⌈

Description:	The NVRAM manager shall guarantee that an accepted write request will be processed.
Rationale:	An application issuing a write request expects the data to be written.

▽

△

Use Case:	1. Queued write jobs originating from the application shall not be influenced by the shutdown process or it's cancellation. 2. On a minor crash the ECU remains operational, the NVRAM Manager shall behave like without this crash. Therefore the request for writing corresponding crash data shall not abort any accepted write job.
Dependencies:	[SRS_Mem_00017],[SRS_Mem_08535],[SRS_Mem_08540], [SRS_Mem_08542]
Supporting Material:	–

⌋

[SRS_Mem_00018] The NVRAM manager shall provide functionality to restore an NVRAM block's associated data from ROM defaults ⌈

Description:	The NVRAM manager shall provide functionality to restore an NVRAM block's associated data from ROM defaults. The availability of this service shall be configurable.
Rationale:	–
Use Case:	Radio factory settings
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_08548] The NVRAM Manager shall request default data from the application ⌈

Description:	If there is no ROM Block available at configuration time, the NVRAM Manager shall request default data from the application. This shall be configurable.
Rationale:	Flexibility
Use Case:	Calibration data cannot be provided by constant data.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_08547] The NVRAM Manager shall be able to distinguish between explicitly invalidated and inconsistent data ⌈

Description:	The NVRAM Manager shall be able to distinguish between explicitly invalidated and inconsistent data.
Rationale:	Explicitly invalidated (or even blank) data blocks shall not denote an error condition to be reported to the DEM.

▽

△

Use Case:	If the NVRAM manager detects inconsistent data, e.g. due to an aborted write operation or CRC error, it shall report an error. On invalidated data, no error will be reported.
Dependencies:	
Supporting Material:	[SRS_MemHwAb_14014], [SRS_MemHwAb_14015], [SRS_MemHwAb_14016]

⌋

[SRS_Mem_08550] The NVRAM Manager shall provide a service for marking permanent RAM data blocks as modified/unmodified ⌈

Description:	The NVRAM Manager shall provide a service for marking permanent RAM data blocks as modified/unmodified. On shutdown only data that have been marked as modified shall be saved to NV. The availability of this service shall be configurable.
Rationale:	Write cycle reduction, speed-up shutdown.
Use Case:	Application knows best when to write back data to NV memory. Using this will reduce the number of required write cycles, and thus the need for walking blocks and NV memory consumption, as well as speed up shut-down operation due to possibly slow write because of significant overhead due to underlying HW (providing pre-erased memory).
Dependencies:	[SRS_Mem_08546] Protection of RAM data blocks against data loss.
Supporting Material:	–

⌋

[SRS_Mem_08545] The NVRAM Manager shall provide a service for marking the permanent RAM data block of an NVRAM block valid ⌈

Description:	The NVRAM Manager shall provide a service for marking the permanent RAM data block of an NVRAM block valid. Additionally this service shall update integrity information, if configured. The availability of this service shall be configurable.
Rationale:	Save only valid data to NV. On start-up increase the possibility to use the most recent data in RAM, rather than reload old data from NV.
Use Case:	A failure on Read may result in invalid RAM content. This content shall not be saved to NV memory. In this case the application is responsible to handle this condition, by presenting data and invoking this service to mark them as valid.
Dependencies:	[SRS_Mem_08546] Protection of RAM data blocks against data loss
Supporting Material:	–

⌋

[SRS_Mem_08011] The NVRAM manager shall provide a service to invalidate a block of data in the non-volatile memory ⌈

Description:	The NVRAM manager shall provide a service to invalidate a block of data in the non-volatile memory. The block is referenced by a unique identifier. The availability of this service shall be configurable.
Rationale:	Avoid loading of outdated data, especially position information.
Use Case:	The position of a sunroof (incremental motor position) has to be saved to an NVRAM block after each movement. Before every new movement of the sunroof this position has to be marked as invalid. Otherwise a reset during sunroof operation would cause an invalid sunroof position (because the old position from NVRAM is loaded). This could result in a damage of the sunroof mechanics (when crashing to end position slide open) or in a serious injury.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_08544] The NVRAM manager shall provide a service to erase the NV block(s) associated with an NVRAM block ⌈

Description:	The NVRAM manager shall provide a service to erase the NV block(s) associated with an NVRAM block. The NVRAM block is referenced by a unique identifier. The availability of this service shall be configurable.
Rationale:	Explicit erase to support the use cases listed below.
Use Case:	Writing of crash data: no delay due to erase-before-write, therefore pre-erase NV block must be done at application-defined point (e.g. via diagnostic service). This cannot be done automatically.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_08007] The NVRAM manager shall provide a service for the selection of valid dataset NV blocks ⌈

Description:	The NVRAM manager shall provide a service for the selection of valid dataset NV blocks. This service shall associate a dataset number (ROM or NVRAM) to the corresponding RAM block.
Rationale:	Make runtime selection of different datasets in ROM/NVRAM possible.

▽

△

Use Case:	One ECU is used within several models/country variants of a car. Depending on the car variant, the corresponding data set in NVRAM or ROM is selected.
Dependencies:	–
Supporting Material:	–BSW08006 , Block management type–dataset

⌋

[SRS_Mem_08542] The NVRAM manager shall provide a prioritization for job processing order ⌈

Description:	The NVRAM manager shall provide a prioritization for job processing order. The highest priority job shall be processed first; jobs with the same priority level shall be executed in FIFO order. This prioritization shall be configurable. If disabled, jobs shall be executed in FIFO order.
Rationale:	Some data must be written faster than other. Therefore processing them shall take precedence. Immediate data shall preempt a running lower priority operation.
Use Case:	Writing of crash data Priority based data saving like NVRAM Manager of SiemensVDO
Dependencies:	[SRS_Mem_08543] ,[SRS_Mem_08541]
Supporting Material:	–

⌋

[SRS_Mem_00020] The NVRAM manager shall provide functionality to read out the status of read/write operations ⌈

Description:	The NVRAM manager shall provide functionality to read out the status of read/ write operations.
Rationale:	The NVRAM manager shall support a variety of non-volatile memory devices. The access time of some devices is long in comparison to the real-time requirements of the ECU. Therefore the NVRAM manager shall provide asynchronous API with a functionality to determine the current status of operations.
Use Case:	Read/write of serial EEPROM
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_00127] The NVRAM manager shall allow enabling/disabling a write protection for each NVRAM block individually ⌈

Description:	The NVRAM manager shall allow enabling/disabling a write protection for each NVRAM block individually. The data area of RAM block of write-protected blocks can be changed without restrictions. If an NV block is protected, further write access jobs for this block are rejected. NVRAM blocks with enabled write protection cannot be saved to NV memory. After reset, the write protection shall be set according to the initial write protection. The availability of this service shall be configurable
Rationale:	Some data blocks are not to be changed by the application, only by special EOL (end of line)/diagnostic services
Use Case:	Coding data, EOL (end of line) data.
Dependencies:	[SRS_Mem_08009] Default write protection of blocks
Supporting Material:	–

⌋

[SRS_Mem_00030] The NVRAM manager shall implement mechanisms for consistency/integrity checks of data saved in NVRAM ⌈

Description:	The NVRAM manager shall implement mechanisms for consistency/integrity checks of data saved in NVRAM during operations even in case of asynchronous reset or power loss. Bit-flipping shall be covered as well. Implementation hint: Checksums are one possibility; additionally write access bytes (valid/invalid) can be used.
Rationale:	Detection of errors.
Use Case:	Signatures for the detection of Bit-flipping.
Dependencies:	[SRS_Mem_08001] Configuration of consistency check of data
Supporting Material:	–

⌋

[SRS_Mem_08555] The NVRAM manager shall provide mechanisms for static verification of the block identifier when reading an NVRAM block ⌈

Description:	The NVRAM manager shall provide mechanisms for static verification of the block identifier when reading an NVRAM block.
Rationale:	Protection against misaddressing of blocks
Use Case:	Hardware addressing error can be detected
Dependencies:	[SRS_Mem_00030] Consistency/integrity check of data

▽

△

Supporting Material:	–

⌋

[SRS_Mem_08556] The NVRAM manager shall provide a mechanism for verification of the written block data by again reading and comparing it ⌈

Description:	The NVRAM manager shall provide a mechanism for verification of the written block data by again reading and comparing it.
Rationale:	Protection against wrongly written data
Use Case:	Hardware write errors can be detected
Dependencies:	[SRS_Mem_08553] Configuration of write verification for data
Supporting Material:	–

⌋

[SRS_Mem_00034] Write accesses of the NVRAM manager to persistent memory shall be executed quasi-parallel to normal operation of the ECU ⌈

Description:	Write accesses of the NVRAM manager to persistent memory shall be executed quasi-parallel (concurrent) to normal operation of the ECU. Data consistency must not be endangered.
Rationale:	The write access of some types of memories is by order of magnitude slower compared to processor register accesses.
Use Case:	EEPROM write accesses.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_08558] The NVRAM manager shall provide a mechanism to remove all unprocessed requests associated with a NVRAM block ⌈

Description:	The NVRAM manager shall provide a mechanism to remove all unprocessed requests associated with a NVRAM block
Rationale:	A request originating from a partition (for example, from a SW-C belonging to that partition) prior to the termination and subsequent restart of that partition should not be processed (and therefore no completion should be reported).
Use Case:	Dealing with termination and restart of partitions.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_00136] The NVRAM manager shall provide functionality for determining updates of data associated with an NVRAM Block during runtime ⌈

Description:	The NVRAM manager shall provide functionality for determining updates of data associated with an NVRAM Block during runtime. The NVRAM Block is referenced by a unique identifier. This functionality will allow skipping write operations to NV memory in case the RAM Block was not updated since last read or write operation. The availability of this functionality shall be configurable.
Rationale:	Determining updates of non-volatile data in RAM Blocks.
Use Case:	Avoid superfluous storing operations.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_00138] The NVRAM manager shall provide a function which triggers first or re-initialization of selected blocks both in RAM and on NV memory ⌈

Description:	There shall be a multiblock operation available which does the following on selected blocks with permanent RAM resp. explicit synchronization: (1) perform explicit recovery from ROM block and/or initialization callback (2) write these contents back to the NV memory media. If these selected blocks do not have a recovery data source (i.e. ROM block and/or initialization callback), invalidation of this NV block shall be performed instead of writing recovery data. In case of DATASET blocks, all NV block instances of this NvM block shall be invalidated if selected for first initialization.
Rationale:	Don't let the application or separately managed factory procedures do the entire job.
Use Case:	(1) Factory (re-) initialization (2) clearing NV memory at development time (3) Re-initialization via tester access
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_00139] Large block support with delta update ⌈

Description:	Larger blocks with delta update only on the changed fragments shall be supported.
Rationale:	The fragmentation to equal blocks together with a delta update of only changed blocks shall take care for a better flash utilization (especially for saving write cycles).
Use Case:	Big data collections with only local updates (e.g. log of driving cycle statistics over the last 100 cycles).
Dependencies:	–
Supporting Material:	–

⌋

4.2.4 Shutdown Operation

[SRS_Mem_08535] The NVRAM manager shall provide a function, which triggers update of integrity information and saving of RAM data blocks to NV memory ⌈

Description:	The NVRAM manager shall provide a function, which triggers update of integrity information (recalculate a checksum, e.g. CRC) and saving of RAM data blocks to NV memory. This function can only affect permanent RAM data blocks because temporary RAM data blocks cannot be written back automatically, e.g. triggered by the ECU state manager. Information: • This function has to be called by the ECU state manager once before shutdown • This function can be called by a diagnostic service (’Save NVRAM’)
Rationale:	Don’t let the application do the entire job.
Use Case:	ECU Shut-down: save NV data of application
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_08540] The NVRAM manager shall provide a function for aborting the shutdown process ⌈

Description:	If during an ECU shutdown an ECU wake-up condition is detected, the NVRAM manager shall provide a function to abort those write jobs originating from the shutdown process. Those write requests in the queue originating from the application shall not be affected by this cancellation routine. This cancellation shall not be destructive, i.e. the NVRAM Block currently being written shall be completed.
Rationale:	Allow fast reaction to wake-up condition during ECU shut down procedure.
Use Case:	ECU Shutdown: the NVRAM manager has started saving all RAM blocks to NVRAM. During this job processing, the ECU wake-up condition occurs. The ECU has to be operable within the given time limits (e.g. 100ms). It is not acceptable that the NVRAM Manager finishes all write jobs which may take e.g. 800 ms.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_00137] The NVRAM manager shall provide a service for auto-validating NVRAM blocks ⌈

Description:	The NVRAM manager shall provide a service for auto-validating NVRAM blocks. This service has to be called by the ECU state manager once in the early phase of ECU shutdown. The availability of this service shall be configurable.
Rationale:	Avoid recovery of a RAM Block with ROM default data in case the RAM Block data is valid.
Use Case:	ECU Shut-down: normal shutdown cannot be completed (e.g. due to time outs, power drops).
Dependencies:	–
Supporting Material:	–

⌋

4.2.5 Fault Operation

[SRS_Mem_00038] Treatable errors shall not affect other software components ⌈

Description:	The NVRAM manager shall not report treatable (healable or recoverable) errors to other software components.
Rationale:	Separation of concerns, avoidance of unnecessary degradation of functionality
Use Case:	Data is saved redundantly (within two identical blocks). If a data corruption is detected in one block, the contents of the redundant block are used. The application is not affected.
Dependencies:	[SRS_Mem_00129] Automatic data repair
Supporting Material:	–

⌋

[SRS_Mem_00129] The NVRAM manager shall repair data in blocks of management type 'NVRAM redundant' ⌈

Description:	The NVRAM manager shall repair data in blocks of management type 'NVRAM redundant' if a data corruption is detected and valid data can be derived. The number of repair cycles shall be limited in order to avoid infinite loops.
Rationale:	Robustness
Use Case:	Data corruption (bit flipping) in a redundant NVRAM block
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Mem_08010] The NVRAM manager shall copy the ROM default data to the data area of the corresponding RAM block if it can not read data from NV into RAM ⌈

Description:	If the NVRAM manager cannot read data from NV into RAM, it shall copy the ROM default data to the data area of the corresponding RAM block.
Rationale:	Robustness
Use Case:	The calibration data set is damaged. A default calibration data set is loaded.
Dependencies:	–
Supporting Material:	See all Blocks management type

⌋

[SRS_Mem_08015] Some of the NV Blocks in the NVRAM shall never be erased nor be replaced with the default ROM data after first initialization ⌈

Description:	After ECU reprogramming some NVRAM data has to be kept secure. This means that some of the NV Blocks in the NVRAM should never be erased nor be replaced with the default ROM data after first initialization.
Rationale:	Immobilizer code or vehicle identification number
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

Note: This is a requirement on configuration / bootloader.

4.3 Non-Functional Requirements (Qualities)

4.3.1 Hardware independence

[SRS_Mem_00011] The NVRAM manager shall be independent from its underlying memory hardware. ⌈

Description:	Existing (standardized) interfaces shall be used by NVRAM-Manager to access the underlying memory hardware. The interfaces shall abstract the memory hardware.
Rationale:	Portability, reusability, hardware cost reduction by flexible usage of memory devices
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

5 References

[1] Glossary
AUTOSAR_FO_TR_Glossary

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 R26-11

A.1.1 Added Requirements in R26-11

none

A.1.2 Changed Requirements in R26-11

none

A.1.3 Deleted Requirements in R26-11

none

A.2 Traceable item history of this document according to AUTOSAR Release R25-11

A.2.1 Added Requirements in R25-11

none

A.2.2 Changed Requirements in R25-11

none

A.2.3 Deleted Requirements in R25-11

none

A.3 Traceable item history of this document according to AUTOSAR Release R24-11

A.3.1 Added Requirements in R24-11

none

A.3.2 Changed Requirements in R24-11

none

A.3.3 Deleted Requirements in R24-11

none