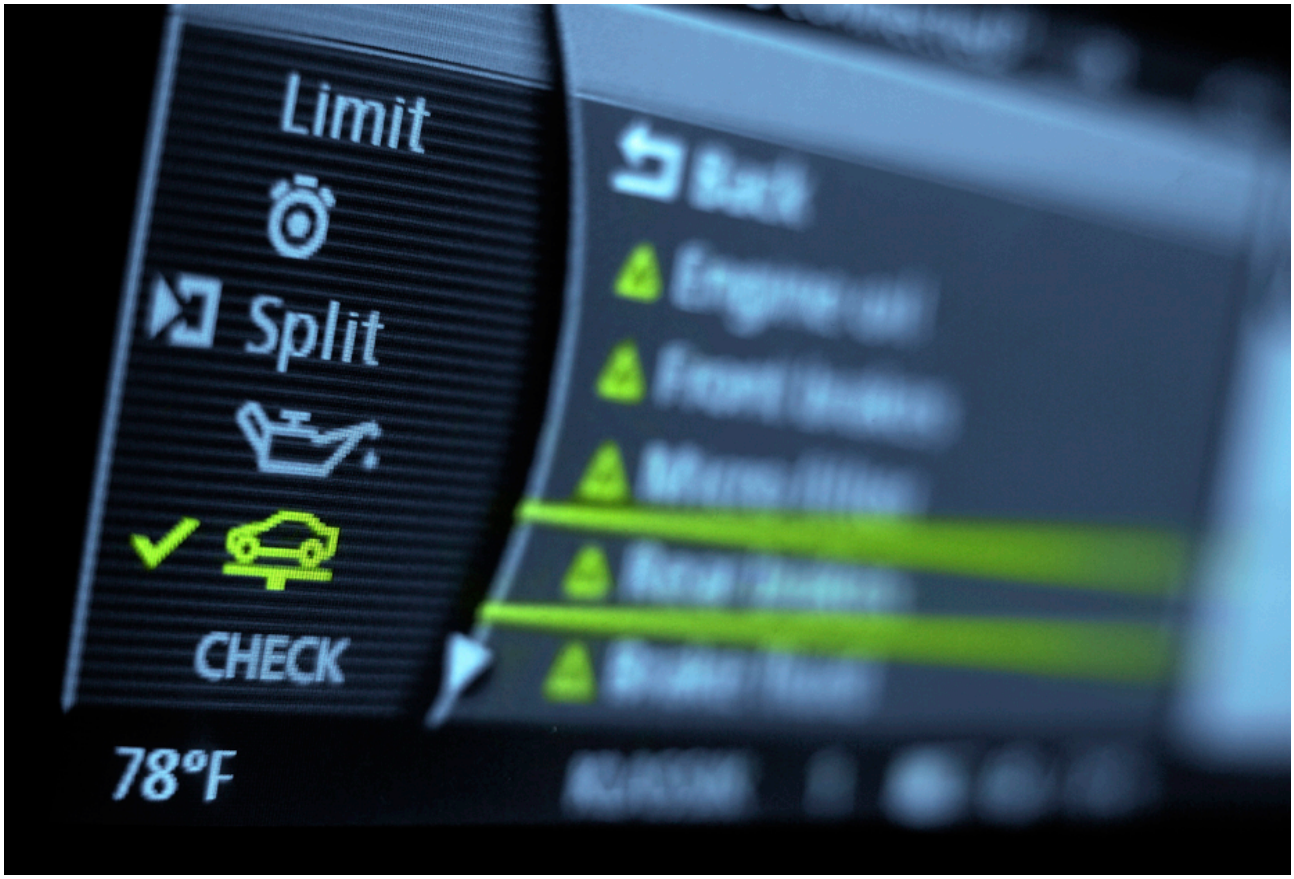
Elektrobit

EB tresos® AutoCore Generic 8 Memory Stack documentation

release notes update for the NvM module

product release 8.8.7



Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

https://www.elektrobit.com/support

Legal disclaimer

Confidential information.

Table of Contents

1. Overview

This document provides you with the release notes to accompany an update to the NvM module. Refer to the changelog Section 2.1, “Change log” for details of changes made for this update.

Release notes details

- EB tresos AutoCore release version: 8.8.7
- EB tresos Studio release version: 29.2.0
- AUTOSAR R4.0 Rev 3
- Build number: B577598

2. NvM module release notes

- AUTOSAR R4.0 Rev 3
- AUTOSAR SWS document version: 3.2.0
- Module version: 6.17.29.B577598
- Supplier: Elektrobit Automotive GmbH

2.1. Change log

This chapter lists the changes between different versions.

Module version 6.17.29

2022-11-14

- New feature: SingleBlockCallback ASR2011 compatibility.

Module version 6.17.28

2022-10-26

- ASCNVM-1502 Fixed known issue : NVM_REQ_RESTORED_FROM_ROM not defined by interface.arxml.

Module version 6.17.27

2022-07-25

- Internal module improvement. This module version update does not affect module functionality.
- New feature: InitBlockCallback ASR2011 compatibility.

Module version 6.17.26

2022-07-04

- New feature: MemAcc module support according to ASR R21-11.
- Internal module improvement. This module version update does not affect module functionality.

- ASCNVM-1499 Fixed known issue : Linking error due to wrong initialized variable allocated in memory section Cleared.
- Internal module improvement. This module version update does not affect module functionality.

Module version 6.17.25

2022-03-09

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.17.24

2021-10-08

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.17.22

2021-06-25

- ASCNVM-1438 Fixed known issue : NvM native block reference to not existing lower layer block
- ASCNVM-1427 Fixed known issue : Erroneous handling of redundant blocks configured with a crypto hook
- Internal module improvement. This module version update does not affect module functionality.

Module version 6.17.21

2021-04-26

- ASCNVM-1418 Fixed known issue : The vendor-specific PreWrite verification causes that a redundant NVRAM block is not written
- ASCNVM-1419 Feature Improvement : Return Pending for multicore blocks is not written
- Updated default values restoring according to ASR R20-11

Module version 6.17.20

2021-03-05

- New feature : Change behavior for PowerOnReset handling of NvM admin header
- Internal module improvement. This module version update does not affect module functionality.

- ASCNVM-1410 Fixed known issue : The NvM generator does not generate correct NVM_ASSERT_STC statements if array is used
- Internal module improvement. This module version update does not affect module functionality.
- ASCNVM-1414 Fixed known issue : NvM does not handle redundant blocks correctly during WriteAll

Module version 6.17.18

2020-10-23

- Internal module improvement. This module version update does not affect module functionality.
- New feature : Data recovery for Blocks with reconfigured length

Module version 6.17.17

2020-08-07

- Internal module improvement. This module version update does not affect module functionality.
- ASCNVM-1395 Fixed known issue : A deadlock occurs if a job for a block with an ID greater than 255 is canceled
- Internal module improvement. This module version update does not affect module functionality.
- Internal module improvement. This module version update does not affect module functionality.

Module version 6.17.15

2020-06-19

- Internal module improvement. This module version update does not affect module functionality.
- Added New Crypto Hooks functionality.
- ASCNVM-1388 Fixed known issue : Queuing mechanism does not work for queues which are greater than 256
- Added New API 'NvM_DisableBlockCheckMechanism' .
- Internal module improvement. This module version update does not affect module functionality.

Module version 6.17.12

2020-04-24

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.17.11

2020-02-21

- ASCNVM-1367 Fixed known issue : NvM doesn't compile on 64 Bits platforms ASCNVM-1370 Fixed known issue : NvM_FirstInitAll does not set a final job result for Invalidated blocks. Internal module improvement. This module version update does not affect module functionality.

Module version 6.17.10

2019-10-11

- Asynchronous CRC calculation queue in FIFO order implemented
- CancelWriteAll operation changed. Pending block stopped if Fls operation has not started.
- Internal module improvement. This module version update does not affect module functionality.

Module version 6.17.9

2019-06-14

- ASCNVM-1343 Fixed known issue : References to callback functions NvMReadRamBlockFromNvCallback and NvMWriteRamBlockToNvCallback are generated when NvMBlockUseSyncMechanism is set to false
- Update 'WriteBlockOnce' feature acording to ASR 4.3.1 .
- Internal module improvement. This module version update does not affect module functionality

Module version 6.17.8

2019-04-04

- Internal module improvement. This module version update does not affect module functionality
- ASCNVM-1332 Fixed known issue : NvM API NvM_ASR40_SetDataIndex fails to compile on a particular configuration.

Module version 6.17.7

2019-02-15

- Internal module improvement. This module version update does not affect module functionality

Module version 6.17.6

2019-01-23

- Added feature: Update of NvM queue sizes
- Internal module improvement. This module version update does not affect module functionality.
- Usage of non continuous NvBlock identifier (redir/search)
- First initialization of NV blocks.
- Added user callout function for passed production errors.
- ASCNVM-1310 Fixed known issue : NvM can get stuck in an infinite loop during BlockCheck mechanism.

Module version 6.17.5

2018-10-25

- Internal module improvement. This module version update does not affect module functionality
- ASCNVM-1279 Fixed known issue: Variables NvM_CalcCrc_CalcBuffer and NvM_ConfigurationId are mapped to a wrong section.
- ASCNVM-1285 Fixed known issue: AutoRepair block mechanism doesn't repair NV Blocks correctly.
- ASCNVM-1299 Fixed known issue: Block Specific Features 'AutoValidation' and 'CRCComparison' cannot work together

Module version 6.17.4

2018-05-25

Module version 6.17.3

2018-03-09

- ASCNVM-1263 Fixed known issue: AutorepairRAM for feature BlockCheck doesn't work as expected.

Module version 6.17.2

2018-02-16

- Added alternative for detecting software change (NvMSoftwareChangeCallout). Block1 that stores ConfigurationId has configurable size now. ConfigurationId range changed.

- Added user callout function for production errors.
- Added Background Block Check functionality.

Module version 6.17.1

2017-12-15

- NvM.xdm updated. Other VSMD violations are covered by deviations.
- ASCNVM-1202 Fixed known issue: Block size check is not working when multiple underlying modules of the same type are used
- Internal module improvement. This module version update does not affect module functionality
- Internal module improvement. This module version update does not affect module functionality
- Internal module improvement. This module version update does not affect module functionality
- Internal module improvement. This module version update does not affect module functionality
- Add feature for 'CRC Comparison mechanism before write'. Add feature for 'Data Comparison mechanism before write'.
- Multicore extension for NvM. Add multicore extension for NvM_CancelJobs API.

Module version 6.17.0

2017-09-22

- Added PRAM APIs for Nvm block Read/Write/RestoreBlockDefaults operations
- Updated job handling when internal operations are ongoing.
- NvM reporting of DET errors changed to be in sync with ASR 4.2.2.
- Internal module improvement. This module version update does not affect module functionality

Module version 6.16.0

2017-03-31

- Internal module improvement. This module version update does not affect module functionality

Module version 6.15.0

2016-11-04

- ASCNVM-1136 Fixed known issue: Multi-block operations do not process VALID blocks if NvM_SetRamBlockStatus() API is disabled
- ASCNVM-1141 Fixed known issue: NvM could discard queued immediate requests when other immediate requests are cancelled
- ASCNVM-1143 Fixed known issue: NvM generates wrong service interface for ReadBlock and RestoreBlockDefaults when NvMEnableASR32ServiceAPI is enabled and generation mode is different than generate_asr32_swcd
- ASCNVM-1155 Fixed known issue: Single block callback is not called by NvM for cancelled requests done from another single block callback
- ASCNVM-1156 Fixed known issue: NvM_CancelWriteAll request could be lost if it is triggered while NvM_WriteAll is currently being processed
- ASCNVM-1157 Fixed known issue: Blocks configured with Ram Block Crc could unnecessarily be read from Nv memory after a SW reset
- ASCNVM-1158 Fixed known issue: NvM could ignore asynchronous single block jobs requested during startup or shutdown
- ASCNVM-1155 Fixed known issue: Single block callback is not called by NvM for cancelled requests done from another single block callback

Module version 6.14.0

2016-05-25

- ASCNVM-1088 Fixed known issue: NvM_RestoreBlockDefaults fails for blocks configured with explicit synchronization and initialization callback
- ASCNVM-1079 Fixed known issue: Single block requests do not use the provided temporary RAM for blocks configured with explicit synchronization
- ASCNVM-1093 Fixed known issue: Multi-block notification is not called by NvM if the multi-block request is interrupted by an immediate request
- ASCNVM-1095 Fixed known issue: The result of NvM_RestoreBlockDefaults is wrongly set to NVM_REQ_OK if the CRC cannot be calculated in one step
- ASCNVM-1099 Fixed known issue: NvM does not calculate CRC again if static block Id failed and retry is performed
- Changed processing of mirror retry mechanism according to AUTOSAR4.2
- Updated mechanism of reporting production errors for read requests
- ASCNVM-1106 Fixed known issue: NvM could perform less read/write retries than configured for a redundant block

- ASCNVM-1107 Fixed known issue:The result of NvM_ReadAll is incorrectly set to NVM_REQ_NOT_OK if blocks are read as INVALID
- Added vendor specific configuration parameter for configuring returned value by the lower layer for blocks not found in the memory
- ASCNVM-1110 Fixed known issue: The short name of the operation EraseNvBlock is wrongly exported by NvM
- ASCNVM-1116 Fixed known issue: NvM can write wrong data while restoring redundancy of a redundant block when lower layer becomes BUSY or BUSY_INTERNAL
- ASCNVM-1117 Fixed known issue: The result of a user single block request could be wrongly changed from NVM_REQ_PENDING during interruptions of a multi-block requests by immediate block user request
- Added support for Debug & Trace with custom header file configurable via parameter `BaseDbgHeaderFile`
- ASCNVM-1113 Fixed known issue: Unnecessary warning could be reported by NvM if block size check is enabled
- Implemented NvM_ValidateAll() functionality according to AUTOSAR 4.2 software specifications
- ASCNVM-1121 Fixed known issue: Multiple calls of NvM_CancelJobs for the same block could lead to a wrong NVM_REQ_CANCELED result for the first processed job
- ASCNVM-1122 Fixed known issue: During read and restore requests, CRC is copied to a temporary RAM for blocks with NvMUserProvidesSpaceForBlockAndCrc
- ASCNVM-1119 Fixed known issue: The returned value of NvM_CancelJobs is E_OK even if the single block request cannot be cancelled
- Changed old memory sections used in NvM

Module version 6.13.0

2016-02-05

- Internal module improvement. This module version update does not affect module functionality
- ASCNVM-1061 Fixed known issue: Corrected the error messages issued by NvM when the configuration of block 1 is invalid
- Added a block-specific configuration parameter which allows the user to select the AUTOSAR version of the R-Port interfaces called by NvM
- ASCNVM-1066 Fixed known issue: NvM processes cancelled single-block requests
- ASCNVM-1071 Fixed known issue: Unexpected behavior could be observed when cancelling immediate requests
- ASCNVM-1081 Fixed known issue: Restore defaults requests do not copy CRC to the RAM block

- ASCNVM-1080 Fixed known issue: Write protection is wrongly enabled during read requests of blocks configured with NvMWriteBlockOnce
- ASCNVM-1077 Fixed known issue: Invalid DataIndex is accepted by NvM_SetDataIndex
- ASCNVM-1078 Fixed known issue: If configuration ID mismatch occurs, NvM does not process the extended runtime preparation correctly

Module version 6.12.0

2015-11-20

- Exported NvM service IDs to NvM users
- Implemented a new vendor specific configuration parameter NvMExportBlockLengths, which enables or disables the exporting of NVRAM block lengths to NvM users
- Added consistency checks regarding the pre-configuration of Configuration ID block
- ASCNVM-1049 Fixed known issue: NvM restores default data in case of invalidated blocks
- Improved the NvMExtraBlockChecks functionality regarding blocks with NvMUserProvidesSpaceForBlockAndCrc configuration parameter enabled. If an old NvM configuration is imported, after updating to this module version please manually change the NvMRamBlockDataAddress configuration parameter of the Configuration ID block from &NvM_ConfigurationId[0] to &NvM_ConfigurationId using a text editor.
- ASCNVM-1054 Fixed known issue: The size of the permanent RAM block configured for block 1 is not correctly calculated
- ASCNVM-1057 Fixed known issue: NvM_EraseNvBlock() shall not depend on job prioritization parameter

Module version 6.11.0

2015-06-19

- ASCNVM-1014 Fixed known issue: Data is copied to RAM during NvM_ReadBlock even if the NV block is invalid or inconsistent for blocks of type NVM_BLOCK_DATASET
- Improved the warning messages issued by the NvMExtraBlockChecks feature
- The queue length generated by RTE for NvM user is not correct
- Blocks configured with explicit synchronization have to be processed during multi-block operations
- Improved the handling of redundant NVRAM blocks
- ASCNVM-1033 Fixed known issue: CRC is not copied to user RAM during read requests

Module version 6.10.0

2015-02-25

- Implemented AUTOSAR Bugzilla RfC #62202
- ASCNVM-998 Fixed known issue: NvMSingleBlockCallback is not called when a single block request is cancelled
- ASCNVM-1009 Fixed known issue: Multi-block requests do not get preempted by an immediate block write request
- Added AUTOSAR 4.2.1 interface wrappers
- Implemented AUTOSAR Bugzilla RfC #52034
- ASCNVM-1017 Fixed known issue: Cancelled single-block requests are removed from the queues only after all other requests have been completely processed

Module version 6.9.0

2015-01-07

- ASCNVM-982 Fixed known issue: NvM_CancelWriteAll() cancels running NvM_WriteAll() as well as NvM_ReadAll() requests
- ASCNVM-988 Fixed known issue: User-defined multi-block callback function is not always called if NvM_WriteAll is queued and NvM_CancelWriteAll requested
- ASCNVM-987 Fixed known issue: Write verification size is not correctly calculated
- ASCNVM-994 Fixed known issue: After a NvM_CancelWriteAll request, the BswM_NvM_CurrentJobMode notification to BswM is not called when expected
- ASCNVM-996 Fixed known issue: NvMRepeatMirrorOperations configured to "0" may lead to an endless loop

Module version 6.8.0

2014-10-02

- ASCNVM-947 Fixed known issue: `NvM_ReadAll` for blocks of management type `NVM_BLOCK_REDUNDANT` does not read the second block and uses default data
- ASCNVM-948 Fixed known issue: `NvM_NvToRamCopyCallbackType` and `NvM_RamToNvCopyCallbackType` are referenced, but never defined
- ASCNVM-955 Fixed known issue: `NvM_ReadAll()` causes RAM data corruption for blocks with explicit synchronization enabled and permanent RAM configured

- ASCNVM-977 Fixed known issue: Data in permanent RAM might be incorrect for handling of improved redundancy
- Updated module version for ACG-7.2.0 release

Module version 6.7.0

2014-04-25

- Added support for function tracing and variable debugging via AUTOSAR Debugging
- Removed unnecessarily generated empty lines from the generated file
- ASCNVM-897 Fixed known issue: The result of blocks is updated during a read multi-block operation, even if they are not selected to be processed by `NvM_ReadAll`
- ASCNVM-903 Fixed known issue: `NvM_ReadAll()` returns `NVM_REQ_NOT_OK` if reading the configuration ID block (block 1) returns `NVM_REQ_REDUNDANCY_FAILED`
- ASCNVM-914 Fixed known issue: Conflicting attributes of `NvM_MainFunction` definition determines a Rte verification error
- ASCNVM-918 Fixed known issue: After executing `NvM_ReadAll()`, the result of block 0 (multi-block request) is set to `NVM_REQ_OK`, but the result of one block selected to be processed during `NvM_ReadAll()` may remain pending
- ASCNVM-921 Fixed known issue: Fixed values for `NvMRomBlockDataAddress` and `NvMRamBlockDataAddress` attributes of block 1 are not enforced by XDM check

Module version 6.6.0

2013-09-20

- ASCNVM-851 Fixed known issue: After a redundant block invalidation, the job result is `NVM_REQ_NOT_OK`
- ASCNVM-860 Fixed known issue: After a request to erase a redundant block, the job result is `NVM_REQ_NOT_OK`
- ASCNVM-867 Fixed known issue: Preprocessor error generated by missing memory sections in the NvM BSW module description
- ASCNVM-874 Fixed known issue: A compiler error may be reported due to missing memory sections in `NvM_ASR40_Bswmd.arxml`
- ASCNVM-825 Fixed known issue: Uninitialized variable `NvM_GlobalGenericStatus` can cause unexpected behavior in NvM module
- Added description of production errors to the module reference documentation
- ASCNVM-877 Fixed known issue: The job result `NVM_REQ_REDUNDANCY_FAILED` is still returned after a corrupted redundant block has been successfully written to NV memory with new data

Module version 6.5.0

2013-06-25

- ASCNVM-789 Fixed known issue: `NvM_SetRamBlockStatus()` is not synchronous
- ASCNVM-823 Fixed known issue: Writing a block fails when the Ea or Fee is busy with internal operations
- ASCNVM-829 Fixed known issue: An overflow of the array `NvM_Queue_Immediate[]` is not handled while `NvMJobPrioritization` and `NvMBlockUseSyncMechanism` are enabled

Module version 6.4.0

2013-02-08

- ASCNVM-717 Fixed known issue: The CRC calculation type CRC8 is not supported by the NvM module
- Implemented possible errors for the client server operations `NvMNotifyJobFinished` and `NvMNotifyInitBlock`
- Changed SWCD internal so that the Rte interface of `NvM_MainFunction` is generated without ASR32/ASR40 infixes
- ASCNVM-776 Fixed known issue: If no `NvMSingleBlockCallback` is configured, the job finished callbacks configured via Rte are not called

Module version 6.3.0

2012-10-19

- ASCNVM-701 Fixed known issue: If no `NvMInitBlockCallback` is configured, the initialization callbacks configured via Rte are not called
- Changed to restrict the use of configuration parameter `NvMUserProvidesSpaceForBlockAndCrc` for permanent RAM blocks only
- ASCNVM-716 Fixed known issue: Error message may occur with correct `NvMBlockJobPriority` and `Ea/FeeImmediateData` parameter configuration
- Changed the top-level structure of the software-component description in the arxml files from `/AUTOSAR/NvM` to `/AUTOSAR_NvM`
- Added AUTOSAR 3.2 support of Rte Interface and SWCD

Module version 6.2.0

2012-06-15

- Modified behavior of NvM when a block is locked and added new development error `NVM_E_BLOCK_LOCKED`
- Updated input parameters in `NvM_WriteBlock()`, C-interface and port interface
- ASCNVM-660 Fixed known issue: The multiplicity of some NvM parameters do not conform to the AUTOSAR 4.0 specification
- Changed the NvM layout and grouped configuration parameters in tabs
- ASCNVM-674 Fixed known issue: A warning or an Rte generation error may occur because the package `SwcBswMappings` is located at the wrong position in the Basic Software Module Description
- ASCNVM-687 Fixed known issue: Rte generation fails if API configuration class is 1 and Rte service port is true
- Added NvM BswM interaction feature

Module version 6.1.0

2012-03-16

- Changed implementation of `NvM_ReadBlock()` and `NvM_RestoreBlockDefaults()` to set the job result to `NVM_REQ_RESTORED_FROM_ROM` also
- Changed implementation of `NvM_WriteAll()` to process blocks with configuration parameter `NvMSelectBlockForWriteAll` set to true
- Changed the Det error in `NvM_GetVersionInfo` from `NVM_E_PARAM_DATA` to `NVM_E_PARAM_POINTER`
- Updated `NvM_SetDataIndex` to return `E_OK` in production mode for native and redundant blocks
- Added support for production error `NVM_E_QUEUE_OVERFLOW`
- Changed implementation of `NvM_ReadBlock()` to handle redundant blocks to recover failed NV block data from the uncorrupt block
- Updated `NvM_WriteAll` to write block 1 as the last block
- Added support for production error `NVM_E_WRITE_PROTECTED`
- Updated immediate request handling, ongoing `NvM_WriteAll` shall not be interrupted
- Added `EcuM_CB_NfyNvMJobEnd` as default configuration value of `NvmMultiBlockCallback`
- Added BSWMD for NvM
- ASCNVM-625 Fixed known issue: The production error symbolic names for the Dem events are defined twice and lead to compilation errors
- Improved redundant block handling
- ASCNVM-616 Fixed known issue: Write-once protection without effect

- ASCNVM-644 Fixed known issue: The explicit synchronization between application and NVRAM manager via ports is not supported

Module version 6.0.0

2011-09-30

- Initial AUTOSAR 4.0 version

2.2. New features

- ASR20-11 callbacks : Support for new ASR20-11 of NvM_SingleBlockCallbackFunction prototype.

  Description: NvM_SingleBlockCallbackFunction prototype based on the ASR20-11 version is available through parameter ASR2011CallbackEnabled.

2.3. Elektrobit-specific enhancements

This chapter lists the enhancements provided by the module.

- New mechanism : Cryptography Hooks for Read/Write NvM Blocks

  Description: A new mechanism in which blocks can be written as encrypted into NV and can be decrypted immediatly after Read.

  CryptoHooks General Configuration:

  New Parameter General : 'NvMEnableCryptoSecurityHooks

  Description : Enables the ussage of crypto security hooks for handling of NvM Blocks's data.

  New Parameter General : 'NvMCryptoReadHook

  Description : Configures the callback API to apply the corresponding crypto algorithm for a NvM Block after Read.

  New Parameter General : 'NvMCryptoWriteHook

  Description : Configures the callback API to apply the corresponding crypto algorithm for a NvM Block before Write.

  Each Block can be selected to be processed by the Read/Write Hooks :

New Parameter per NvM Block : 'NvMEnableBlockCryptoSecurityHandling

Description : Enables crypto handling of NvM user data corresponding to this NvM Block.

New Parameter per NvM Block : 'NvMCryptoExtraInfoSize

Description : The paramter shall be considered when calculating the actual block size when calling lower layers. This information can be either a MAC, a Hash or a Crypto Initialization Vector plus a MAC.

- New API : NvM_DisableBlockCheckMechanism

Description: Enables/Disables the 'BlockCheck' mechanism.

Description:

When called with 'TRUE' the mechanism will be disabled.

When called with 'FALSE' the mechanism will be enabled.

As for default the NvM starts with the mechanism enabled.

- Data Comparison Mechanism

Description:

The vendor-specific parameter `NvMPreWriteDataComp` enables the use of the data comparison mechanism before a block is written.

With the `NvMPreWriteDataComp` enabled, when processing a Write Job, triggered from any NvM APIs(WriteBlock, WritePRAMBlock, WriteAll) the Block Reads from NV into an internal buffer and a comparison will be done with the data stored currently in RAM that is to be written. If the data is equal then the Write Job will be finished without actual writing but with Job_OK, but if the data is different, then the Write Job will proceed as normal.

Rationale:

This feature is useful to save EEPROM/FEE Flash from wear out. Also it improves runtime in configuration that might trigger NV Write jobs more often than the data is actually changed.

- Cancel Internal Operations of underlying modules

Description:

When processing a job with immediate priority the vendor-specific parameter `NvMCancelInternalOperations` enables the checking of status of all configured memory devices and a cancel request, `MemIf_Cancel`, will be issued for all devices with status different then `MEMIF_IDLE`.

Rationale:

This feature is needed to avoid concurrent access of two underlying modules to the same HW driver. For example an immediate job assigned to module FEE_1 is pending to be executed but we have an additional underlying module, FEE_2 which is already executing a job on the same HW driver. To be sure the immediate job is executed immediate we need to cancel the internal handling of FEE_2 also.

- Hook feature

  Description:

  The vendor-specific parameter `NvMReadBlockHook` enables the use of a hook function for `NvM_ReadBlock()/NvM_ReadAll()`. The vendor-specific parameter `NvMWriteBlockHook` enables the use of a hook function for `NvM_WriteBlock()/NvM_WriteAll()`.

  The hook feature is supported only for blocks configured with a permanent RAM.

  If hook features are enabled, only NV block data is updated by users. Read hook function is called after reading NV data and write hook function is called before writing NV data.

  Rationale:

  Hook features are useful for application that needs encrypt/decrypt data.

- Extra block size checks

  Description:

  The vendor-specific parameter `NvMExtraBlockChecks` enables the compiler to check if the configured block lengths (including CRC if enabled by configuration) match the size of the RAM or ROM variables associated to the blocks if a size can be determined by the `sizeof()` C-function. The check is done at compile time and no additional resources are required.

  Rationale:

  Wrongly configured block sizes lead to hard-to-find problems during development. These checks help to find these misconfigurations.

- Enhanced production error reporting

  Description:

  An enhanced production error reporting mechanism has been introduced. This allows to configure the following options independently for each Dem event:

  - Report production errors to the *Diagnostics Event Manager* (Dem).
  - Report production errors to the *Development Error Tracer* (Det) as development errors.
  - Do not report production errors at all.

  If a production error is redirected towards the Det, you may configure the reported Det error-ID.

Rationale:

This enhancement implements the HIS requirements concerning fault operation and error detection: HisGen0007, HisGen0008 and HisGen0009.

- Optimized RAM consumption

  Description:

  A new vendor-specific parameter `NvmUserProvidesSpaceForBlockAndCrc` has been introduced to configure where the CRC is stored for each block in case of a permanent RAM block.

  If set to true, NvM stores the CRC in the RAM block itself. In this case, the user of the block reserves memory directly behind the RAM block to be used by the NvM for storing CRC.

  If set to false, NvM uses an internal buffer to store the block data and CRC before writing or reading from the Nv memory.

  Rationale:

  The data and CRC are stored together as single block in the underlying memory abstraction modules (Fee or Ea). Therefore, NvM needs to copy data and CRC to an internal buffer before writing to NV memory. The size of this internal buffer depends on the length of the largest block configured in NvM. Therefore, enabling `NvmUserProvidesSpaceForBlockAndCrc` for large blocks saves RAM because the size of the internal buffer can be reduced to fit the smaller blocks. It also reduces runtime by not having to copy to and from the internal buffer.

- Unified RAM mirror and internal buffer

  Description:

  In order to reduce the RAM usage of the NvM module, the RAM Mirror (defined by AUTOSAR NvM SWS) and NvM's internal buffer share the same RAM space.

  Rationale:

  As stated in the NvM SWS, the RAM Mirror size shall be equal to the size of the largest NVRAM block configured with explicit synchronization enabled. By using the same memory location for the two buffers, the resource consumption of the NvM module is reduced.

  Note:

  If *Write Verification* or *Data Comparison Mechanism* is enabled for at least one NVRAM block, an additional RAM buffer (*Write Verification Buffer*) is allocated. The size of this buffer is equal to the maximum value of the `NVM_WRITE_VERIFICATION_DATA_SIZE` parameter.

- Enhanced the *Advanced Recovery* feature

Description:

The vendor-specific general parameter NvmAdvancedRecovery has been made block-specific so that it can be configured differently for each block.

If NvmAdvancedRecovery is set to true for a block, NvM ensures that irrespective of write-protection, data recovered from the default data of the configured ROM block or user-provided data (NvmInitBlockCallback) during a read operation is written to NV memory during the execution of NvM_WriteAll.

If NvmAdvancedRecovery is set to false for a block, NvM does not automatically write the recovered data to NV memory if the block is write-protected or data has been recovered from the redundant NV block.

Rationale:

Implicit and explicit error recovery, as explained in the SWS, does not apply to blocks which are write-protected. Hence for such blocks, recovered data is not written to NV memory resulting in data loss.

Therefore, enabling NvmAdvancedRecovery does recover the block data when implicit error recovery is not possible.

- Improved handling of redundant blocks

Description:

The handling of redundant NVRAM blocks during a read operation has been improved to read both blocks associated with a redundant block. If one copy has been read successfully and the other copy is detected as corrupted, the job result is set to NVM_REQ_OK or NVM_REQ_REDUNDANCY_FAILED, depending on the value of the configuration parameter NvMRedundantRecovery.

If both copies of a redundant block have been read successfully, additionally the CRCs of both copies are compared. If this comparison fails, the block result is set to NVM_REQ_OK or NVM_REQ_REDUNDANCY_FAILED depending on the value of the configuration parameter NvMRedundantRecovery.

During the processing of NvM_WriteBlock, both NV blocks of a redundant block are always written. During the processing ofNvM_WriteAll, both NV blocks of a redundant block are written only if the RAM block is marked as *VALID* and *CHANGED*.

If only one NV block of a NVRAM Block of the type NVM_BLOCK_REDUNDANT has been previously read successfully, no error is reported to DEM and the redundancy will be restored during the next NvM_WriteBlock or NvM_WriteAll request. If the block is marked as changed or NvM_WriteBlock is currently being processed, both copies are written with data from RAM. If the block is not marked as changed the block is automatically repaired with data from the correct NV block.

- Configuration parameter NvMMainFunctionCycleTime

Description:

The vendor-specific configuration parameter `NvMMainFunctionCycleTime` is implemented to specify the period with which `NvM_MainFunction` shall be invoked by the BSW scheduler.

Rationale:

The BSW scheduler of the Rte requires the *PERIOD* attribute of a timing event which triggers a schedulable entity in the Basic Software Module description. The Rte calculates the activation time of the OS schedule table and the execution time of a BSW main function within an OS task.

- Configuration parameter `NvMUserHeader`

  Description:

  The vendor-specific configuration parameter `NvMUserHeader` is implemented to specify user-specific header files for providing user dependent definitions and declaration (either directly or indirectly).

  Rationale:

  The user has to define and declare symbols (RAM and ROM blocks, CallBacks) based on the requirement. The list `NvMUserHeader` allows users to include the header files containing user defined symbols.

- Function tracing support via AUTOSAR Debugging

  Description:

  The module NvM supports tracing of function entry and exit via the EB Dbg module.

  Function tracing records following parameters for each function:

  - function name
  - values of the function arguments
  - point in time of function invocation
  - point in time of function termination
  - return value of the function

- Variable Debugging

  Description:

  The NvM module supports debugging of variables that contain module's state, job results and status.

  Variables available for debugging:

  - `NvM_GlobalWorkingStatus` (16 bits bitfield):
    - bit 0: RAM data has been changed
    - bit 1: block write protected
    - bit 2: block is invalidated

- bit 3: block is locked
- bit 4: Rom data was loaded
- bit 5: permanent write protection
- bit 6: block address changed
- bit 7: no loss of redundancy
- bit 8: CRC mismatch
- bit 9: temporary RAM block used
- bit 10: block inconsistent
- bit 11: static block ID check failed
- bit 12: NvM mirror used
- bit 13: `SetRamBlockStatusChangedFlag`

- `NvM_GlobalErrorStatus` (8 bits bitfield equivalent with `NvM_RequestResultType`)
- `NvM_GlobalGenericStatus` (8 bits bitfield):
  - bit 0: Initialization flag
  - bit 1: Cancel Write All flag
  - bit 2: Main Function Processing flag
  - bit 3: Polling Mode Ignore flag
  - bit 4: Dynamic Cofiguration flag
  - bit 5: Multi Request Keep Job Pending flag
  - bit 6: Block request Canceled Flag
- `NvM_GlobalCallLevel` (of type `uint8`: store the call level of the currently processed state machine)

- Export block lengths

  Description:

  The vendor-specific parameter `NvMExportBlockLengths` enables or disables the exporting of lengths for all configured NVRAM blocks. The exported lengths correspond to the value of the `NvMNvBlockLength` configuration parameter of each block.

  Rationale:

  The configured block lengths represent internal information of the NvM module. By enabling this configuration parameter, NvM users can avoid duplicating information regarding the lengths of the NVRAM blocks.

- New mechanism : Data recovery for Blocks with reconfigured length

Description: A new mechanism in which blocks can be read from the NV memory even if their configured length has changed,

Description : When after a Read the lower layer returns the job result as MEMIF_JOB_OK_SIZE_DECREASED or MEMIF_JOB_OK_SIZE_INCREASED the NvM shall be able to forward the truncated data as if the Block was read successful. If Crc is configured for a block the NvM shall consider it passed and will not do it in this instance. The NvM shall report the the upper layer the job results NVM_REQ_OK_BLK_DECREASED or NVM_REQ_OK_BLK_INCREASED.

- ASR20-11 callbacks : Support for new ASR20-11 of NvM_InitBlockCallbackFunction prototype.

  Description: NvM_InitBlockCallbackFunction prototype based on the ASR20-11 version is available through parameter ASR2011CallbackEnabled.

2.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- Parameter `NvMDrvModeSwitch`multiplicity.

  Description:

  Multiplicity of NvMDrvModeSwitch shall be 0..1. Fast mode functionality is enabled by `NvMMemAccUsage` configuration set to 'false'

  Rationale:

  Retro-compatibility of NvM with direct acces to memory drivers pre-ASR 21-11.

  Requirements:

  NVM496_Conf

- Parameter `NvMBufferAlignmentValue`multiplicity.

  Description:

  Multiplicity of NvMBufferAlignmentValue shall be 0..1. Alignment functionality is enabled by `NvMMemAccUsage` configuration set to 'true'

  Rationale:

  Retro-compatibility of NvM with memory drivers pre-ASR 21-11.

  Requirements:

  ECUC_NvM_00573

- Parameter `NvMBufferAlignmentValue`shall have Type Enum instead of choice reference to [EaGeneral, FeeGeneral].

  Description:

  Name NvMBufferAlignmentValue Parent Container NvMCommon Description Parameter determines the alignment of the start address that NvM buffers need to have. Value shall be set according to the largest alignment of the used EaBufferAlignmentValue / FeeBufferAlignmentValue. Type Type Enumeration Range: NVM_ALIGN_8_BITS No alignment constraints NVM_ALIGN_16_BITS Internal buffers aligned to 16 bits NVM_ALIGN_32_BITS Internal buffers aligned to 32 bits NVM_ALIGN_64_BITS Internal buffers aligned to 64 bits Value Configuration Class Pre-compile time X All Variants Link time – Post-build time – Scope / Dependency scope: ECU

  Rationale:

  NvM could use multiple Fee or Ea modules

  Requirements:

  ECUC_NvM_00573

- Parameter `NvMBlockUseCRCCompMechanism` shall have a multiplicity of "0..1".

  Description:

  Parameter `NvMBlockUseCRCCompMechanism` shall have a multiplicity of "0..1" as opposed to a multiplicity of "1" as specified by NVRAM Manager Release 4.3.0 says.

  Rationale:

  Because the parameter is optional, it can also have a multiplicity of "0".

  Requirements:

  ECUC_NvM_00556

- Description:

  In contrast to the AUTOSAR NVRAM Manager specification 4.3.0, the Crc compare functionality shall be provided also to NVRAM blocks with parameter NvMCalcRamBlockCrc set to False.

  Rationale:

  The parameter NvMCalcRamBlockCrc has no influence over the CRC compare functionality.

  Requirements:

  NvM.Crc.Comparison_CalcRamBlock

- ~~DEM error `NVM_E_HARDWARE` reporting is not supported.~~

Description:

In contrast to the AUTOSAR NVRAM Manager specification, the NVM_E_HARDWARE production error is not reported by NvM.

Rationale:

For MEMIF_JOB_FAILED, MEMIF_BLOCK_INCONSISTENT or a CRC mismatch reported by the lower layer, NvM is reporting extended production errors according to AUTOSAR specification. From architecture point of view NvM cannot detect HW errors as it is HW independent.

Requirements:

SWS_NvM_00835

- Configuration variant `VARIANT-LINK-TIME` is not supported

Description:

In contrast to the requirement `NVM727`, the NvM module does not support the configuration variant `VARIANT-LINK-TIME`. The `VARIANT-LINK-TIME` is designed for the use cases where parameters are fixed at link time.

Requirements:

NVM727

- `NvMNvramDeviceId` maximum configurable value is limited to 7

Description:

In contrast to requirement NVM035_Conf, the NvM module supports only `NvMNvramDeviceId` values up to 7 instead of 254.

Requirements:

NVM035_Conf

- The `NvMWriteVerificationDataSize` maximum configurable value is 65535

Description:

In contrast to the requirement NVM538_Conf, the NvM module allows a maximum value of 65535 instead of 65536 for the `NvMWriteVerificationDataSize` parameter.

Rationale:

http://www.autosar.org/bugzilla/show_bug.cgi?id=47033 has been raised in AUTOSAR Bugzilla to fix the issue in NvM SWS. Since the maximum value of block length(NvMNvBlockLength) is 65535, write veri-

fication performs on block length. Thus the maximum configurable value of `NvMWriteVerification-DataSize` shall be 65535.

Requirements:

NVM538_Conf

- The `NvMNvramBlockIdentifier` minimum configurable value is 1 instead of 2

Description:

In contrast to the requirement NVM481_Conf, the NvM module allows a minimum value of 1 for the `NvM-NvramBlockIdentifier` parameter.

Requirements:

NVM481_Conf

- Operations in port interfaces for notifications

Description:

In contrast to the AUTOSAR specification, the operations of the client-server interfaces `NvMNotifyJobFinished` and `NvMNotifyInitBlock` specify the possible error `E_NOT_OK`. Therefore, the compatibility check in the Rte would fail if a port of the NvM refers to one of these interfaces that is connected to a port of a SwC that refers to a correct interface of the same name.

This problem does not occur in case the respective interfaces of the NvM are referred directly by the port declarations of the SwC.

Rationale:

The C-interfaces `NvM_SingleBlockCallbackFunction` and `InitBlockCallbackFunction` as specified in the AUTOSAR specification have `Std_ReturnType` as return type. To be able to link to the respective C-interfaces, the client-server interfaces `NvMNotifyJobFinished` and `NvMNotifyInitBlock` have been adapted by adding the possible error `E_NOT_OK`.

Requirements:

NVM735, NVM736

- Symbolic port name support

Description:

The port names provided by the NvM are not named by their numeric index of the blocks as suggested by the AUTOSAR NvM SWS. Instead the ports are postfixed by the symbolic name of the related configuration container.

Rationale:

With symbolic names, port names do not change when ports are deleted or inserted and renumbered. Therefore ports must not be re-connected.

- Production errors may be switched off

  Description:

  In contrast to the requirement NVM189, the reporting of production errors may be switched off by configuration.

  Rationale:

  See the rationale in the list of module enhancements related to *Enhanced production error reporting*.

  Requirements:

  NVM189

- `NvM_CancelWriteAll` (reference to product description: ASCPD-46)

  Description:

  In contrast to the AUTOSAR NVRAM Manager specification, the function `NvM_CancelWriteAll()` is implemented as a synchronous function instead of an asynchronous one.

  Rationale:

  Since `NvM_CancelWriteAll()` must not be queued as per requirement NVM420, `NvM_CancelWriteAll()` does not need to be an asynchronous function. This should be changed in the AUTOSAR NVRAM specification.

  Requirements:

  NVM458

- Polling mode (reference to product description: ASCPD-45)

  Description:

  The NvM and the memory stack modules Ea and Fee do not work reliably with interrupts calls made from the memory drivers Fls and Eep. The modules Ea, Fee, Fls, Eep must be configured so that Ea and Fee poll the underlaying Fls and Eep drivers. Additionally the NvM, Ea, and Fee main functions must be called from one task so that they cannot preempt each other.

  The configuration parameter `NvmPollingMode` is ignored. The callback functions `NvM_JobEndNotification()` and `NvM_JobErrorNotification()` can be used, although the configuration parameter `NvmPollingMode` is enabled. These functions are only linked to the user application if they are invoked.

  Requirements:

NVM441,NVM440,NVM501_Conf

- Parameter `NvMNvramBlockIdentifier` shall accept minimum value 1.

Description:

Parameter `NvMNvramBlockIdentifier` shall have the range: `min = 1 max = 2^(16- NVM_DATASET_SELECTION_BITS)-1` as Specification of NVRAM Manager Release 4.3.0 says.

Rationale:

The reserved block that stores the configuration identifier must have NvMNvramBlockIdentifier = 1, and it is preconfigured, therefore the lower boundery should be 1. The user can still only configure this parameter starting with value 2.

Requirements:

NVM481_Conf

- Parameter `NvMWriteVerificationDataSize` shall shall have multiplicity 0..1.

Description:

Parameter `NvMWriteVerificationDataSize` shall have multiplicity 0..1 because its availability is dependent on the value of parameter `NvMWriteVerification`.

Rationale:

If parameter `NvMWriteVerification` is configured TRUE, `NvMWriteVerificationDataSize` will be enabled. If parameter `NvMWriteVerification` is configured FALSE, `NvMWriteVerificationDataSize` will be disabled.

Requirements:

NvM.NvMWriteVerificationDataSize_Conf

- Vendor specific parameters shall not be added in alphabetical order.

Description:

Additional vendor specific parameter definitions (using ParameterTypes), container definitions and references shall NOT be added to the VSMD according to the alphabetical order.

Rationale:

The warning is known and accepted.

Requirements:

EcucSws_1014

- The parameter NvMCompiledConfigId size is extended to uint64 size.

  Description:

  The parameter NvMCompiledConfigId shall have the same size with the ConfigId read from NV memory. The ConfigId read from NV memory is stored in the reserved Block 1, which has configurable size. Therefore NvMCompiledConfigId must have a maximum range of 0..9223372036854775807.

  Rationale:

  Size of configId block shall be configurable (not limited to 2 bytes). Please see ticket ASCNVM-1212.

  Requirements:

  NVM492_Conf
- Configuration id's comparison shall be done very early at ReadAll.

  Description:

  The comparison between the stored configuration ID and the compiled configuration ID shall be done as the first step of the job of the function NvM_ReadAll during startup, after reading Block1.

  Rationale:

  Should be done very early at ReadAll process, but not before reading Block1 from NvRAM, otherwise the comparison can not take place without knowing the stored configuration id (Block1's content).

  Requirements:

  NvM073

2.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- Restriction on starting new jobs

  Description:

  A new asynchronous job request can be started only after the single-block callback notification for a previous request for the same block is received.

  Rationale:

A new job request would be accepted even before the callback for a previous job is issued. But in such cases, the receiver of the notification cannot know for which request the callback was invoked.

- Limitation on R-Ports called by NvM

  Description:

  NvM can only invoke one version of the generated R-Port interfaces. This limitation refers to the following interfaces:

  - NvMNotifyInitBlock
  - NvMNotifyJobFinished
  - ReadRamBlockFromNvm
  - WriteRamBlockToNvm

  Rationale:

  It is possible to enable the generation of multiple AUTOSAR version service APIs and call-backs by configuration. In this case, NvM can not determine during run-time which version of the service APIs was called for triggering requests, and thus it can not invoke the same version of the call-back functions. The NvM user has the possibility to specify the expected call-back functions version using the NvMRPortInterfacesASRVersion configuration parameter.

- Limitation on single block callbacks for cancelled requests

  Description:

  If a pending single-block request for an NVRAM block is cancelled and another request for the same block is queued before the next invokation of NvM_MainFunction(), the single-block callback for the former (cancelled) request will not be called.

  Rationale:

  This limitation allows for a more efficient implementation, as NvM does not have to store the history of cancelled requests for each NVMRAM block.

2.6. Open-source software

NvM does not use open-source software.