Document Title	SWS_NVRAMManager: Complete Change Documentation 4.3.0 - 4.3.1
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	695

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.1

Table of Contents

1 SWS_NVRAMManager

1.1 Specification Item ECUC_NvM_00072

Trace References:

none

Content:

Name	NvMWriteBlockOnceNvMBlockDescriptor.NvMWriteBlockOnce		
Description	Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time . 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 initializationor if the block was already written and it is detected as valid and consistent during a read for it. [NVM276]. true: Defines write protection after first write is enabled. false: Defines write protection after first write is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74058: [NvM] Write protection and erase requests for NvMWriteBlockOnce blocks

  Problem description:

  Name: Delia Batica
  Phone: +40 356 78 4202
  Role: Developer

  Description/Motivation:

  Currently, the behavior of NvM API's for written blocks with NvMWriteBlockOnce configured TRUE that have write protection unset is unclear.

  I will refer as wbo the blocks with NvMWriteBlockOnce configured TRUE.

How should the wbo be handled during read requests if the block is reported as inconsistent?
In SWS_NvM_00316 for eg., NvM_ReadBlock sets wbo to write protected if block is valid. (same for SWS_NvM_00314, SWS_NvM_00784).

How should NvM_EraseNvBlock and NvM_InvalidateNvBlock handle a wbo if the write protection is still unset? Should the 2 APIs set the write protection for a block configured with NvMWriteBlockOnce set to TRUE?

NVM072_Conf : says that NVRAM manager".. sets the
write protection bit after the NV block was written the first time. This means that some of the NV blocks in the NVRAM should never be erased ..after first initialization.

Also, it should be made clear that "write protection", "write protection bit", "write protection attribute" all refer to the current known state of write protection of a block.

Agreed solution:

(1) Change the Description for requirement ECUC_NvM_00072 to:
"Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block was already written and it is detected as valid and consistent during a read for it. [NVM276].
true: Defines write protection after first write is enabled.
false: Defines write protection after first write is disabled."

(2) Change the requirement SWS_NvM_00316 to:
"The job of the function NvM_ReadBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(3) Change the requirement SWS_NvM_00314 to:
"The job of the function NvM_ReadAll shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(4) Change the requirement SWS_NvM_00784 to:

"The job of the function NvM_ReadPRAMBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(5) Add a requirement, in chapter "7.2.2.13 NVRAM block write protection", stating the following:
"For a block configured with MVM_WRITE_BLOCK_ONCE (TRUE), NvM shall reject any Write/Erase/Invalidate request made prior to the first read request."

(6) For the above requirement, add the following Rationale:
"In case of a reset, the write protection flag of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE), from the NvM Administrative block, is cleared. In order to reactivate the protection, the block must be read prior to a first Write/Erase/Invalidate request being processed, in order to set the write proctection only for a block that is valid and consistent. The first read request can be done either as a single block request or as part of NvM_ReadAll."

(7) Add a requirement, in chapter "7.3.1 Development errors", stating the following:
"The development error NVM_E_WRITE_ONCE_STATUS_UNKNOWN (0x1A) shall be detectable by the NvM module when a Write/Erase/Invalidate is made for a block with MVM_WRITE_BLOCK_ONCE (TRUE), prior to the first read request made for that block, depending on whether the build version mode is development mode."

(8) Add the following requirements in chapter "7.4 Error Detection":
- If development error detection is enabled for NvM module, the function NvM_WriteBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the function NvM_WritePRAMBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_WriteAll shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when the processing of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_EraseNvBlock shall report the DET error

NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_InvalidateNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

(9) Add a requirement, in chapter "8.1.3.2.4 NvM_EraseNvBlock", after the 00423 requirement, which states the following:
"The job of the function NvM_EraseNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."

(10) Add a requirement, in chapter "8.1.3.2.6 NvM_InvalidateNvBlock", after the 00417 requirement, which states the following:
"The job of the function NvM_InvalidateNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."
–Last change on issue 74058 comment 15–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.2 Specification Item ECUC_NvM_00481

Trace References:

none

Content:

Name	NvMNvramBlockIdentifierNvMBlockDescriptor.NvMNvramBlockIdentifier
Description	Identification of a NVRAM block via a unique block identifier. Implementation Type: NvM_BlockIdType. min = 1 2 max = 2(̂16- NVM_DATASET_SELECTION_BITS)-1 Reserved NVRAM block IDs: 0 -> to derive multi block request results via NvM_GetErrorStatus 1 -> redundant NVRAM block which holds the configuration ID (generation tool should check that this block is correctly configured from type,CRC and size point of view)
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	1 2 .. 65535
Default value	–

Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local dependency: NVM_DATASET_SELECTION_BITS		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73816: Correction of the range of the configuration parameters in NvM

  Problem description:

  Autosar version 4.2.1.

  Collection of the relevant informations from NvM SWS:

  1.
  [SWS_NvM_00471] NvM_BlockIdType
  Range: 0..2^(16-NvMDatasetSelectionBits)-1

  Identification of a NVRAM block via a unique block identifier.

  Reserved NVRAM block IDs:
  0 -> to derive multi block request results via NvM_GetErrorStatus
  1 -> redundant NVRAM block which holds the configuration ID

  2.
  ECUC_NvM_00494: NvMDatasetSelectionBits
  Range 0 .. 8

  0: No dataset or redundant NVRAM blocks are configured at all, no
  selection bits required.
  1: In case of redundant NVRAM blocks are configured, but no dataset
  NVRAM blocks.

  3. NvMBlockDescriptor 1 .. 65536
  4. NvMNvBlockBaseNumber 1 .. 65534
  5. NvMNvramBlockIdentifier 1 .. 65535

  Please see the following statements:
  1. NvMDatasetSelectionBits cannot be 0, as Block 1 is redundant (which holds the configuration ID)
  2. Seeing the NvM_BlockIdType, the range of the NvMBlockDescriptor can be 1 ..

65535
3. The amount of NvMNvBlockBaseNumber is less than the amount of NvMBlockDescriptor.

Is this understanding correct?

Proposed solution (A):
1. Set NvMDatasetSelectionBits to "Range 1 .. 8", and
2. Set NvMBlockDescriptor and NvMNvramBlockIdentifier range to "1..65534"

Proposed solution (B):
1. Set NvMDatasetSelectionBits to "Range 1 .. 8", and
2. More limitation of NvMBlockDescriptor/NvMNvBlockBaseNumber/NvMNvramBlockIdentifier, as the range of NvMDatasetSelectionBits starts with 1.

Maybe other configuration parameters have to be corrected.
–Last change on issue 73816 comment 22–

Agreed solution:

(1) Extend the Description for requirement ECUC_NvM_00497: (related to NvMDynamicConfiguration) by adding: "This parameter affects all NvM processing related to Block with ID 1 and all processing related to Resistant to Changed Software. If the Dynamic Configuration is disabled, Block 1 cannot be used by NvM."

(2) For the NvMBlockDescriptor container (chapter 10.2.3), for the NvMNvramBlockIdentifier configuration parameter (requirement ECUC_NvM_00481), change range "min = 1" to "min = 2" .
(3) change in description ECUC_NvM_00481: min = 2 max = 2^(16-NVM_DATASET_SELECTION_BITS)-1
–Last change on issue 73816 comment 19–

BW-C-Level:

Application	Specification	Bus
4	4	1

1.3 Specification Item ECUC_NvM_00497

Trace References:

none

Content:

Name	NvMDynamicConfigurationNvMCommon.NvMDynamicConfiguration		
Description	Preprocessor switch to enable the dynamic configuration management handling by the NvM_ReadAll request. true: Dynamic configuration management handling enabled. false: Dynamic configuration management handling disabled. This parameter affects all NvM processing related to Block with ID 1 and all processing related to Resistant to Changed Software. If the Dynamic Configuration is disabled, Block 1 cannot be used by NvM.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73816: Correction of the range of the configuration parameters in NvM

  Problem description:

  Autosar version 4.2.1.

  Collection of the relevant informations from NvM SWS:

  1.
  [SWS_NvM_00471] NvM_BlockIdType
  Range: 0..2^(16-NvMDatasetSelectionBits)-1

  Identification of a NVRAM block via a unique block identifier.

  Reserved NVRAM block IDs:
  0 -> to derive multi block request results via NvM_GetErrorStatus
  1 -> redundant NVRAM block which holds the configuration ID

  2.
  ECUC_NvM_00494: NvMDatasetSelectionBits
  Range 0 .. 8

  0: No dataset or redundant NVRAM blocks are configured at all, no
  selection bits required.
  1: In case of redundant NVRAM blocks are configured, but no dataset

NVRAM blocks.

3. NvMBlockDescriptor 1 .. 65536
4. NvMNvBlockBaseNumber 1 .. 65534
5. NvMNvramBlockIdentifier 1 .. 65535

Please see the following statements:
1. NvMDatasetSelectionBits cannot be 0, as Block 1 is redundant (which holds the configuration ID)
2. Seeing the NvM_BlockIdType, the range of the NvMBlockDescriptor can be 1 .. 65535
3. The amount of NvMNvBlockBaseNumber is less than the amount of NvMBlockDescriptor.

Is this understanding correct?

Proposed solution (A):
1. Set NvMDatasetSelectionBits to "Range 1 .. 8", and
2. Set NvMBlockDescriptor and NvMNvramBlockIdentifier range to "1..65534"

Proposed solution (B):
1. Set NvMDatasetSelectionBits to "Range 1 .. 8", and
2. More limitation of NvMBlockDescriptor/NvMNvBlockBaseNumber/NvMNvramBlockIdentifier, as the range of NvMDatasetSelectionBits starts with 1.

Maybe other configuration parameters have to be corrected.
–Last change on issue 73816 comment 22–

Agreed solution:

(1) Extend the Description for requirement ECUC_NvM_00497: (related to NvMDynamicConfiguration) by adding: "This parameter affects all NvM processing related to Block with ID 1 and all processing related to Resistant to Changed Software. If the Dynamic Configuration is disabled, Block 1 cannot be used by NvM."

(2) For the NvMBlockDescriptor container (chapter 10.2.3), for the NvMNvramBlockIdentifier configuration parameter (requirement ECUC_NvM_00481), change range "min = 1" to "min = 2" .
(3) change in description ECUC_NvM_00481: min = 2 max = 2^(16-NVM_DATASET_SELECTION_BITS)-1
–Last change on issue 73816 comment 19–

BW-C-Level:

Application	Specification	Bus
4	4	1

1.4 Specification Item SWS_NvM_00314

Trace References:

none

Content:

The job of the function NvM_ReadAll shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) and that is not detected by underlying SW as being invalidated, shall be marked as write protected as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlock Protection.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74058: [NvM] Write protection and erase requests for NvMWriteBlockOnce blocks

  Problem description:

  Name: Delia Batica
  Phone: +40 356 78 4202
  Role: Developer

  Description/Motivation:

  Currently, the behavior of NvM API's for written blocks with NvMWriteBlockOnce configured TRUE that have write protection unset is unclear.

  I will refer as wbo the blocks with NvMWriteBlockOnce configured TRUE.

  How should the wbo be handled during read requests if the block is reported as inconsistent?
  In SWS_NvM_00316 for eg., NvM_ReadBlock sets wbo to write protected if block is valid. (same for SWS_NvM_00314, SWS_NvM_00784).

  How should NvM_EraseNvBlock and NvM_InvalidateNvBlock handle a wbo if the write protection is still unset? Should the 2 APIs set the write protection for a block configured with NvMWriteBlockOnce set to TRUE?

NVM072_Conf : says that NVRAM manager".. sets the
write protection bit after the NV block was written the first time. This means that some of the NV blocks in the NVRAM should never be erased ..after first initialization.

Also, it should be made clear that "write protection", "write protection bit", "write protection attribute" all refer to the current known state of write protection of a block.

Agreed solution:

(1) Change the Description for requirement ECUC_NvM_00072 to:
"Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block was already written and it is detected as valid and consistent during a read for it. [NVM276].
true: Defines write protection after first write is enabled.
false: Defines write protection after first write is disabled."

(2) Change the requirement SWS_NvM_00316 to:
"The job of the function NvM_ReadBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(3) Change the requirement SWS_NvM_00314 to:
"The job of the function NvM_ReadAll shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(4) Change the requirement SWS_NvM_00784 to:
"The job of the function NvM_ReadPRAMBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(5) Add a requirement, in chapter "7.2.2.13 NVRAM block write protection", stating the following:
"For a block configured with MVM_WRITE_BLOCK_ONCE (TRUE), NvM shall

reject any Write/Erase/Invalidate request made prior to the first read request."

(6) For the above requirement, add the following Rationale:
"In case of a reset, the write protection flag of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE), from the NvM Administrative block, is cleared. In order to reactivate the protection, the block must be read prior to a first Write/Erase/Invalidate request being processed, in order to set the write proctection only for a block that is valid and consistent. The first read request can be done either as a single block request or as part of NvM_ReadAll."

(7) Add a requirement, in chapter "7.3.1 Development errors", stating the following:
"The development error NVM_E_WRITE_ONCE_STATUS_UNKNOWN (0x1A) shall be detectable by the NvM module when a Write/Erase/Invalidate is made for a block with MVM_WRITE_BLOCK_ONCE (TRUE), prior to the first read request made for that block, depending on whether the build version mode is development mode."

(8) Add the following requirements in chapter "7.4 Error Detection":
- If development error detection is enabled for NvM module, the function NvM_WriteBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the function NvM_WritePRAMBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_WriteAll shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when the processing of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_EraseNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_InvalidateNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

(9) Add a requirement, in chapter "8.1.3.2.4 NvM_EraseNvBlock", after the 00423 requirement, which states the following:
"The job of the function NvM_EraseNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."

(10) Add a requirement, in chapter "8.1.3.2.6 NvM_InvalidateNvBlock", after the 00417 requirement, which states the following:
"The job of the function NvM_InvalidateNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."
–Last change on issue 74058 comment 15–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.5 Specification Item SWS_NvM_00316

Trace References:

none

Content:

The job of the function NvM_ReadBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) that is not detected by underlying SW as being invalidated, shall be marked as write protected as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74058: [NvM] Write protection and erase requests for NvMWriteBlockOnce blocks

  Problem description:

  Name: Delia Batica
  Phone: +40 356 78 4202
  Role: Developer

  Description/Motivation:

Currently, the behavior of NvM API's for written blocks with NvMWriteBlockOnce configured TRUE that have write protection unset is unclear.

I will refer as wbo the blocks with NvMWriteBlockOnce configured TRUE.

How should the wbo be handled during read requests if the block is reported as inconsistent?
In SWS_NvM_00316 for eg., NvM_ReadBlock sets wbo to write protected if block is valid. (same for SWS_NvM_00314, SWS_NvM_00784).

How should NvM_EraseNvBlock and NvM_InvalidateNvBlock handle a wbo if the write protection is still unset? Should the 2 APIs set the write protection for a block configured with NvMWriteBlockOnce set to TRUE?

NVM072_Conf : says that NVRAM manager".. sets the
write protection bit after the NV block was written the first time. This means that some of the NV blocks in the NVRAM should never be erased ..after first initialization.

Also, it should be made clear that "write protection", "write protection bit", "write protection attribute" all refer to the current known state of write protection of a block.

Agreed solution:

(1) Change the Description for requirement ECUC_NvM_00072 to:
"Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block was already written and it is detected as valid and consistent during a read for it. [NVM276].
true: Defines write protection after first write is enabled.
false: Defines write protection after first write is disabled."

(2) Change the requirement SWS_NvM_00316 to:
"The job of the function NvM_ReadBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(3) Change the requirement SWS_NvM_00314 to:
"The job of the function NvM_ReadAll shall mark every NVRAM block that has been

configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(4) Change the requirement SWS_NvM_00784 to:
"The job of the function NvM_ReadPRAMBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(5) Add a requirement, in chapter "7.2.2.13 NVRAM block write protection", stating the following:
"For a block configured with MVM_WRITE_BLOCK_ONCE (TRUE), NvM shall reject any Write/Erase/Invalidate request made prior to the first read request."

(6) For the above requirement, add the following Rationale:
"In case of a reset, the write protection flag of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE), from the NvM Administrative block, is cleared. In order to reactivate the protection, the block must be read prior to a first Write/Erase/Invalidate request being processed, in order to set the write proctection only for a block that is valid and consistent. The first read request can be done either as a single block request or as part of NvM_ReadAll."

(7) Add a requirement, in chapter "7.3.1 Development errors", stating the following:
"The development error NVM_E_WRITE_ONCE_STATUS_UNKNOWN (0x1A) shall be detectable by the NvM module when a Write/Erase/Invalidate is made for a block with MVM_WRITE_BLOCK_ONCE (TRUE), prior to the first read request made for that block, depending on whether the build version mode is development mode."

(8) Add the following requirements in chapter "7.4 Error Detection":
- If development error detection is enabled for NvM module, the function NvM_WriteBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the function NvM_WritePRAMBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_WriteAll shall report the DET error

NVM_E_WRITE_ONCE_STATUS_UNKNOWN when the processing of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_EraseNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_InvalidateNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

(9) Add a requirement, in chapter "8.1.3.2.4 NvM_EraseNvBlock", after the 00423 requirement, which states the following:
"The job of the function NvM_EraseNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."

(10) Add a requirement, in chapter "8.1.3.2.6 NvM_InvalidateNvBlock", after the 00417 requirement, which states the following:
"The job of the function NvM_InvalidateNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."
–Last change on issue 74058 comment 15–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.6 Specification Item SWS_NvM_00628

Trace References:

none

Content:

If development error detection is enabled for NvM module, the function NvM_RestoreBlockDefaults shall report the DET error NVM_E_BLOCK_CONFIG when Default data is not available/configured for the referenced NVRAM block.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76453: [Nvm] incorrect DET error for no default data and no NvMInitBlockCallback + redundant requirements

  Problem description:

  1) It seems that DET errors in [SWS_NvM_00628] and [SWS_NvM_00833] should be changed to NVM_E_BLOCK_WITHOUT_DEFAULTS (from current NVM_E_BLOCK_CONFIG), according to newly added requirements (@ R4.2.2) [SWS_NvM_00887] and [SWS_NvM_00885].

  2) [SWS_NvM_00628] is already covered by newly added requirement (@ R4.2.2) [SWS_NvM_00887]. Removal required.

  3) [SWS_NvM_00833] is already covered by newly added requirement (@ R4.2.2) [SWS_NvM_00887]. Removal required.

  4) [SWS_NvM_00885] is also redundant requirement of [SWS_NvM_00887]. Removal required.

  FYI – Just for helping quick understanding for the document history:

  [SWS_NvM_00887] The NVM_E_BLOCK_WITHOUT_DEFAULTS (0x11) development error shall be detectable by the NvM module when either the NvM_RestoreBlockDeafults or NvM_RestorePRAMBlockDefaults is called for a valid block ID that has no default data and no NvMInitBlockCallback configured for it. ( )
  (in sec. 7.3.1 Development Errors)
  @R413: not available
  @R421: not available
  @R422: available (added here)
  @R430: available

  [SWS_NvM_00885] If the block has no default data, it has no InitBlockCallbackFunction configured and the development error detection is enabled then the NvM_RestoreBlockDefaults API shall report the error NVM_E_BLOCK_WITHOUT_DEFAULTS error to the Det module. ( )
  (in sec. 8.1.3.2.3 NvM_RestoreBlockDefaults)
  @R413: not available
  @R421: not available
  @R422: available (added here)
  @R430: available

  [SWS_NvM_00628] If development error detection is enabled for NvM mod-

ule, the function NvM_RestoreBlockDefaults shall report the DET error NVM_E_BLOCK_CONFIG when Default data is not available/configured for the referenced NVRAM block. ( )
(in sec. 7.4 Error detection)
@R413: available
@R430: available

[SWS_NvM_00833] If development error detection is enabled for NvM module, the function NvM_RestorePRAMBlockDefaults shall report the DET error NVM_E_BLOCK_CONFIG when Default data is not available/configured for the referenced NVRAM block. ( )
(in sec. 7.4 Error detection)
@R413: available
@R430: available
–Last change on issue 76453 comment 11–

Agreed solution:

Remove the requirements SWS_NvM_00628 and SWS_NvM_00833 (they are redundant with the requirements SWS_NvM_00885 and SWS_NvM_00886).
–Last change on issue 76453 comment 5–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.7 Specification Item SWS_NvM_00784

Trace References:

none

Content:

The job of the function NvM_ReadPRAMBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) that is not detected by underlying SW as being invalidated, shall be marked as write protected as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74058: [NvM] Write protection and erase requests for NvMWriteBlockOnce blocks

Problem description:

Name: Delia Batica
Phone: +40 356 78 4202
Role: Developer

Description/Motivation:

Currently, the behavior of NvM API's for written blocks with NvMWriteBlockOnce configured TRUE that have write protection unset is unclear.

I will refer as wbo the blocks with NvMWriteBlockOnce configured TRUE.

How should the wbo be handled during read requests if the block is reported as inconsistent?
In SWS_NvM_00316 for eg., NvM_ReadBlock sets wbo to write protected if block is valid. (same for SWS_NvM_00314, SWS_NvM_00784).

How should NvM_EraseNvBlock and NvM_InvalidateNvBlock handle a wbo if the write protection is still unset? Should the 2 APIs set the write protection for a block configured with NvMWriteBlockOnce set to TRUE?

NVM072_Conf : says that NVRAM manager".. sets the
write protection bit after the NV block was written the first time. This means that some of the NV blocks in the NVRAM should never be erased ..after first initialization.

Also, it should be made clear that "write protection", "write protection bit", "write protection attribute" all refer to the current known state of write protection of a block.

Agreed solution:

(1) Change the Description for requirement ECUC_NvM_00072 to:
"Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block was already written and it is detected as valid and consistent during a read for it. [NVM276].
true: Defines write protection after first write is enabled.
false: Defines write protection after first write is disabled."

(2) Change the requirement SWS_NvM_00316 to:
"The job of the function NvM_ReadBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(3) Change the requirement SWS_NvM_00314 to:
"The job of the function NvM_ReadAll shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(4) Change the requirement SWS_NvM_00784 to:
"The job of the function NvM_ReadPRAMBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(5) Add a requirement, in chapter "7.2.2.13 NVRAM block write protection", stating the following:
"For a block configured with MVM_WRITE_BLOCK_ONCE (TRUE), NvM shall reject any Write/Erase/Invalidate request made prior to the first read request."

(6) For the above requirement, add the following Rationale:
"In case of a reset, the write protection flag of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE), from the NvM Administrative block, is cleared. In order to reactivate the protection, the block must be read prior to a first Write/Erase/Invalidate request being processed, in order to set the write proctection only for a block that is valid and consistent. The first read request can be done either as a single block request or as part of NvM_ReadAll."

(7) Add a requirement, in chapter "7.3.1 Development errors", stating the following:
"The development error NVM_E_WRITE_ONCE_STATUS_UNKNOWN (0x1A) shall be detectable by the NvM module when a Write/Erase/Invalidate is made for a block with MVM_WRITE_BLOCK_ONCE (TRUE), prior to the first read request made for that block, depending on whether the build version mode is development mode."

(8) Add the following requirements in chapter "7.4 Error Detection":
- If development error detection is enabled for NvM module, the function NvM_WriteBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made

for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the function NvM_WritePRAMBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_WriteAll shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when the processing of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_EraseNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_InvalidateNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

(9) Add a requirement, in chapter "8.1.3.2.4 NvM_EraseNvBlock", after the 00423 requirement, which states the following:
"The job of the function NvM_EraseNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."

(10) Add a requirement, in chapter "8.1.3.2.6 NvM_InvalidateNvBlock", after the 00417 requirement, which states the following:
"The job of the function NvM_InvalidateNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."
–Last change on issue 74058 comment 15–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.8 Specification Item SWS_NvM_00833

Trace References:

none

Content:

If development error detection is enabled for NvM module, the function NvM_RestorePRAMBlockDefaults shall report the DET error NVM_E_BLOCK_CONFIG when Default data is not available/configured for the referenced NVRAM block.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76453: [Nvm] incorrect DET error for no default data and no NvMInitBlockCallback + redundant requirements

  Problem description:

  1) It seems that DET errors in [SWS_NvM_00628] and [SWS_NvM_00833] should be changed to NVM_E_BLOCK_WITHOUT_DEFAULTS (from current NVM_E_BLOCK_CONFIG), according to newly added requirements (@ R4.2.2) [SWS_NvM_00887] and [SWS_NvM_00885].

  2) [SWS_NvM_00628] is already covered by newly added requirement (@ R4.2.2) [SWS_NvM_00887]. Removal required.

  3) [SWS_NvM_00833] is already covered by newly added requirement (@ R4.2.2) [SWS_NvM_00887]. Removal required.

  4) [SWS_NvM_00885] is also redundant requirement of [SWS_NvM_00887]. Removal required.

  FYI – Just for helping quick understanding for the document history:

  [SWS_NvM_00887] The NVM_E_BLOCK_WITHOUT_DEFAULTS (0x11) development error shall be detectable by the NvM module when either the NvM_RestoreBlockDeafults or NvM_RestorePRAMBlockDefaults is called for a valid block ID that has no default data and no NvMInitBlockCallback configured for it. ( )
  (in sec. 7.3.1 Development Errors)
  @R413: not available
  @R421: not available
  @R422: available (added here)
  @R430: available

  [SWS_NvM_00885] If the block has no default data, it has no InitBlockCallbackFunction configured and the development error detection is en-

abled then the NvM_RestoreBlockDefaults API shall report the error NVM_E_BLOCK_WITHOUT_DEFAULTS error to the Det module. ( )
(in sec. 8.1.3.2.3 NvM_RestoreBlockDefaults)
@R413: not available
@R421: not available
@R422: available (added here)
@R430: available

[SWS_NvM_00628] If development error detection is enabled for NvM module, the function NvM_RestoreBlockDefaults shall report the DET error NVM_E_BLOCK_CONFIG when Default data is not available/configured for the referenced NVRAM block. ( )
(in sec. 7.4 Error detection)
@R413: available
@R430: available

[SWS_NvM_00833] If development error detection is enabled for NvM module, the function NvM_RestorePRAMBlockDefaults shall report the DET error NVM_E_BLOCK_CONFIG when Default data is not available/configured for the referenced NVRAM block. ( )
(in sec. 7.4 Error detection)
@R413: available
@R430: available
–Last change on issue 76453 comment 11–

Agreed solution:

Remove the requirements SWS_NvM_00628 and SWS_NvM_00833 (they are redundant with the requirements SWS_NvM_00885 and SWS_NvM_00886).
–Last change on issue 76453 comment 5–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.9 Specification Item SWS_NvM_00856

Trace References:

SRS_Mem_00137

Content:

If auto validation is configured for an NVRAM Block (NvMBlockUseAutoValidation == TRUE)and the RAM Block status is not INVALID , the function NvM_ValidateAll shall set the RAM Block status to "VALID / CHANGED".

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #72401: NvM_ValidateAll only processes blocks with the RAM block status VALID

  Problem description:

  According to SWS_NvM_00856 NvM_ValidateAll shall not process blocks that have the RAM block status INVALID. This means that the service NvM_ValidateAll does not validate blocks, it shall only mark the blocks as changed.

  [SWS_NvM_00856] If auto validation is configured for an NVRAM Block (NvM-BlockUseAutoValidation == TRUE) and the RAM Block status is not INVALID the function NvM_ValidateAll shall set the RAM Block status to VALID / CHANGED.

  In my opinion the NvM_ValidateAll shall first validate and then mark as CHANGED all blocks configured with NvMBlockUseAutoValidation = TRUE. Is this correct?

  Agreed solution:

  Update [SWS_NvM_00856] If auto validation is configured for an NVRAM Block (NvMBlockUseAutoValidation == TRUE), the function NvM_ValidateAll shall set the RAM Block status to VALID / CHANGED.
  –Last change on issue 72401 comment 10–

  BW-C-Level:

Application	Specification	Bus
4	4	1

1.10 Specification Item SWS_NvM_00951

Trace References:

SRS_Mem_00018

Content:

Implicit recovery shall be provided during NvM_ReadBlock() or NvM_ReadPRAMBlock() requests for NVRAM blocks of type NVM_BLOCK_NATIVE and NVM_BLOCK_REDUNDANT.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #72921: Clarification regarding implicit recovery of dataset blocks

  Problem description:

  According to requirement SWS_NvM_00657, NvM shall restore default data for a block (regardless of the block management type) if reading the NV memory fails:

  [SWS_NvM_00657] The job of the function NvM_ReadBlock shall load the default values according to processing of NvM_RestoreBlockDefaults (also set the job result to NVM_REQ_RESTORED_FROM_ROM) if the read request passed to the underlying layer fails (MemIf reports MEMIF_JOB_FAILED or MEMIF_BLOCK_INCONSISTENT) and if the default values are available.

  Additionally, requirement SWS_NvM_00353 states that NvM_RestoreBlockDefaults() shall return with E_NOT_OK if the block management type is NVM_BLOCK_DATASET and the data index points to an NV block:

  [SWS_NvM_00353] The function NvM_RestoreBlockDefaults shall return with E_NOT_OK if the block management type of the given NVRAM block is NVM_BLOCK_DATASET, at least one ROM block is configured and the data index points at a NV block.

  Other relevant requirements:
  [SWS_NvM_00340] In case of NVRAM block management type NVM_BLOCK_DATASET, the job of the function NvM_ReadBlock shall copy only that NV block to the corresponding RAM block which is selected via the data index in the administrative block.

  [SWS_NvM_00354] The job of the function NvM_ReadBlock shall copy the ROM block to RAM and set the job result to NVM_REQ_OK if the NVRAM block management type is NVM_BLOCK_DATASET and the dataset index points at a ROM block.

  understanding: the requirements above are in contradiction and the scenario is not covered by the NvM specifications.
  –Last change on issue 72921 comment 24–

  Agreed solution:

  Add a new requirement in chapter "7.2.2.7 Implicit recovery of a RAM block with ROM default data" stating:

[SWS_NvM_00xxx] Implicit recovery shall be provided during NvM_ReadBlock() or NvM_ReadPRAMBlock() requests for NVRAM blocks of type NVM_BLOCK_NATIVE and NVM_BLOCK_REDUNDANT.
–Last change on issue 72921 comment 17–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.11 Specification Item SWS_NvM_00952

Trace References:

none

Content:

For a block configured with MVM_WRITE_BLOCK_ONCE (TRUE), NvM shall reject any Write/Erase/Invalidate request made prior to the first read request.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74058: [NvM] Write protection and erase requests for NvMWriteBlockOnce blocks

  Problem description:

  Name: Delia Batica
  Phone: +40 356 78 4202
  Role: Developer

  Description/Motivation:

  Currently, the behavior of NvM API's for written blocks with NvMWriteBlockOnce configured TRUE that have write protection unset is unclear.

  I will refer as wbo the blocks with NvMWriteBlockOnce configured TRUE.

  How should the wbo be handled during read requests if the block is reported as inconsistent?
  In SWS_NvM_00316 for eg., NvM_ReadBlock sets wbo to write protected if block is valid. (same for SWS_NvM_00314, SWS_NvM_00784).

How should NvM_EraseNvBlock and NvM_InvalidateNvBlock handle a wbo if the write protection is still unset? Should the 2 APIs set the write protection for a block configured with NvMWriteBlockOnce set to TRUE?

NVM072_Conf : says that NVRAM manager".. sets the
write protection bit after the NV block was written the first time. This means that some of the NV blocks in the NVRAM should never be erased ..after first initialization.

Also, it should be made clear that "write protection", "write protection bit", "write protection attribute" all refer to the current known state of write protection of a block.

Agreed solution:

(1) Change the Description for requirement ECUC_NvM_00072 to:
"Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block was already written and it is detected as valid and consistent during a read for it. [NVM276].
true: Defines write protection after first write is enabled.
false: Defines write protection after first write is disabled."

(2) Change the requirement SWS_NvM_00316 to:
"The job of the function NvM_ReadBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(3) Change the requirement SWS_NvM_00314 to:
"The job of the function NvM_ReadAll shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(4) Change the requirement SWS_NvM_00784 to:
"The job of the function NvM_ReadPRAMBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(5) Add a requirement, in chapter "7.2.2.13 NVRAM block write protection", stating the following:
"For a block configured with MVM_WRITE_BLOCK_ONCE (TRUE), NvM shall reject any Write/Erase/Invalidate request made prior to the first read request."

(6) For the above requirement, add the following Rationale:
"In case of a reset, the write protection flag of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE), from the NvM Administrative block, is cleared. In order to reactivate the protection, the block must be read prior to a first Write/Erase/Invalidate request being processed, in order to set the write proctection only for a block that is valid and consistent. The first read request can be done either as a single block request or as part of NvM_ReadAll."

(7) Add a requirement, in chapter "7.3.1 Development errors", stating the following:
"The development error NVM_E_WRITE_ONCE_STATUS_UNKNOWN (0x1A) shall be detectable by the NvM module when a Write/Erase/Invalidate is made for a block with MVM_WRITE_BLOCK_ONCE (TRUE), prior to the first read request made for that block, depending on whether the build version mode is development mode."

(8) Add the following requirements in chapter "7.4 Error Detection":
- If development error detection is enabled for NvM module, the function NvM_WriteBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the function NvM_WritePRAMBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_WriteAll shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when the processing of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_EraseNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_InvalidateNvBlock shall report the DET error

NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

(9) Add a requirement, in chapter "8.1.3.2.4 NvM_EraseNvBlock", after the 00423 requirement, which states the following:
"The job of the function NvM_EraseNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."

(10) Add a requirement, in chapter "8.1.3.2.6 NvM_InvalidateNvBlock", after the 00417 requirement, which states the following:
"The job of the function NvM_InvalidateNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."
–Last change on issue 74058 comment 15–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.12 Specification Item SWS_NvM_00953

Trace References:

none

Content:

The development error NVM_E_WRITE_ONCE_STATUS_UNKNOWN (0x1A) shall be detectable by the NvM module when a Write/Erase/Invalidate is made for a block with MVM_WRITE_BLOCK_ONCE (TRUE), prior to the first read request made for that block, depending on whether the build version mode is development mode.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74058: [NvM] Write protection and erase requests for NvMWriteBlockOnce blocks

  Problem description:

  Name: Delia Batica
  Phone: +40 356 78 4202
  Role: Developer

Description/Motivation:

Currently, the behavior of NvM API's for written blocks with NvMWriteBlockOnce configured TRUE that have write protection unset is unclear.

I will refer as wbo the blocks with NvMWriteBlockOnce configured TRUE.

How should the wbo be handled during read requests if the block is reported as inconsistent?
In SWS_NvM_00316 for eg., NvM_ReadBlock sets wbo to write protected if block is valid. (same for SWS_NvM_00314, SWS_NvM_00784).

How should NvM_EraseNvBlock and NvM_InvalidateNvBlock handle a wbo if the write protection is still unset? Should the 2 APIs set the write protection for a block configured with NvMWriteBlockOnce set to TRUE?

NVM072_Conf : says that NVRAM manager".. sets the
write protection bit after the NV block was written the first time. This means that some of the NV blocks in the NVRAM should never be erased ..after first initialization.

Also, it should be made clear that "write protection", "write protection bit", "write protection attribute" all refer to the current known state of write protection of a block.

Agreed solution:

(1) Change the Description for requirement ECUC_NvM_00072 to:
"Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block was already written and it is detected as valid and consistent during a read for it. [NVM276].
true: Defines write protection after first write is enabled.
false: Defines write protection after first write is disabled."

(2) Change the requirement SWS_NvM_00316 to:
"The job of the function NvM_ReadBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(3) Change the requirement SWS_NvM_00314 to:
"The job of the function NvM_ReadAll shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(4) Change the requirement SWS_NvM_00784 to:
"The job of the function NvM_ReadPRAMBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(5) Add a requirement, in chapter "7.2.2.13 NVRAM block write protection", stating the following:
"For a block configured with MVM_WRITE_BLOCK_ONCE (TRUE), NvM shall reject any Write/Erase/Invalidate request made prior to the first read request."

(6) For the above requirement, add the following Rationale:
"In case of a reset, the write protection flag of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE), from the NvM Administrative block, is cleared. In order to reactivate the protection, the block must be read prior to a first Write/Erase/Invalidate request being processed, in order to set the write proctection only for a block that is valid and consistent. The first read request can be done either as a single block request or as part of NvM_ReadAll."

(7) Add a requirement, in chapter "7.3.1 Development errors", stating the following:
"The development error NVM_E_WRITE_ONCE_STATUS_UNKNOWN (0x1A) shall be detectable by the NvM module when a Write/Erase/Invalidate is made for a block with MVM_WRITE_BLOCK_ONCE (TRUE), prior to the first read request made for that block, depending on whether the build version mode is development mode."

(8) Add the following requirements in chapter "7.4 Error Detection":
- If development error detection is enabled for NvM module, the function NvM_WriteBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the function NvM_WritePRAMBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

- If development error detection is enabled for NvM module, the job of the function NvM_WriteAll shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when the processing of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_EraseNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_InvalidateNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

(9) Add a requirement, in chapter "8.1.3.2.4 NvM_EraseNvBlock", after the 00423 requirement, which states the following:
"The job of the function NvM_EraseNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."

(10) Add a requirement, in chapter "8.1.3.2.6 NvM_InvalidateNvBlock", after the 00417 requirement, which states the following:
"The job of the function NvM_InvalidateNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."
–Last change on issue 74058 comment 15–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.13 Specification Item SWS_NvM_00954

Trace References:

none

Content:

If development error detection is enabled for NvM module, the function NvM_WriteBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74058: [NvM] Write protection and erase requests for NvMWriteBlockOnce blocks

  Problem description:

  Name: Delia Batica
  Phone: +40 356 78 4202
  Role: Developer

  Description/Motivation:

  Currently, the behavior of NvM API's for written blocks with NvMWriteBlockOnce configured TRUE that have write protection unset is unclear.

  I will refer as wbo the blocks with NvMWriteBlockOnce configured TRUE.

  How should the wbo be handled during read requests if the block is reported as inconsistent?
  In SWS_NvM_00316 for eg., NvM_ReadBlock sets wbo to write protected if block is valid. (same for SWS_NvM_00314, SWS_NvM_00784).

  How should NvM_EraseNvBlock and NvM_InvalidateNvBlock handle a wbo if the write protection is still unset? Should the 2 APIs set the write protection for a block configured with NvMWriteBlockOnce set to TRUE?

  NVM072_Conf : says that NVRAM manager".. sets the
  write protection bit after the NV block was written the first time. This means that some of the NV blocks in the NVRAM should never be erased ..after first initialization.

  Also, it should be made clear that "write protection", "write protection bit", "write protection attribute" all refer to the current known state of write protection of a block.

  Agreed solution:

  (1) Change the Description for requirement ECUC_NvM_00072 to:
  "Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block

was already written and it is detected as valid and consistent during a read for it. [NVM276].
true: Defines write protection after first write is enabled.
false: Defines write protection after first write is disabled."

(2) Change the requirement SWS_NvM_00316 to:
"The job of the function NvM_ReadBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(3) Change the requirement SWS_NvM_00314 to:
"The job of the function NvM_ReadAll shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(4) Change the requirement SWS_NvM_00784 to:
"The job of the function NvM_ReadPRAMBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(5) Add a requirement, in chapter "7.2.2.13 NVRAM block write protection", stating the following:
"For a block configured with MVM_WRITE_BLOCK_ONCE (TRUE), NvM shall reject any Write/Erase/Invalidate request made prior to the first read request."

(6) For the above requirement, add the following Rationale:
"In case of a reset, the write protection flag of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE), from the NvM Administrative block, is cleared. In order to reactivate the protection, the block must be read prior to a first Write/Erase/Invalidate request being processed, in order to set the write proctection only for a block that is valid and consistent. The first read request can be done either as a single block request or as part of NvM_ReadAll."

(7) Add a requirement, in chapter "7.3.1 Development errors", stating the following:
"The development error NVM_E_WRITE_ONCE_STATUS_UNKNOWN (0x1A) shall be detectable by the NvM module when a Write/Erase/Invalidate is made for a block with MVM_WRITE_BLOCK_ONCE (TRUE), prior to the first read request made for that block, depending on whether the build version mode is development mode."

(8) Add the following requirements in chapter "7.4 Error Detection":
- If development error detection is enabled for NvM module, the function NvM_WriteBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the function NvM_WritePRAMBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_WriteAll shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when the processing of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_EraseNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_InvalidateNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

(9) Add a requirement, in chapter "8.1.3.2.4 NvM_EraseNvBlock", after the 00423 requirement, which states the following:
"The job of the function NvM_EraseNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."

(10) Add a requirement, in chapter "8.1.3.2.6 NvM_InvalidateNvBlock", after the 00417 requirement, which states the following:
"The job of the function NvM_InvalidateNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."
–Last change on issue 74058 comment 15–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.14 Specification Item SWS_NvM_00955

Trace References:

none

Content:

If development error detection is enabled for NvM module, the function NvM_WritePRAMBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74058: [NvM] Write protection and erase requests for NvMWriteBlockOnce blocks

  Problem description:

  Name: Delia Batica
  Phone: +40 356 78 4202
  Role: Developer

  Description/Motivation:

  Currently, the behavior of NvM API's for written blocks with NvMWriteBlockOnce configured TRUE that have write protection unset is unclear.

  I will refer as wbo the blocks with NvMWriteBlockOnce configured TRUE.

  How should the wbo be handled during read requests if the block is reported as inconsistent?
  In SWS_NvM_00316 for eg., NvM_ReadBlock sets wbo to write protected if block is valid. (same for SWS_NvM_00314, SWS_NvM_00784).

  How should NvM_EraseNvBlock and NvM_InvalidateNvBlock handle a wbo if the write protection is still unset? Should the 2 APIs set the write protection for a block configured with NvMWriteBlockOnce set to TRUE?

  NVM072_Conf : says that NVRAM manager".. sets the
  write protection bit after the NV block was written the first time. This means that some of the NV blocks in the NVRAM should never be erased ..after first initialization.

Also, it should be made clear that "write protection", "write protection bit", "write protection attribute" all refer to the current known state of write protection of a block.

Agreed solution:

(1) Change the Description for requirement ECUC_NvM_00072 to:
"Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block was already written and it is detected as valid and consistent during a read for it. [NVM276].
true: Defines write protection after first write is enabled.
false: Defines write protection after first write is disabled."

(2) Change the requirement SWS_NvM_00316 to:
"The job of the function NvM_ReadBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(3) Change the requirement SWS_NvM_00314 to:
"The job of the function NvM_ReadAll shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(4) Change the requirement SWS_NvM_00784 to:
"The job of the function NvM_ReadPRAMBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(5) Add a requirement, in chapter "7.2.2.13 NVRAM block write protection", stating the following:
"For a block configured with MVM_WRITE_BLOCK_ONCE (TRUE), NvM shall reject any Write/Erase/Invalidate request made prior to the first read request."

(6) For the above requirement, add the following Rationale:
"In case of a reset, the write protection flag of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE), from the NvM Administrative block, is

cleared. In order to reactivate the protection, the block must be read prior to a first Write/Erase/Invalidate request being processed, in order to set the write proctection only for a block that is valid and consistent. The first read request can be done either as a single block request or as part of NvM_ReadAll."

(7) Add a requirement, in chapter "7.3.1 Development errors", stating the following:
"The development error NVM_E_WRITE_ONCE_STATUS_UNKNOWN (0x1A) shall be detectable by the NvM module when a Write/Erase/Invalidate is made for a block with MVM_WRITE_BLOCK_ONCE (TRUE), prior to the first read request made for that block, depending on whether the build version mode is development mode."

(8) Add the following requirements in chapter "7.4 Error Detection":
- If development error detection is enabled for NvM module, the function NvM_WriteBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the function NvM_WritePRAMBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_WriteAll shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when the processing of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_EraseNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_InvalidateNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

(9) Add a requirement, in chapter "8.1.3.2.4 NvM_EraseNvBlock", after the 00423 requirement, which states the following:
"The job of the function NvM_EraseNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."

(10) Add a requirement, in chapter "8.1.3.2.6 NvM_InvalidateNvBlock", after the 00417 requirement, which states the following:
"The job of the function NvM_InvalidateNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."
–Last change on issue 74058 comment 15–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.15 Specification Item SWS_NvM_00956

Trace References:

none

Content:

If development error detection is enabled for NvM module, the job of the function NvM_WriteAll shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when the processing of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74058: [NvM] Write protection and erase requests for NvMWriteBlockOnce blocks

  Problem description:

  Name: Delia Batica
  Phone: +40 356 78 4202
  Role: Developer

  Description/Motivation:

  Currently, the behavior of NvM API's for written blocks with NvMWriteBlockOnce configured TRUE that have write protection unset is unclear.

  I will refer as wbo the blocks with NvMWriteBlockOnce configured TRUE.

  How should the wbo be handled during read requests if the block is reported

as inconsistent?
In SWS_NvM_00316 for eg., NvM_ReadBlock sets wbo to write protected if block is valid. (same for SWS_NvM_00314, SWS_NvM_00784).

How should NvM_EraseNvBlock and NvM_InvalidateNvBlock handle a wbo if the write protection is still unset? Should the 2 APIs set the write protection for a block configured with NvMWriteBlockOnce set to TRUE?

NVM072_Conf : says that NVRAM manager".. sets the
write protection bit after the NV block was written the first time. This means that some of the NV blocks in the NVRAM should never be erased ..after first initialization.

Also, it should be made clear that "write protection", "write protection bit", "write protection attribute" all refer to the current known state of write protection of a block.

Agreed solution:

(1) Change the Description for requirement ECUC_NvM_00072 to:
"Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block was already written and it is detected as valid and consistent during a read for it. [NVM276].
true: Defines write protection after first write is enabled.
false: Defines write protection after first write is disabled."

(2) Change the requirement SWS_NvM_00316 to:
"The job of the function NvM_ReadBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(3) Change the requirement SWS_NvM_00314 to:
"The job of the function NvM_ReadAll shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(4) Change the requirement SWS_NvM_00784 to:
"The job of the function NvM_ReadPRAMBlock shall mark every NVRAM block that

has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(5) Add a requirement, in chapter "7.2.2.13 NVRAM block write protection", stating the following:
"For a block configured with MVM_WRITE_BLOCK_ONCE (TRUE), NvM shall reject any Write/Erase/Invalidate request made prior to the first read request."

(6) For the above requirement, add the following Rationale:
"In case of a reset, the write protection flag of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE), from the NvM Administrative block, is cleared. In order to reactivate the protection, the block must be read prior to a first Write/Erase/Invalidate request being processed, in order to set the write proctection only for a block that is valid and consistent. The first read request can be done either as a single block request or as part of NvM_ReadAll."

(7) Add a requirement, in chapter "7.3.1 Development errors", stating the following:
"The development error NVM_E_WRITE_ONCE_STATUS_UNKNOWN (0x1A) shall be detectable by the NvM module when a Write/Erase/Invalidate is made for a block with MVM_WRITE_BLOCK_ONCE (TRUE), prior to the first read request made for that block, depending on whether the build version mode is development mode."

(8) Add the following requirements in chapter "7.4 Error Detection":
- If development error detection is enabled for NvM module, the function NvM_WriteBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the function NvM_WritePRAMBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_WriteAll shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when the processing of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_EraseNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made

for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_InvalidateNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

(9) Add a requirement, in chapter "8.1.3.2.4 NvM_EraseNvBlock", after the 00423 requirement, which states the following:
"The job of the function NvM_EraseNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."

(10) Add a requirement, in chapter "8.1.3.2.6 NvM_InvalidateNvBlock", after the 00417 requirement, which states the following:
"The job of the function NvM_InvalidateNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."
–Last change on issue 74058 comment 15–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.16 Specification Item SWS_NvM_00957

Trace References:

none

Content:

If development error detection is enabled for NvM module, the job of the function NvM_EraseNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74058: [NvM] Write protection and erase requests for NvMWriteBlockOnce blocks

  Problem description:

Name: Delia Batica
Phone: +40 356 78 4202
Role: Developer

Description/Motivation:

Currently, the behavior of NvM API's for written blocks with NvMWriteBlockOnce configured TRUE that have write protection unset is unclear.

I will refer as wbo the blocks with NvMWriteBlockOnce configured TRUE.

How should the wbo be handled during read requests if the block is reported as inconsistent?
In SWS_NvM_00316 for eg., NvM_ReadBlock sets wbo to write protected if block is valid. (same for SWS_NvM_00314, SWS_NvM_00784).

How should NvM_EraseNvBlock and NvM_InvalidateNvBlock handle a wbo if the write protection is still unset? Should the 2 APIs set the write protection for a block configured with NvMWriteBlockOnce set to TRUE?

NVM072_Conf : says that NVRAM manager".. sets the
write protection bit after the NV block was written the first time. This means that some of the NV blocks in the NVRAM should never be erased ..after first initialization.

Also, it should be made clear that "write protection", "write protection bit", "write protection attribute" all refer to the current known state of write protection of a block.

Agreed solution:

(1) Change the Description for requirement ECUC_NvM_00072 to:
"Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block was already written and it is detected as valid and consistent during a read for it. [NVM276].
true: Defines write protection after first write is enabled.
false: Defines write protection after first write is disabled."

(2) Change the requirement SWS_NvM_00316 to:

"The job of the function NvM_ReadBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(3) Change the requirement SWS_NvM_00314 to:
"The job of the function NvM_ReadAll shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(4) Change the requirement SWS_NvM_00784 to:
"The job of the function NvM_ReadPRAMBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(5) Add a requirement, in chapter "7.2.2.13 NVRAM block write protection", stating the following:
"For a block configured with MVM_WRITE_BLOCK_ONCE (TRUE), NvM shall reject any Write/Erase/Invalidate request made prior to the first read request."

(6) For the above requirement, add the following Rationale:
"In case of a reset, the write protection flag of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE), from the NvM Administrative block, is cleared. In order to reactivate the protection, the block must be read prior to a first Write/Erase/Invalidate request being processed, in order to set the write proctection only for a block that is valid and consistent. The first read request can be done either as a single block request or as part of NvM_ReadAll."

(7) Add a requirement, in chapter "7.3.1 Development errors", stating the following:
"The development error NVM_E_WRITE_ONCE_STATUS_UNKNOWN (0x1A) shall be detectable by the NvM module when a Write/Erase/Invalidate is made for a block with MVM_WRITE_BLOCK_ONCE (TRUE), prior to the first read request made for that block, depending on whether the build version mode is development mode."

(8) Add the following requirements in chapter "7.4 Error Detection":
- If development error detection is enabled for NvM module, the function NvM_WriteBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

- If development error detection is enabled for NvM module, the function NvM_WritePRAMBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_WriteAll shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when the processing of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_EraseNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_InvalidateNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

(9) Add a requirement, in chapter "8.1.3.2.4 NvM_EraseNvBlock", after the 00423 requirement, which states the following:
"The job of the function NvM_EraseNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."

(10) Add a requirement, in chapter "8.1.3.2.6 NvM_InvalidateNvBlock", after the 00417 requirement, which states the following:
"The job of the function NvM_InvalidateNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."
–Last change on issue 74058 comment 15–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.17 Specification Item SWS_NvM_00958

Trace References:

none

Content:

If development error detection is enabled for NvM module, the job of the function NvM_InvalidateNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74058: [NvM] Write protection and erase requests for NvMWriteBlockOnce blocks

  Problem description:

  Name: Delia Batica
  Phone: +40 356 78 4202
  Role: Developer

  Description/Motivation:

  Currently, the behavior of NvM API's for written blocks with NvMWriteBlockOnce configured TRUE that have write protection unset is unclear.

  I will refer as wbo the blocks with NvMWriteBlockOnce configured TRUE.

  How should the wbo be handled during read requests if the block is reported as inconsistent?
  In SWS_NvM_00316 for eg., NvM_ReadBlock sets wbo to write protected if block is valid. (same for SWS_NvM_00314, SWS_NvM_00784).

  How should NvM_EraseNvBlock and NvM_InvalidateNvBlock handle a wbo if the write protection is still unset? Should the 2 APIs set the write protection for a block configured with NvMWriteBlockOnce set to TRUE?

  NVM072_Conf : says that NVRAM manager".. sets the
  write protection bit after the NV block was written the first time. This means that some of the NV blocks in the NVRAM should never be erased ..after first initialization.

  Also, it should be made clear that "write protection", "write protection bit", "write protection attribute" all refer to the current known state of write protection of a block.

Agreed solution:

(1) Change the Description for requirement ECUC_NvM_00072 to:
"Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block was already written and it is detected as valid and consistent during a read for it. [NVM276].
true: Defines write protection after first write is enabled.
false: Defines write protection after first write is disabled."

(2) Change the requirement SWS_NvM_00316 to:
"The job of the function NvM_ReadBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(3) Change the requirement SWS_NvM_00314 to:
"The job of the function NvM_ReadAll shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(4) Change the requirement SWS_NvM_00784 to:
"The job of the function NvM_ReadPRAMBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(5) Add a requirement, in chapter "7.2.2.13 NVRAM block write protection", stating the following:
"For a block configured with MVM_WRITE_BLOCK_ONCE (TRUE), NvM shall reject any Write/Erase/Invalidate request made prior to the first read request."

(6) For the above requirement, add the following Rationale:
"In case of a reset, the write protection flag of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE), from the NvM Administrative block, is cleared. In order to reactivate the protection, the block must be read prior to a first Write/Erase/Invalidate request being processed, in order to set the write proctection only for a block that is valid and consistent. The first read request can be done either as a single block request or as part of NvM_ReadAll."

(7) Add a requirement, in chapter "7.3.1 Development errors", stating the following:
"The development error NVM_E_WRITE_ONCE_STATUS_UNKNOWN (0x1A) shall be detectable by the NvM module when a Write/Erase/Invalidate is made for a block with MVM_WRITE_BLOCK_ONCE (TRUE), prior to the first read request made for that block, depending on whether the build version mode is development mode."

(8) Add the following requirements in chapter "7.4 Error Detection":
- If development error detection is enabled for NvM module, the function NvM_WriteBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the function NvM_WritePRAMBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_WriteAll shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when the processing of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_EraseNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_InvalidateNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

(9) Add a requirement, in chapter "8.1.3.2.4 NvM_EraseNvBlock", after the 00423 requirement, which states the following:
"The job of the function NvM_EraseNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."

(10) Add a requirement, in chapter "8.1.3.2.6 NvM_InvalidateNvBlock", after the 00417 requirement, which states the following:
"The job of the function NvM_InvalidateNvBlock shall leave the write protection

unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."
–Last change on issue 74058 comment 15–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.18 Specification Item SWS_NvM_00959

Trace References:

none

Content:

The job of the function NvM_EraseNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74058: [NvM] Write protection and erase requests for NvMWriteBlockOnce blocks

  Problem description:

  Name: Delia Batica
  Phone: +40 356 78 4202
  Role: Developer

  Description/Motivation:

  Currently, the behavior of NvM API's for written blocks with NvMWriteBlockOnce configured TRUE that have write protection unset is unclear.

  I will refer as wbo the blocks with NvMWriteBlockOnce configured TRUE.

  How should the wbo be handled during read requests if the block is reported as inconsistent?
  In SWS_NvM_00316 for eg., NvM_ReadBlock sets wbo to write protected if block is valid. (same for SWS_NvM_00314, SWS_NvM_00784).

  How should NvM_EraseNvBlock and NvM_InvalidateNvBlock handle a wbo if the write protection is still unset? Should the 2 APIs set the write protection for a block configured with NvMWriteBlockOnce set to TRUE?

NVM072_Conf : says that NVRAM manager".. sets the
write protection bit after the NV block was written the first time. This means that some of the NV blocks in the NVRAM should never be erased ..after first initialization.

Also, it should be made clear that "write protection", "write protection bit", "write protection attribute" all refer to the current known state of write protection of a block.

Agreed solution:

(1) Change the Description for requirement ECUC_NvM_00072 to:
"Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block was already written and it is detected as valid and consistent during a read for it. [NVM276].
true: Defines write protection after first write is enabled.
false: Defines write protection after first write is disabled."

(2) Change the requirement SWS_NvM_00316 to:
"The job of the function NvM_ReadBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(3) Change the requirement SWS_NvM_00314 to:
"The job of the function NvM_ReadAll shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(4) Change the requirement SWS_NvM_00784 to:
"The job of the function NvM_ReadPRAMBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(5) Add a requirement, in chapter "7.2.2.13 NVRAM block write protection", stating the following:

"For a block configured with MVM_WRITE_BLOCK_ONCE (TRUE), NvM shall reject any Write/Erase/Invalidate request made prior to the first read request."

(6) For the above requirement, add the following Rationale:
"In case of a reset, the write protection flag of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE), from the NvM Administrative block, is cleared. In order to reactivate the protection, the block must be read prior to a first Write/Erase/Invalidate request being processed, in order to set the write proctection only for a block that is valid and consistent. The first read request can be done either as a single block request or as part of NvM_ReadAll."

(7) Add a requirement, in chapter "7.3.1 Development errors", stating the following:
"The development error NVM_E_WRITE_ONCE_STATUS_UNKNOWN (0x1A) shall be detectable by the NvM module when a Write/Erase/Invalidate is made for a block with MVM_WRITE_BLOCK_ONCE (TRUE), prior to the first read request made for that block, depending on whether the build version mode is development mode."

(8) Add the following requirements in chapter "7.4 Error Detection":
- If development error detection is enabled for NvM module, the function NvM_WriteBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the function NvM_WritePRAMBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_WriteAll shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when the processing of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_EraseNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_InvalidateNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read

request was made prior to this.

(9) Add a requirement, in chapter "8.1.3.2.4 NvM_EraseNvBlock", after the 00423 requirement, which states the following:
"The job of the function NvM_EraseNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."

(10) Add a requirement, in chapter "8.1.3.2.6 NvM_InvalidateNvBlock", after the 00417 requirement, which states the following:
"The job of the function NvM_InvalidateNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."
–Last change on issue 74058 comment 15–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.19 Specification Item SWS_NvM_00960

Trace References:

none

Content:

The job of the function NvM_InvalidateNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74058: [NvM] Write protection and erase requests for NvMWriteBlockOnce blocks

  Problem description:

  Name: Delia Batica
  Phone: +40 356 78 4202
  Role: Developer

  Description/Motivation:

  Currently, the behavior of NvM API's for written blocks with NvMWriteBlockOnce configured TRUE that have write protection unset is unclear.

I will refer as wbo the blocks with NvMWriteBlockOnce configured TRUE.

How should the wbo be handled during read requests if the block is reported as inconsistent?
In SWS_NvM_00316 for eg., NvM_ReadBlock sets wbo to write protected if block is valid. (same for SWS_NvM_00314, SWS_NvM_00784).

How should NvM_EraseNvBlock and NvM_InvalidateNvBlock handle a wbo if the write protection is still unset? Should the 2 APIs set the write protection for a block configured with NvMWriteBlockOnce set to TRUE?

NVM072_Conf : says that NVRAM manager".. sets the
write protection bit after the NV block was written the first time. This means that some of the NV blocks in the NVRAM should never be erased ..after first initialization.

Also, it should be made clear that "write protection", "write protection bit", "write protection attribute" all refer to the current known state of write protection of a block.

Agreed solution:

(1) Change the Description for requirement ECUC_NvM_00072 to:
"Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block was already written and it is detected as valid and consistent during a read for it. [NVM276].
true: Defines write protection after first write is enabled.
false: Defines write protection after first write is disabled."

(2) Change the requirement SWS_NvM_00316 to:
"The job of the function NvM_ReadBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(3) Change the requirement SWS_NvM_00314 to:
"The job of the function NvM_ReadAll shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by

NvM_SetBlockProtection."

(4) Change the requirement SWS_NvM_00784 to:
"The job of the function NvM_ReadPRAMBlock shall mark every NVRAM block that has been configured with NVM_WRITE_BLOCK_ONCE (TRUE) as write protected if that block is valid and with consistent data. This write protection cannot be cleared by NvM_SetBlockProtection."

(5) Add a requirement, in chapter "7.2.2.13 NVRAM block write protection", stating the following:
"For a block configured with MVM_WRITE_BLOCK_ONCE (TRUE), NvM shall reject any Write/Erase/Invalidate request made prior to the first read request."

(6) For the above requirement, add the following Rationale:
"In case of a reset, the write protection flag of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE), from the NvM Administrative block, is cleared. In order to reactivate the protection, the block must be read prior to a first Write/Erase/Invalidate request being processed, in order to set the write proctection only for a block that is valid and consistent. The first read request can be done either as a single block request or as part of NvM_ReadAll."

(7) Add a requirement, in chapter "7.3.1 Development errors", stating the following:
"The development error NVM_E_WRITE_ONCE_STATUS_UNKNOWN (0x1A) shall be detectable by the NvM module when a Write/Erase/Invalidate is made for a block with MVM_WRITE_BLOCK_ONCE (TRUE), prior to the first read request made for that block, depending on whether the build version mode is development mode."

(8) Add the following requirements in chapter "7.4 Error Detection":
- If development error detection is enabled for NvM module, the function NvM_WriteBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the function NvM_WritePRAMBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_WriteAll shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when the processing of a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request

was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_EraseNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.
- If development error detection is enabled for NvM module, the job of the function NvM_InvalidateNvBlock shall report the DET error NVM_E_WRITE_ONCE_STATUS_UNKNOWN when a write request is made for a block configured with NVM_WRITE_BLOCK_ONCE (TRUE) for which no read request was made prior to this.

(9) Add a requirement, in chapter "8.1.3.2.4 NvM_EraseNvBlock", after the 00423 requirement, which states the following:
"The job of the function NvM_EraseNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."

(10) Add a requirement, in chapter "8.1.3.2.6 NvM_InvalidateNvBlock", after the 00417 requirement, which states the following:
"The job of the function NvM_InvalidateNvBlock shall leave the write protection unchanged for the blocks configured with MVM_WRITE_BLOCK_ONCE (TRUE)."
–Last change on issue 74058 comment 15–

BW-C-Level:

Application	Specification	Bus
1	4	1