

HYUNDAI AUTOEVER

AUTOSAR NvM User Manual

DOC. NO

SCOPE OF APPLICATION All Project/Engineering
Responsibility: Classic AUTOSAR Team

File Name: NvM_UM.pdf

Creation YJ Yun 2022/03/25

Check JH Jung 2022/03/25

Approval SH Yoo 2022/03/25

Edition Date: 24th 2022/03/25

Document Management System

Any user/Gahyun Kim Classic AUTOSAR Team. This document contains proprietary information of HyundaiAutoEver and is not to be reproduced or duplicated without permission. Any such act could result in restrictions imposed by company rules and related laws.

Document Change Histroy

Date (YYYY-MM-DD)	Ver.	Editor	Chap	Description (before -> after revision)
2016-04-04	1.0	CY Song		• Initial Creation
2016-05-30	1.1	CY Song	Chap 5 Chap 9.2.7 Chap 9.2.8	• Changed format and configuration default values • ReadAll Time updated • WriteAll Time updated
2016-05-31	1.2	CY Song	Chap 4.3 Chap 9.1.2	• Change Log updated • Caution 4) added
2016-06-28	1.3	CY Song	Chap 4.3.1.1	• Change Log updated
2016-07-20	1.4	CY Song	Chap 4.3 Chap 9.6.3	• Change Log deleted • Added cautions when setting callbacks
2016-08-08	1.5	CY Song	Chap 5.1.1 Chap 5.2.1 Chap 9.5	• Modified NvMDatasetSelectionBits, NvMDynamicConfiguration, NvMJobPrioritization items • Modified FeeSetModeSupported items (value) • Summary description added
2016-08-12	1.6	CY Song	Chap 5.1.2 Chap 9.1.3 Chap 9.2.9	• Description of 2)NvMBlockJobPriority Std Block modified • Description of Immediate Block added • Chap 8.2.9 added
2016-09-22	1.7	CY Song	Chap 4.3.1.1 Chap 5.1.1	• Change Log updated • 3) description added
2016-10-14	1.8	CY Song	Chap 4.3.1.1 Chap 5.1.1 Chap 9.2.10	• Change Log updated • Dynamic Configuration Category changed • Virgin Internal EEPROM added during Fee Init
2016-11-02	1.9	CY Song	Chap 4.3.1.1 Chap 9.2.2 Chap 9.6	• Change Log updated • Added calculation method of RH850 Internal EEPROM Size • Added configuration method of Application callbacks
2016-12-06	1.10	CY Song	Chap 4.3.1.1 Chap 5.1.1	• Change Log updated • NvMMultiBlockCallback item modified to changeable
2017-06-30	1.11	YJ Yun	Chap 4.3.1.1 Cahp 9.8	• Change Log updated • Deleted details related to external EEPROM size • Document layout modified
2017-09-27	1.12	YJ Yun	Chap 4.3.1.1	• Change Log updated

2019-01-31	1.13	YJ Yun	Chap 4.3.1.1 Chap 9.2.2 Chap 9.2.10 Chap 9.2.11.2	<ul style="list-style-type: none"> • Change Log updated • Internal EEPROM size modified • Erase all when changing memory layout • Mem_Integration_User added
2019-05-09	1.14	YJ Yun	Chap 8 Chap 9.2.11	<ul style="list-style-type: none"> • Dem event description added • Cautions about garbage collection added • Moved EEPROM calculation formula to Application Guide
2019-03-28	1.15	YJ Yun	Chap 4.3.1.1	<ul style="list-style-type: none"> • Change Log updated
2019-08-20	1.16	YJ Yun	Chap 4.3.1.1	<ul style="list-style-type: none"> • Change Log updated • Appendix NvMMainfunctionTriggerRef added
2019-09-23	1.17	YJ Yun	Chap 4.3.1.1 Chap 6.3.7 Chap 5.1.1	<ul style="list-style-type: none"> • Change Log updated • Cddlf added • Description of NvMUserJobFunction added
2019-10-17	1.18	YJ Yun	Chap 5 Chap 4.3.1.1	<ul style="list-style-type: none"> • Configuration category modified • Change Log updated
2020-12-31	1.5.1.0	YJ Yun	Chap 5 Chap 4.3.1.1	<ul style="list-style-type: none"> • Configuration category modified • Change Log updated
2021-01-20	1.5.2.0	YJ Yun	Chap 4.3.1.1	<ul style="list-style-type: none"> • Change Log updated
2021-02-01	1.5.3.0	YJ Yun	Chap 4.3.1.1	<ul style="list-style-type: none"> • Change Log updated
2021-12-30	1.5.4.0	JH Lim	Chap 4.3.1.1	<ul style="list-style-type: none"> • Change Log updated
2022-03-25	1.6.0.0	YJ Yun	Chap 4.3.1.1 Chap 5.2.5 Chap 7.2.1.1 Chap 6.3.2.6	<ul style="list-style-type: none"> • Generator validation message added/ deleted • ERR020076, ERR020078, ERR020079 added • WRN020052, WRN020054 deleted • Change Log updated • Fee configuration guide added • Description of SetRamBlockStatus function added
2022-06-15	1.6.1.0	SH Park	Chap 4.3.1.1 Chap 5.1.1 Chap 5.1.2 Chap 7.2.1.1 Chap 9.1.5	<ul style="list-style-type: none"> • Change log updated • Added ReadAll/WriteAll Order • Generator validation message updated • ERR020068, ERR020069, ERR020070 수정 • ERR020080 added • Added ReadAll/WriteAll Order configuration example
2022-08-19	1.6.2.0	YJ Yun	Chap 4.3.1.1	<ul style="list-style-type: none"> • Change log updated
2022-12-01	1.6.2.1	YJ Yun	Chap 4.3.1.1	<ul style="list-style-type: none"> • Change log updated
2022-12-20	1.7.0.0	SH Park	Chap 4.3.1.1 Chap 5.1.1 Chap 6.3.7.3	<ul style="list-style-type: none"> • Change log updated • Added NvMUserWdgToggleFunction at Common Container • Added NvMUserWdgToggleFunction description

Table of Contents

1.	OVERVIEW	- 7 -
2.	REFERENCE	- 7 -
3.	AUTOSAR SYSTEM.....	- 8 -
3.1	Overview of Software Layers	- 8 -
3.2	AUTOSAR Memory Stack	- 8 -
4.	PRODUCT RELEASE NOTES.....	- 10 -
4.1	Overview	- 10 -
4.2	Scope of the release.....	- 10 -
4.3	Module release notes	- 10 -
4.3.1	NvM.....	- 10 -
4.3.1.1	Change Log	- 10 -
4.3.1.2	Limitations	- 20 -
4.3.1.3	Deviation.....	- 20 -
5.	CONFIGURATION GUIDE	- 22 -
5.1	NvM Module.....	- 22 -
5.1.1	Common Container	- 22 -
5.1.2	NvMBlockDescriptor Configuration.....	- 24 -
5.1.3	NvmDemEventParameterRefs Configuration.....	- 27 -
5.2	FEE Module (Internal EEPROM)	- 27 -
5.2.1	FeeGeneral Container	- 27 -
5.2.2	BlockConfiguration Configuration	- 27 -
5.2.3	FeelfxSpecificConfig (Only Aurix).....	- 28 -
5.2.4	Fee Information	- 28 -
5.2.5	Redundant Block Distributed Arrangement (Only Chorus/Bolero).....	- 28 -
5.3	FLS Module (Internal EEPROM)	- 29 -
5.3.1	FlsGeneral Container	- 29 -
5.3.2	FlsConfigSet	- 29 -
5.3.3	FlsDemEventParameterRefs	- 29 -
5.3.4	FlsSectorList.....	- 30 -
5.4	System Configuration	- 31 -
5.4.1	ApplicationSwComponentType Configuration	- 31 -

5.4.2	CompositionSwComponentType Configuration	- 32 -
6.	APPLICATION PROGRAMMING INTERFACE (API)	- 33 -
6.1	Type Definitions.....	- 33 -
6.1.1	NvM_RequestResultType	- 33 -
6.2	Macro Constants	- 34 -
6.3	Functions.....	- 34 -
6.3.1	Initialization.....	- 34 -
6.3.2	Synchronous Requests	- 35 -
6.3.2.1	NvM_SetDataIndex.....	- 35 -
6.3.2.2	NvM_GetDataIndex	- 36 -
6.3.2.3	NvM_SetBlockProtection	- 37 -
6.3.2.4	NvM_GetErrorStatus	- 38 -
6.3.2.5	NvM_GetVersionInfo	- 38 -
6.3.2.6	NvM_SetRamBlockStatus.....	- 39 -
6.3.2.7	NvM_SetBlockLockStatus	- 40 -
6.3.3	Asynchronous Single Block Requests.....	- 41 -
6.3.3.1	NvM_ReadBlock	- 41 -
6.3.3.2	NvM_WriteBlock	- 42 -
6.3.3.3	NvM_RestoreBlockDefaults	- 42 -
6.3.3.4	NvM_EraseNvBlock	- 43 -
6.3.3.5	NvM_CancelWriteAll	- 44 -
6.3.3.6	NvM_InvalidateNvBlock.....	- 44 -
6.3.3.7	NvM_CancelJobs	- 45 -
6.3.4	Asynchronous Multi Block Requests	- 46 -
6.3.4.1	NvM_ReadAll	- 46 -
6.3.4.2	NvM_WriteAll.....	- 46 -
6.3.5	Callback Notifications.....	- 47 -
6.3.5.1	NvM_JobEndNotification	- 47 -
6.3.5.2	NvM_JobErrorNotification	- 47 -
6.3.6	Scheduled Functions	- 48 -
6.3.6.1	NvM_Mainfunction	- 48 -
6.3.7	CddIf.....	- 48 -
6.3.7.1	NvM_CddGetStatus	- 49 -
6.3.7.2	NvM_UserJobFunction.....	- 49 -
6.3.8	Notes.....	- 51 -
6.3.8.1	In Communication with application SW-C.....	- 51 -
7.	GENERATOR.....	- 52 -
7.1	Generator Option.....	- 52 -
7.1.1	NvM.....	- 52 -
7.2	Generator Error Message	- 52 -
7.2.1	NvM.....	- 52 -
7.2.1.1	Error Messages	- 52 -
7.2.1.2	Warning Messages	- 62 -

7.2.1.3	Information Messages	- 63 -
8.	SWP ERROR CODE	- 64 -
8.1	SWP Error Code List	- 64 -
8.1.1	NVM_E_INTEGRITY_FAILED	- 64 -
8.1.2	NVM_E_LOSS_OF_REDUNDANCY	- 65 -
8.1.3	NVM_E_QUEUE_OVERFLOW	- 65 -
8.1.4	NVM_E_REQ_FAILED	- 66 -
8.1.5	NVM_E_VERIFY_FAILED	- 66 -
8.1.6	NVM_E_WRITE_PROTECTED	- 67 -
8.1.7	NVM_E_WRONG_BLOCK_ID	- 67 -
9.	APPENDIX.....	- 68 -
9.1	Function-specific Configuration Guide.....	- 68 -
9.1.1	Redundant Block Configuration (2 copies).....	- 68 -
9.1.2	CRC Implement	- 68 -
9.1.3	Immediate Block Configuration	- 68 -
9.1.4	ReadAll / WriteAll Configuration.....	- 69 -
9.2	Cautions during Design	- 70 -
9.2.1	NvM Block Identifier	- 70 -
9.2.2	RamBlock Length	- 70 -
9.2.3	Immediate Block	- 70 -
9.2.4	Request Return Value and Checking Block Status.....	- 70 -
9.2.5	ReadAll Time.....	- 70 -
9.2.6	WriteAll Time.....	- 70 -
9.2.7	NvM API Call-Context.....	- 71 -
9.2.8	Fee Init of Virgin Internal EEPROM.....	- 71 -
9.2.9	Erase All When Changing Memory Layout	- 71 -
9.2.10	Mem_Integration_User Implementation	- 71 -
9.2.10.1	MEM_WRITEALL_FAST_MODE	- 71 -
9.2.10.2	User Callbacks	- 72 -
9.2.11	Garbage Collection.....	- 72 -
9.2.12	NvMMainfunctionTriggerRef.....	- 73 -
9.3	Bswmd (Bsw Module Description)	- 74 -
9.3.1	MainFunction Period Configuration	- 74 -
9.3.2	Bsw Module Version Setting.....	- 74 -
9.4	Exclusive Areas	- 74 -
9.4.1	Module-specific SchM Apis	- 74 -
9.4.2	Configuration Method	- 76 -
9.5	Normal and extended runtime preparation of NVRAM blocks	- 77 -
9.6	Notification Interface with Application	- 77 -

9.6.1	SingleBlock Callback.....	- 77 -
9.6.2	InitBlock Callback.....	- 79 -
9.7	Edit Cases following Port Name Change.....	- 80 -
9.7.1	InitBlock, SingleBlock Callback Name Edit.....	- 80 -
9.7.2	Generate.py Edit	- 80 -
9.7.3	Port Connection Edit under EcucValueCollection Tap	- 80 -
9.8	PIM (PerInstanceMemory) Configuration Method	- 81 -
9.8.1	ArTypedPerInstanceMemory Addition	- 81 -
9.8.2	In case of using ReadAll/WriteAll with PIM (PerInstanceMemory)	- 82 -

1. Overview

This document is created based on the standardized Autosar SRS/SWS. See the reference documents below for more details on the functionalities of modules. And for detailed configuration of Fee and FIs, see user manuals and integration manuals that each MCU support.

Each configuration category is defined as follows.

- Changeable (C): Items that can be configured by users
- Fixed (F): Items that cannot be changed by users
- NotSupported (N): Items that are not used

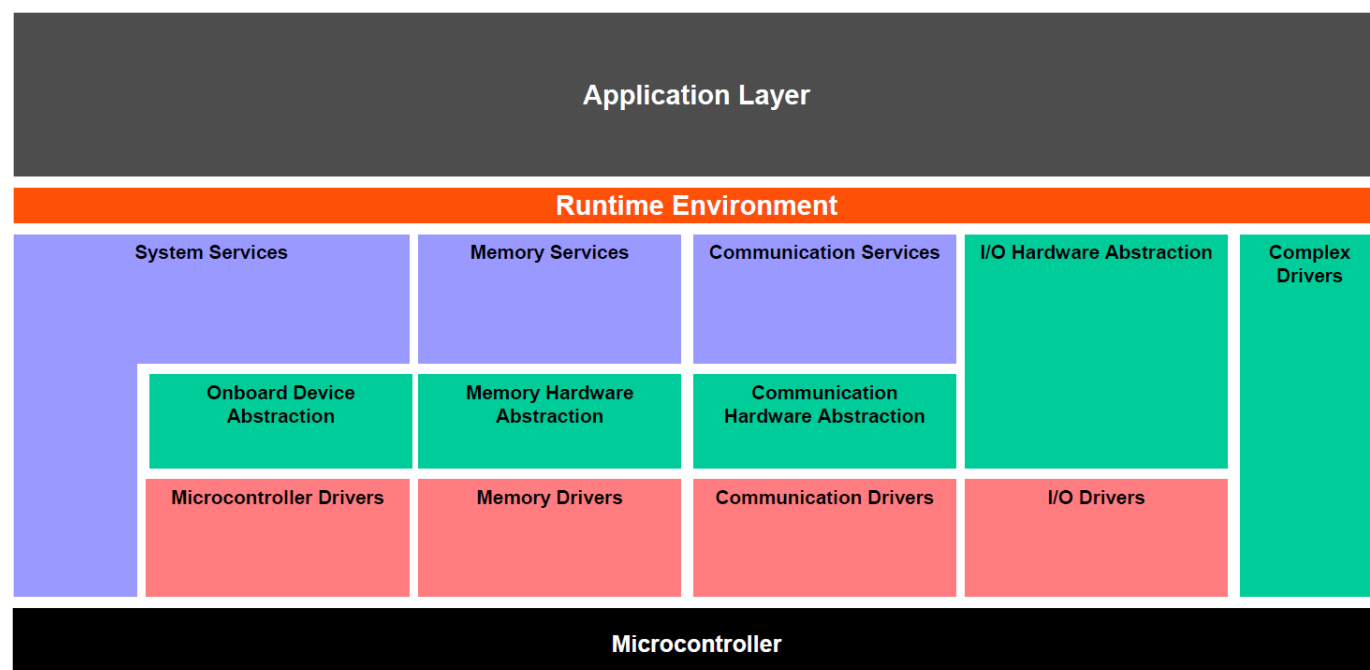
2. Reference

Sl. No.	Title	Version
1.	AUTOSAR BSW Service API Guide.doc	1.0.0 or later
2.	AUTOSAR_SWS_NVRAMManager.pdf	3.2.0
3.	AUTOSAR_SWS_EEPROMAbstraction.pdf	2.0.0
4.	AUTOSAR_SWS_EEPROMDriver.pdf	3.2.0
5.	AUTOSAR_SWS_FlashDriver.pdf	3.2.0
6.	AUTOSAR_SWS_FlashEEPROMEmulation.pdf	2.0.0
7.	AUTOSAR_SWS_CRCLibrary.pdf	4.2.0
8.	AUTOSAR_SWS_MemoryAbstractionInterface.pdf	1.4.0

3. AUTOSAR System

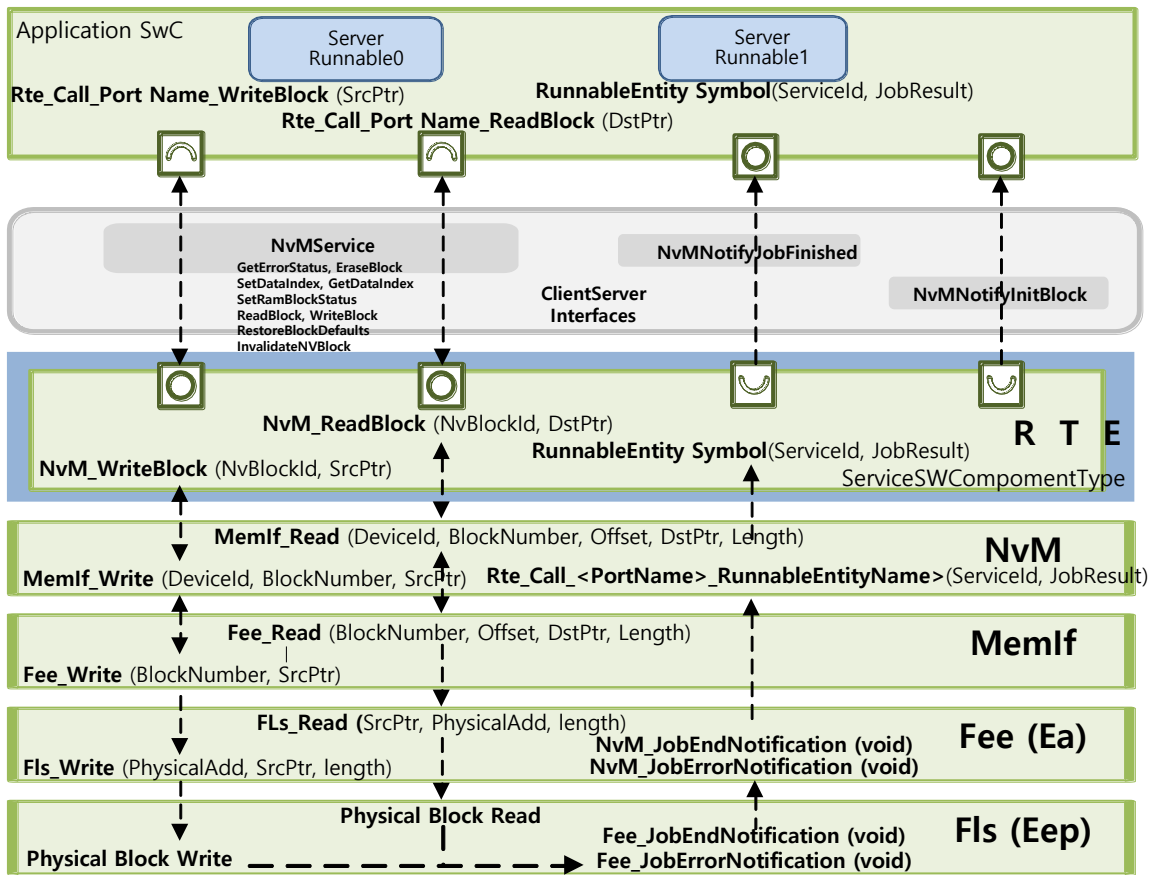
3.1 Overview of Software Layers

The AUTOSAR platform is a layered architecture as illustrated below. The AUTOSAR platform can be divided into Service Layer, ECU Abstraction Layer, Complex Device Drivers and Microcontroller Abstraction Layer.



3.2 AUTOSAR Memory Stack

Memory Stack modules consists of NvM, MemIf, Ea, Eep, Fee and Fls modules. Fee and Fls modules are part of Mcal, and see each MCU manufacturer's user manuals and integration manuals for basic information. Basically NvM and MemIf modules are necessary to use EEPROM, and Fee and Fls modules are additionally required for internal EEPROM while Ea and Eep modules are required for external EEPROM. The Autosar layers and the interface between modules for using EEPROM are shown below.



Write requests (From NvM_Sws_NvM698)

Applications have to adhere to the following rules during write request for implicit synchronization between application and NVRAM manager:

1. The application fills a RAM block with the data that has to be written by the NvM module
2. The application issues the NvM_WriteBlock request which transfers control to the NvM module.
3. From now on the application must not modify the RAM block until success or failure of the request is signaled or derived via polling. In the meantime the contents of the RAM block may be read.
4. An application can use polling to get the status of the request or can be informed via a callback function asynchronously.
5. After completion of the NvM module operation, the RAM block is reusable for modifications.

Read requests (From NvM_Sws_NvM699)

Applications have to adhere to the following rules during read request for implicit synchronization between application and NVRAM manager:

1. The application provides a RAM block that has to be filled with NVRAM data from the NvM module's side.
2. The application issues the NvM_ReadBlock request which transfers control to the NvM module.
3. From now on the application must not read or write to the RAM block until success or failure of the request is signaled or derived via polling.
4. An application can use polling to get the status of the request or can be informed via a callback function.
5. After completion of the NvM module operation, the RAM block is available with new data for use by the application.

4. Product Release Notes

4.1 Overview

This chapter provides the release information on NvM Product, describing the features and restrictions of different versions of the NvM software.

4.2 Scope of the release

All content in this document is limited to the following NvM module.

Module	Autosar version	SWS version	Module version
NvM	4.0.3	3.2.0	1.7.0

※ Module version refers to the SW version of the BswModule Description (Bswmd) file of each module.

4.3 Module release notes

4.3.1 NvM

4.3.1.1 Change Log

➤ Version 1.7.0.0 (2022-12-20)

- New feature

■ Add NvMUserWdgToggleFunction Callout when memory initialization

Rationale		When External Watchdog is used, reset may occur when the memory size is large or the execution time of the initialization logic is long during memory initialization. To prevent this case, add User Watchdog Toggle callout so that the user can trigger the External Watchdog at an appropriate period.
Impact behavior	on	If the user configures NvMUserWdgToggleFunction, user functions are repeatedly performed in mem_EalnitPerform and Mem_FeelnitPerform until initialization is completed.
Impact settings	on	NvMCommon/NvMUserWdgToggleFunction
Required ASW actions		When the user configured the NvMUserWdgToggleFunction, user should implement the user function.

※ This function should be used only when Reset occurs before memory initialization is completed because External Watchdog is used and Timer configuration time is short. In general, it is recommended to use the Watchdog function provided by the platform.

- Improvement

■ N/A

➤ Version 1.6.2.1 (2022-12-01)

- Task

- Add User Manual

Rationale	Add the user manual English version
Impact on behavior	None
Impact on settings	None
Required ASW actions	None

- Version 1.6.2.0 (2022-08-20)

- New feature

- N/A

- Improvement

- Improved security coding to comply with the UNECE Cyber Security regulations

Rationale	To improve security coding to comply with the UNECE Cyber Security regulations
Impact on behavior	None
Impact on settings	None
Required ASW actions	None

- Version 1.6.1.0 (2022-06-15)

- New feature

- Added the functionality to configure the order in which blocks are processed during ReadAll/WriteAll operation

Rationale	Request a function that allows the user to determine the processing order of blocks when processing ReadAll/WriteAll
Impact on behavior	If the function is enabled, the blocks are processed in the order defined by the user
Impact on settings	NvMReadAllOrderSupport, NvMWriteAllOrderSupport, NvMReadAllOrder, NvMWriteAllOrder
Required ASW actions	Refer to 5.1.1 Common Container, 5.1.2 NvMBlockDescriptor

- Improvement

- N/A

- Version 1.6.0.0 (2022-03-28)

- New feature

- N/A

- Improvements

- Added a Fee configuration guide

Rationale		Needed to reduce user technical review time
Impact on behavior	on	Applied technical review items to UM Added a Bolero/Chorus MCAL Fee configuration guide
Impact on settings	on	None
Required ASW actions		See 5.2.5 Redundant Block distributed arrangement (Only Chorus/Bolero)

- Enhanced Generator Validation Rule

Rationale		Needed to reduce user technical review time
Impact on behavior	on	Applied technical review items to Generator Validation Rule 1. The parameters when applying CRC 2. The parameter NvMRamBlockDataAddress when using ReadAll/WriteAll
Impact on settings	on	None
Required ASW actions		None

- Improved the function SetRamBlockStatus

Rationale		When the parameter NvMSetRamBlockStatusApi is set as TRUE according to AUTOSAR 4.0.3, WriteBlock runs only when SetRamBlockStatus (TRUE) is invoked. In case of VALID/UNCHANGED, only OK is returned without executing WriteBlock.
Impact on behavior	on	Applied AUTOSAR 4.3.0 and modified the module so that WriteBlock is executed regardless of SetRamBlockStatus callout when NvMSetRamBlockStatusApi is set as TRUE.
Impact on settings	on	None
Required ASW actions		6.3.2.6 See NvM_SetRamBlockStatus.

➤ Version 1.5.4.0 (2021-12-30)

- New feature

- N/A

- Improvement

- Improved security coding to comply with the UNECE Cyber Security regulations

Rationale	To improve security coding to comply with the UNECE Cyber Security regulations
Impact behavior on	None
Impact settings on	None
Required ASW actions	None

➤ Version 1.5.3.0 (2021-02-01)

- New feature
 - N/A
- Improvement
 - Modified return value of redundant blocks

Rationale	To set the standard of return value when failing to read original and copy blocks of redundant blocks * Non-AUTOSAR Specification
Impact behavior on	When failing to read original and copy blocks of redundant blocks, the job result of the original block (first block) is returned to the return value of the relevant block. If one of the two blocks is virgin, the job result of the other block is returned.
Impact settings on	None
Required ASW actions	None

➤ Version 1.5.2.0 (2021-01-20)

- New feature
 - N/A
- Improvement
 - Applied MISRA Rule

Rationale	Applied MISRA Rule
Impact behavior on	None
Impact settings on	None
Required ASW actions	None

➤ Version 1.5.1.0 (2020-12-31)

- New feature

- N/A

- Improvements

- Applied MISRA Rule

Rationale	Applied MISRA Rule
Impact behavior on	None
Impact settings on	None
Required ASW actions	None

- Changed conditions for Dem Event occurrence

Rationale	Dem Event NVM_E_LOSS_OF_REDUNDANCY occurs when redundant blocks are all virgin (invalidated).
Impact behavior on	Modified so that Dem Event NVM_E_LOSS_OF_REDUNDANCY does not occur when redundant blocks are all virgin (invalidated). Changed to block unnecessary Dem Event because it is not written once
Impact settings on	None
Required ASW actions	None

➤ Version 1.5.0.0 (2019-10-17)

- New feature

- N/A

- Improvement

- Modified Parameter Definition File Category

Rationale	To make source code available
Impact behavior on	None
Impact settings on	Change permissions only for configurations that can be set by the user
Required ASW actions	Users can modify the configurations that the category has changed to changeable.

➤ Version 1.5.0 (2019-09-26)

- New features

- Added NvM_CddGetStatus

Rationale	To check memory stack state
Impact behavior on	None

Impact settings on	None
Required ASW actions	It is a function used when there is a need to check the state of BSW memory modules.

■ Added NvMUserJobFunction callouts

Rationale	To add user job callouts
Impact behavior on	None
Impact settings on	NvMCommon/NvMUserJobFunction
Required ASW actions	When users set NvMUserJobFunction, UserJob function is driven in OsTask_BSW_Mem_Process Task.

- Improvement

■ N/A

➤ Version 1.4.2 (2019-08-30)

- New features

■ N/A

- Improvements

■ Changed multiplicity of NvMMMainfunctionTriggerRef to 0..1

Rationale	To Change multiplicity of NvMMMainfunctionTriggerRef to 0..1
Impact behavior on	None
Impact settings on	NvMMMainfunctionTriggerRef
Required ASW actions	Can delete the configuration of NvMMMainfunctionTriggerRef if using alarms is unnecessary. A change request is required, and if not configured, the SetEvent API must be invoked to drive the memory stack main function. See Appendix.

■ Issue of changing generation code without configuration change

Rationale	Even without configuration change, the location of include keyword of the generation code NvM_Cfg.c is changed.
Impact behavior on	None
Impact settings on	None

Required ASW actions	None
----------------------	------

➤ Version 1.4.1 (2019-04-04)

- New features

■ N/A

- Improvements

■ Modified Write Verification error

Rationale	At the WriteAll stage, NvM blocks whose Write Verification is set as true write and then read again to compare two values. This is to verify that it is normally written. If Read fails, WriteAll is interrupted. As a result, the shutdown/Reset sequence stops in WriteAll stage.
Impact on behavior	None
Impact on settings	None
Required ASW actions	None

■ Revised manual user guide (Appendix)

Rationale	- . To add cautions about garbage collection - . To move DFlash size calculation formula to SAG document
Impact on behavior	None
Impact on settings	None
Required ASW actions	Aurix : review of MCAL FIs configuration on the basis of cautions

➤ Version 1.4.0 (2018-01-31)

- New features

■ N/A

- Improvements

■ Reinforced cautions when designing

Rationale	To add Internal EEPROM size calculation formula of chorus MCU. Measures to take when changing memory layout. To add a description of Mem_Integration_User Code that
-----------	---

		requires user implementation
Impact on behavior	on	None
Impact on settings	on	None
Required ASW actions		See cautions when designing Appendix.

■ [Aurix TCxx] Applied SPI DMA restrictions

Rationale		When using DMA in SPI, the following restriction exists: <i>“DMA implements a circular buffer with a maximum width of 32KB. So the source/destination address should not cross the 32KB boundary if sequential data are to be transferred. This address alignment is checked. ((address + length) & 0x00007FFF) <= 0x00008000)”</i> For this reason, NVM_RAM_VAR_CLEARED_ALIGNED_ADDR_UNSPECIFIED memory section is added.
Impact on behavior	on	None
Impact on settings	on	None
Required ASW actions		None

■ Task optimization

Rationale		Simplified the configuration by changing the previous structure triggered by GPT to Alarm. Optimized the implementation speed through a simple task structure
Impact on behavior	on	When using GPT, NvM_Mainfunction is not allocated at 5ms.
Impact on settings	on	None
Required ASW actions		None

■ Prevented the creation of port of NvBlock for Dem

Rationale		A port is created at NvM sw-component when adding a block for Dem, which requires additional configuration at RTE. Since this is unnecessary, the module was revised so that ports are not created for Dem blocks.
Impact on behavior	on	None
Impact on settings	on	None

settings	
Required ASW actions	None

■ Corrected Rom Block errors

Rationale	<p>Preconditions:</p> <ol style="list-style-type: none"> 1. NvMRomBlockDataAddress is set. 2. NvMNvBlockLength > 32 <p>In case of failing to read blocks that meet the preconditions, only 32 bytes of Rom Data are used as default value while other bytes are not copied. This was fixed.</p>
Impact on behavior	None
Impact on settings	None
Required ASW actions	None

➤ Version 1.3.5 (2017-06-30)

- New features

■ N/A

- Improvements

■ Removed compile warnings

Rationale	To remove compile warnings
Impact on behavior	None
Impact on settings	None
Required ASW actions	None

■ Adjusted the size of Read/Write temporary buffers

Rationale	Increased the buffer size by 8 bytes to share buffers with Ea module. No functional correction
Impact on behavior	None
Impact on settings	None
Required ASW actions	None

➤ 1.3.4 (2016-12-06)

- New features
 - N/A
- Improvement
 - Improved features so that MultiBlockCallback can be used at CDD

Rationale	Request to use MultiBlockCallback on CDD
Impact on behavior	None
Impact on settings	None
Required ASW actions	None

➤ Version 1.3.3 (2016-11-02)

- New features
 - N/A
- Improvement
 - Added RH850 Internal EEPROM Size calculation method and Callback Task configuration method to User Manual.
 Rationale Added cautions when setting up
 Impact on behavior None
 Impact on settings None
 Required ASW actions See updates (Chap 8.6)
 - Improved problems that occur when using ImmediateBlock
 Rationale When requesting ImmediateBlock Write during Redundant block Write, it is cancelled.
 Then malfunction may occur.
 Impact on behavior None
 Impact on settings None
 Required ASW actions None

➤ Version 1.3.2 (2016-10-17)

- New features
 - N/A

- Improvement

- Improved dependence on Library

- Rationale To improve the convenience of user configuration

- Impact on behavior None

- Impact on settings Setting ReadAll/WriteAll option of NvMBlock_ConfigID (BlockID #1) as True

- Required ASW actions Same as above

- Version 1.3.1 (2016-09-22)

- New features

- N/A

- Improvement

- Improved Queue Clear time

- Rationale In case of setting Queue Size as 255, there is time delay when clearing Queue.

- Impact on behavior None

- Impact on settings None

- Required ASW actions None

4.3.1.2 Limitations

- The same task shall be mapped to prevent preemption between the main functions of NvM, Fe, Fls, Ea, and Eep

4.3.1.3 Deviation

- The VARIANT-LINK-TIME is not supported. (Requirement NVM727)
- Explicit Sync Mechanism feature is not supported. (SWS - Chap 7.2.2.17)
 - Sync mechanism feature can be set at the NvM module, and this feature is used when communicating with NvSWC which was added in version 4.0. (Communication is carried out using NvData Interface with Mirror Buffer in place at Ram, and data of this buffer is read and written at NvM.)
- AUTOSAR Debugging feature is not supported. (SWS - Chap 7.7)

- When creating Swcd, PortName is generated according to AUTOSAR_SWS_NVRAMManager 4.2.2 specifications.

- The module was revised so that NvM_WriteBlock is normally performed even when RamBlockStatus is VALID/UNCHANGED according to AUTOSAR_SWS_NVRAMManager 4.3.0 specification.

5. Configuration Guide

5.1 NvM Module

5.1.1 Common Container

See the following configuration.

Parameter Name	Value	Category
NvMApiConfigClass	User Defined	C
NvMBswMMultiBlockJobStatus Information	True	F
NvMCompiledConfigId ⁵⁾	51	C
NvMCrcNumOfBytes	16	N
NvMDatasetSelectionBits ¹⁾	User Defined (From SRS)	C
NvMDevErrorDetect	True	C
NvMDrvModeSwitch	true	F
NvMDynamicConfiguration ⁵⁾	User Defined	C
NvMJobPrioritization ²⁾	User Defined (From SRS)	C
NvMMultiBlockCallback ⁶⁾	BswM_NvM_CurrentJobMode	C
NvMPollingMode	False	N
NvMRepeatMirrorOperations	0	N
NvMSetRamBlockStatusApi	False	C
NvMSizeImmediateJobQueue ³⁾	User Defined	C
NvMSizeStandardJobQueue ³⁾	User Defined	C
NvMVersionInfoApi	User Defined	C
NvMMainFunctionCallCycle	N/A	N
NvMMainfunctionTriggerRef	User Defined	C
NvMUserIncludeFiles ⁴⁾ (Vendor specific)	User Defined	C
NvMUserJobFunction ⁷⁾ (Vendor specific)	User Defined	C
NvMReadAllOrderSupport ⁸⁾	User Defined	C
NvMWriteAllOrderSupport ⁸⁾	User Defined	C
NvMUserWdgToggleFunction ⁹⁾	User Defined	C

1) If DataSet Type Block Usage of SRS is No: 1

If DataSet Type Block Usage of SRS is Yes: It varies depending on the maximum value among NvBlockNum values of DataSet Type Blocks, and the number of NvBlockNum must be designed at Application.

The maximum number of NvBlocks (Fee/Ea Block) that can be set at DataSet Type Blocks varies depending on the value of NvMDatasetSelectionBits. (For all DataSet Type Blocks, NvBlocks can be set up to 2^n .) n: NvMDatasetSelectionBits)

- Ex) If this value is 2, the maximum number of NvBlocks that all DataSet Blocks can have is 4.
- Ex) If this value is 3, the maximum number of NvBlocks that DataSet Blocks can have is 8.

The value of BlockNumber of all Fee/Ea Blocks differs depending on this setting value.
(Chap 5.2.2)

Basic formula: $2^n * \text{NvMNvBlockBaseNumber} + \text{Index}$

(n : NvMDatasetSelectionBits, Index : sequential starting from 0, as many as the number of NvBlock (Fee/Ea Block))

Ex) The configuration parameter NvMDatasetSelectionBits is configured to be 2.

- For a native NVRAM block with NvMNvBlockBaseNumber = 2:
 - NV block is accessed with FEE/EA_BLOCK_NUMBER = 8
 - For a redundant NVRAM block with NvMNvBlockBaseNumber = 3:
 - 1st NV block with data index 0 is accessed with FEE/EA_BLOCK_NUMBER = 12
 - 2nd NV block with data index 1 is accessed with FEE/EA_BLOCK_NUMBER = 13
 - For a dataset NVRAM block with NvMNvBlockBaseNumber = 4, NvMNvBlockNum = 3:
 - NV block #0 with data index 0 is accessed with FEE/EA_BLOCK_NUMBER = 16
 - NV block #1 with data index 1 is accessed with FEE/EA_BLOCK_NUMBER = 17
 - NV block #2 with data index 2 is accessed with FEE/EA_BLOCK_NUMBER = 18
- 2) If Immediate Block Usage of SRS is Yes (when using Immediate Block at Application), set NvMJobPrioritization items as True.
- If Immediate Block is used, the NvM module shall use two queues, one for immediate write jobs (crash data) another for all other jobs (including immediate read/erase jobs, standard jobs). Otherwise the NvM Module shall use one queue and processes all jobs in FCFS order.
- 3) NvMSizeImmediateJobQueue / NvMSizeStandardJobQueue
- The number of saved requests (Read/Write jobs, etc.) that are requested by NvM Manger
 - QueueOverFlow Dem Error occurs if requests (Read/Write) are stacked in the queue surpassing the set value. In this case, the value must be increased.
 - Each number of set blocks for Immediate Block and Standard Block is set at Application.
- 4) If each block sets RamBlockDataAddress/RomBlockDataAddress, NvM created files refer to the address, and a compiler error occurs because the address (buffer) is not declared as extern. This is why a header file in which the address (buffer) is declared as extern must be included in NvM when setting RamBlockDataAddress/RomBlockDataAddress, and the Header File Name must be added to NvMUserIncludeFiles.
- 5) Set according to DynamicConfiguration Usage items of SRS Information.

Note: If there is no relevant item in SRS, the relevant MCU does not support DynamicConfiguration.

DynamicConfiguration can be explained as follows:

The job of the function NvM_ReadAll shall process an extended runtime preparation for all blocks which are configured with NvMResistantToChangedSw == FALSE and NvMDynamicConfiguration == TRUE and configuration ID mismatch occurs.

The job of the function NvM_ReadAll shall process the normal runtime preparation of all NVRAM blocks when they are configured with NvMResistantToChangedSw == TRUE and NvMDynamicConfiguration == TRUE and if a configuration ID mismatch occurs.

The job of the function NvM_ReadAll shall update the configuration ID from the RAM block assigned to the reserved NVRAM block with ID 1 according to the new (compiled) configuration ID, mark the NVRAM block to be written during NvM_WriteAll and request a CRC recalculation if a configuration ID mismatch occurs and if the NVRAM block is configured with NvMDynamicConfiguration == TRUE. (See 8.5 about extended/normal runtime)

- 6) When job is finished after MultiBlock Request (ex, ReadAll) is invoked (ex. ReadAll completion), configured callback is invoked.
- 7) The configured NvMUserJobFunction is invoked in the Task OsTask_BSW_Mem_Process.
- 8) It is a function that sets the block order processed by the user during the ReadAll/WriteAll operation, and operates based on the AUTOSAR Specification when disabled.
- 9) If the user configures NvMUserWdgToggleFunction, user functions are repeatedly performed in mem_EalnitPerform and Mem_FealnitPerform until initialization is completed.
This function should be used only when Reset occurs before memory initialization is completed because External Watchdog is used and Timer configuration time is short. In general, it is recommended to use the Watchdog function provided by the platform.

5.1.2 NvMBlockDescriptor Configuration

See the following configuration.

Parameter Name	Value	Category
NvMBlockCrcType ¹⁾	User Defined	C
NvMBlockJobPriority ²⁾	User Defined	C
NvMBlockManagementType ³⁾	User Defined	C
NvMBlockUseCrc ¹⁾	User Defined	C
NvMBlockUseSyncMechanism	False	N
NvMBlockWriteProt	False(User Defined)	C
NvMBswMBlockStatusInformation	False(User Defined)	C
NvMCalcRamBlockCrc ¹⁾	User Defined	C
NvMInitBlockCallback ¹⁴⁾	User Defined	C
NvMMaxNumOfReadRetries	1	C
NvMMaxNumOfWriteRetries	1	C
NvMNvBlockBaseNumber ⁴⁾	User Defined	C
NvMNvBlockLength ⁵⁾	User Defined	C
NvMNvBlockNum ⁶⁾	User Defined	C

Parameter Name	Value	Category
NvMNvramBlockIdentifier ⁷⁾	User Defined	C
NvMNvramDeviceId ⁸⁾	User Defined	C
NvMRamBlockDataAddress ⁹⁾	User Defined	C
NvMReadRamBlockFromNvCallback	-	N
NvMResistantToChangedSw ¹⁵⁾	User Defined	C
NvMRomBlockDataAddress	-	C
NvMRomBlockNum ¹⁶⁾	0	C
NvMSelectBlockForReadAll ¹⁰⁾	User Defined	C
NvMSelectBlockForWriteAll ¹⁰⁾	User Defined	C
NvMSingleBlockCallback ¹¹⁾	User Defined	C
NvMStaticBlockIDCheck	False(User Defined)	C
NvMWriteBlockOnce	False(User Defined)	C
NvMWriteRamBlockToNvCallback	-	N
NvMWriteVerification	False	C
NvMWriteVerificationDataSize	1	C
NvMDefaultRomCRCEnabled (Vendor Specific)	false	N
NvMTargetBlockReference ¹²⁾	User Defined	C
NvMNameOfFeeBlock/ NvMNameOfEaBlock ¹³⁾	User Defined	C
NvMReadAllOrder ¹⁷⁾	User Defined	C
NvMWriteAllOrder ¹⁷⁾	User Defined	C

- 1) When using CRC
 - NvMBlockCrcType : Set as you wish at App.
 - NvMBlockUseCrc, NvMCalcRamBlockCrc : Set as True. (If not using, set as False.)
 - The size of Fee / Ea Block must be set as NvMNvBlockLength + CRC Byte.
- 2) NvMBlockJobPriority
 - In case of Immediate Block: Set as 0.
 - In case of Standard Block: Set a value other than 0 (in consideration of priorities between blocks).
- 3) NvMBlockManagementType
 - NVM_BLOCK_NATIVE : one copy
 - NVM_BLOCK_REDUNDANT : two copy
 - NVM_BLOCK_DATASET : an array of equally sized data blocks.
- 4) NvMNvBlockBaseNumber
 - Set equally as NvMNvramBlockIdentifier.
- 5) NvMNvBlockLength
 - Set according to App design (Must be equal as the length of Fee/Ea blocks linked to submodule blocks).
- 6) NvMNvBlockNum
 - NVM_BLOCK_NATIVE : Set as 1.
 - NVM_BLOCK_REDUNDANT : Set as 2.
 - NVM_BLOCK_DATASET : Set according to App design.
- 7) NvMNvramBlockIdentifier
 - Set as sequential as other Block ID. (From Autosar specifications)
- 8) NvMNvramDeviceId

- Device ID of linked submodule (Fls/Eep)
 - Set as 0 in case of using only one In / Ex EEPROM.
 - In case of using In / Ex EEPROM at the same time, set as 0 for blocks saved in Internal EEPROM and 1 for blocks saved in External EEPROM.
- 9) NvMRamBlockDataAddress
- Setting is a must when using ReadAll / WriteAll.
 - Read NvBlock value for RamBlock set in ReadAll and write RamBlock value set in WriteAll at NvBlock.
 - A file declared as extern must be added to UserIncludeFile of NvM Common Container so that set RamBlock can be included in NvM_Cfg.c.
 - As the NvM module is updated only with the set start address and length of RamBlock, ram areas of other variables can be invaded if the length of RamBlock is shorter than that of NvBlock. Therefore the length of both RamBlock and NvBlock must be equal.
- 10) NvMSelectBlockForReadAll / NvMSelectBlockForWriteAll
- ReadAll : A value saved in EEPROM is read to Ram during StartUp.
 - WriteAll : Ram value is written in EEPROM during ShutDown.
- When using WriteAll, the measurements of WriteAll Time must be applied to the time taken from WdgDelnit to WdgReset.
- 11) NvMSingleBlockCallback
- Set as "Rte_Call_NvM_PNJF_{Block}_JobFinished" when using (see Chap 8.7).
Block = {ecuc(NvM/NvMBlockDescriptor.SHORT-NAME)}
- 12) NvMTargetBlockReference
- Select Fee in case of blocks for Internal EEPROM.
 - Select Ea in case of blocks for External EEPROM.
- 13) NvMNameOfFeeBlock/ NvMNameOfEaBlock
- After generating Fee / Ea blocks in line with NvM configuration, link with the first block.
- 14) NvMInitBlockCallback
- If InitBlockCallback is set in case of Read Fail, call out Callback. In Application, handle Ram Block value in line with design intention within the callback. Assuming that RamBlock is handled in Application, NVM sets Block as NVM_REQ_OK after InitBlockCallback is called out.
 - Set as "Rte_Call_NvM_PNIB_{Block}_InitBlock" when using (see Chap 8.7).
Block = {ecuc(NvM/NvMBlockDescriptor.SHORT-NAME)}
- 15) NvMResistantToChangedSw
- Meaningful when NvMDynamicConfiguration is set as TRUE.
 - The job of the function NvM_ReadAll shall process an extended runtime preparation for all blocks which are configured with NvMResistantToChangedSw == FALSE and NvMDynamicConfiguration == TRUE and configuration ID mismatch occurs.
 - The job of the function NvM_ReadAll shall process the normal runtime preparation of all NVRAM blocks when they are configured with NvMResistantToChangedSw == TRUE and NvMDynamicConfiguration == TRUE and if a configuration ID mismatch occurs.
- 16) NvMRomBlockNum
- Always set as 0 if RomBlock is not set.
 - NvMReadAllOrder / NvMWriteAllOrder
 - In the case of blocks that do not require an order setting, it is not necessary to set an order.
 - For blocks that require ordering, Order must start with <1> and be set to sequential with other Orders.

- If the Block ID is <0> or <1>, Order shall not be set.

5.1.3 NvmDemEventParameterRefs Configuration

See the following configuration.

Parameter Name	Value	Category
NVM_E_INTEGRITY_FAILED	NVM_E_INTEGRITY_FAILED	C
NVM_E_LOSS_OF_REDUNDANCY	NVM_E_LOSS_OF_REDUNDANCY	C
NVM_E_QUEUE_OVERFLOW	NVM_E_QUEUE_OVERFLOW	C
NVM_E_REQ_FAILED	NVM_E_REQ_FAILED	C
NVM_E_VERIFY_FAILED	NVM_E_VERIFY_FAILED	C
NVM_E_WRITE_PROTECTED	NVM_E_WRITE_PROTECTED	C
NVM_E_WRONG_BLOCK_ID	NVM_E_WRONG_BLOCK_ID	C

5.2 FEE Module (Internal EEPROM)

FEE is an Mcal module, and see first Fee User Manual / Fee Integration Manual of MCU manufacturer for matters concerning configuration.

5.2.1 FeeGeneral Container

See the following configuration.

Parameter Name	Value	Category
FeeDevErrorDetect	True	C
FeeIndex	0	F
FeeNvmJobEndNotification	NvM_JobEndNotification	F
FeeNvmJobErrorNotification	NvM_JobErrorNotification	F
FeePollingMode	False	N
FeeSetModeSupported	True	F
FeeVersionInfoApi	True	C
¹⁾ FeeVirtualPageSize	User Defined	C

1) FeeVirtualPageSize:

Mcu dependant items (see Fee User/Integration Manual of MCU manufacturer.)

5.2.2 BlockConfiguration Configuration

Fee blocks must be added as many as the number of NvBlock (NvMNvBlockNum setting value) of NvMBlock.

For example, if the value of NvMNvBlockNum is 2 (Redundant Block), two Fee blocks must be created.

If the Fee blocks are linked to the same NvMBlock, other setting values excluding FeeBlockNumber must be the same,

and FeeBlockNumber of the second block is +1 (index) to FeeBlockNumber of the first block.

See the following configuration.

Parameter Name	Value	Category
FeeBlockNumber ¹⁾	User Defined	C
FeeBlockSize ²⁾	User Defined	C
FeeImmediateData	False	C
FeeNumberOfWriteCycles	0	C
FeeDeviceIndex	FlsGeneral	F

1) BlockNumber

Basic formula: $2^n * \text{NvMNvBlockBaseNumber} + \text{Index}$ (n : NvMDatasetSelectionBits)

Ex) The configuration parameter NvMDatasetSelectionBits is configured to be 2.

- For a native NVRAM block with NvMNvBlockBaseNumber = 2:
 - NV block is accessed with FEE/EA_BLOCK_NUMBER = 8
- For a redundant NVRAM block with NvMNvBlockBaseNumber = 3:
 - 1st NV block with data index 0 is accessed with FEE/EA_BLOCK_NUMBER = 12
 - 2nd NV block with data index 1 is accessed with FEE/EA_BLOCK_NUMBER = 13
- For a dataset NVRAM block with NvMNvBlockBaseNumber = 4, NvMNvBlockNum = 3:
 - NV block #0 with data index 0 is accessed with FEE/EA_BLOCK_NUMBER = 16
 - NV block #1 with data index 1 is accessed with FEE/EA_BLOCK_NUMBER = 17
 - NV block #2 with data index 2 is accessed with FEE/EA_BLOCK_NUMBER = 18

2) BlockSize

Set the same as linked NvMNvBlockLength.

5.2.3 FeelfxSpecificConfig (Only Aurix)

Parameter Name	Value	Category
FeeMaxBlockCount ¹⁾	User Defined	C

1) FeeMaxBlockCount

Put the number of configured blocks.

※ Users must not change an unmentioned configuration.

5.2.4 Fee Information

1) Aurix

- In the state of Internal EEPROM Virgin, Fee Init (i.e., in factory) must not be interrupted (i.e., reset). (During the first switch on (i.e., in factory), the FEE driver marks the state pages to indicate a valid state. (This operation is indicated by FEE status != MEMIF_IDLE). It is to be ensured that there are no interruptions during this FEE operation else the FEE might reach the Illegal state.) - See Fee User's Manual for more details.

5.2.5 Redundant Block Distributed Arrangement (Only Chorus/Bolero)

The distributed arrangement of two Fee blocks linked to redundant NvM block in ClusterGroup0 and ClusterGroup1 is better than arranging in one place in terms of DFLASH life span.

5.3 FLS Module (Internal EEPROM)

FLS is an Mcal module, and see first FLS User Manual / Integration Manual of MCU manufacturer for matters concerning configuration.

5.3.1 FLSGeneral Container

See the following configuration.

Parameter Name	Value	Category
FlsAcLoadOnJobStar	False	C
FlsBaseAddress	0	C
FlsCancelApi	False	C
FlsCompareApi	False	C
FlsDevErrorDetect	User Defined	C
FlsDriverIndex	0	F
FlsGetJobResultApi	True	F
FlsGetStatusApi	True	F
FlsSetModeApi	True	F
FlsTotalSize	User Defined	C
FlsUseInterrupts	False	N
FlsVersionInfoApi	False	C

1) FlsTotalSize

Mcu dependant items (see FLS User/Integration Manual of MCU manufacturer.)

5.3.2 FLSConfigSet

See the following configuration.

Parameter Name	Value	Category
FlsAcErase ¹⁾	User Defined	C
FlsAcWrite ¹⁾	User Defined	C
FlsCallCycle ¹⁾	User Defined	C
FlsDefaultMode	MEMIF_MODE_FAST	F
FlsJobEndNotification	Fee_JobEndNotification	F
FlsJobErrorNotification	Fee_JobErrorNotification	F
FlsMaxReadFastMode ¹⁾	User Defined	C
FlsMaxReadNormalMode ¹⁾	User Defined	C
FlsMaxWriteFastMode ¹⁾	User Defined	C
FlsMaxWriteNormalMode ¹⁾	User Defined	C
FlsProtection ¹⁾	User Defined	C

1) Mcu dependant items (see FLS User/Integration Manual of MCU manufacturer.)

5.3.3 FLSDemEventParameterRefs

See the following configuration.

Parameter Name	Value	Category
FLS_E_COMPARE_FAILED	FLS_E_COMPARE_FAILED	C
FLS_E_ERASE_FAILED	FLS_E_ERASE_FAILED	C
FLS_E_READ_FAILED	FLS_E_READ_FAILED	C
FLS_E_UNEXPECTED_FLASH_ID	FLS_E_UNEXPECTED_FLASH_ID	C
FLS_E_WRITE_FAILED	FLS_E_WRITE_FAILED	C

5.3.4 FlsSectorList

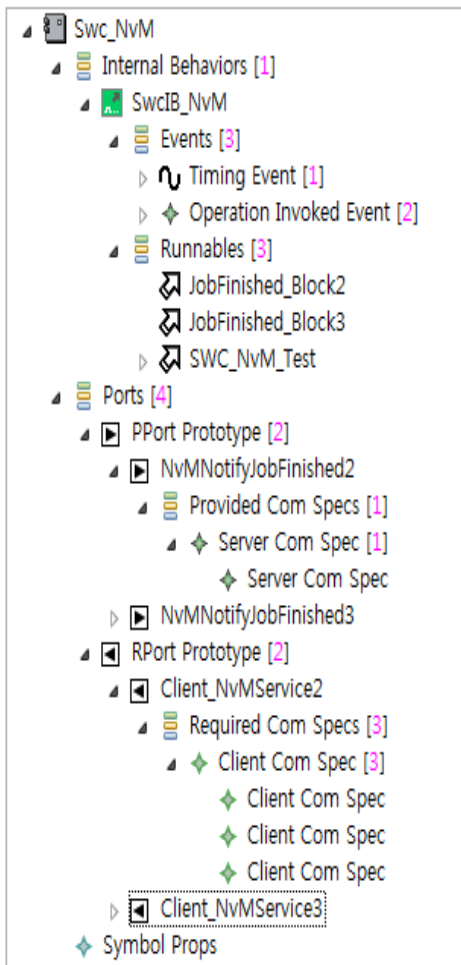
See the following configuration.

Parameter Name	Value	Category
FlsNumberOfSectors ¹⁾	User Defined	C
FlsPageSize ¹⁾	User Defined	C
FlsSectorSize ¹⁾	User Defined	C
FlsSectorStartaddress ¹⁾	User Defined	C

1) Mcu dependant items (see Fls User/Integration Manual of MCU manufacturer.)

5.4 System Configuration

5.4.1 ApplicationSwComponentType Configuration



Generate one Pport and one Rport each.

PPort

short name : NvMNotifyJobFinished<Block ID>
(Port for Notification)

NvMNotifyJobFinished mapping on Provided Interface

ServerComSpec generation and JobFinished mapping on

Operation

Enter 1 in Queue.

RPort

short name : Client_NvMService<Block ID> (Port for Service)

NvMService mapping on Required Interface

ClientComSpec generation

API mapping necessary for each operation item

(GetErrorStatus, ReadBlock, WriteBlock, etc.)

SwcInternalBehavior

Runnable Entity generation

Short name : JobFinished_Block<Block ID>

Symbol : JobFinished_Block<Block ID>

CanBeInvokedConcurrently = false

Events / OperationInvokedEvent generation

Short name : OperationInvokedEvent_JobFinished_<Block ID>

JobFinished_Block<Block ID> mapping on StartOnEvent

Operation / POperation In Atomic Swc Instance Ref generation

Context P Port : NvMNotifyJobFinished<Block ID>

Target Provided Operation : JobFinished

Ex) Service Interface Runnables registration (GetErrorStatus , ReadBlock, WriteBlock)

Short name : SWC_NvM_Test (function to call out NvM Api)

Symbol : SWC_NvM_Test

Server call points/generation of three SynchronousServerCallPoint

short name : SynchronousServerCallPoint_GetErrorStatus_<Block ID>

Operation / ROperation In Atomic Swc Instance Ref generation

Context R Port : Client_NvMService<Block ID>

Target Required Operation : GetErrorStatus

short name : SynchronousServerCallPoint_ReadBlock_<Block ID>

Operation / ROperation In Atomic Swc Instance Ref generation

Context R Port : Client_NvMService<Block ID>

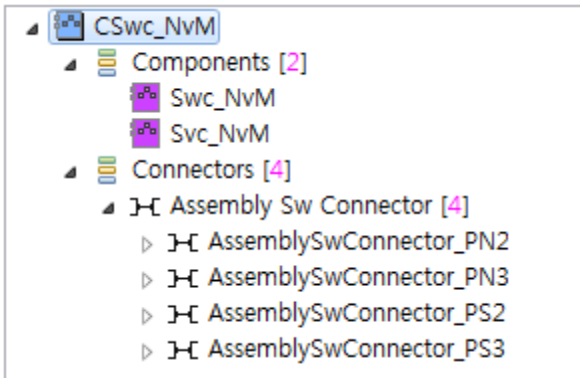
Target Required Operation : ReadBlock

short name : SynchronousServerCallPoint_WriteBlock_<Block ID>

Operation / ROperation In Atomic Swc Instance Ref generation

Context R Port : Client_NvMService<Block ID>
Target Required Operation : WriteBlock

5.4.2 CompositionSwComponentType Configuration



AssemblySwConnector generation

Generation of one Provider and one Requester each at AssemblySwConnector.

Short name : AssemblySwConnector_PS<Block ID>

PPort In Composition Instance Ref

Context Component : NvM

Target P Port : PS<Block ID>

Base : CompositionSwComponentType

RPort In Composition Instance Ref

Context Component : SWC_Service

Target R Port : Client_NvMService<Block ID>

Base : CompositionSwComponentType

Short name : AssemblySwConnector_PN<Block ID>

PPort In Composition Instance Ref

Context Component : SWC_Service

Target P Port : NvMNotifyJobFinished<Block ID>

Base : CompositionSwComponentType

RPort In Composition Instance Ref

Context Component : NvM

Target R Port : PN<Block ID>

Base : CompositionSwComponentType

※ See AUTOSAR BSW Service API Guide.doc for the configuration of ClientServer Interface.

6. Application Programming Interface (API)

6.1 Type Definitions

6.1.1 NvM_RequestResultType

Type:	uint8		
Range	NVM_REQ_OK	0x00	The last asynchronous read/write/control request has been finished successfully. This shall be the default value after reset. This status shall have the value 0.
	NVM_REQ_NOT_OK	0x01	The last asynchronous read/write/control request has been finished unsuccessfully.
	NVM_REQ_PENDING	0x02	An asynchronous read/write/control request is currently pending.
	NVM_REQ_INTEGRITY_FAILED	0x03	The result of the last asynchronous request NvM_ReadBlock or NvM_ReadAll is a data integrity failure. Note: In case of NvM_ReadBlock the content of the RAM block has changed but has become invalid. The application is responsible to renew and validate the RAM block content.
	NVM_REQ_BLOCK_SKIPPED	0x04	The referenced block was skipped during execution of NvM_ReadAll or NvM_WriteAll, e.g. Dataset NVRAM blocks (NvM_ReadAll) or NVRAM blocks without a permanently configured RAM block.
	NVM_REQ_NV_INVALIDATED	0x05	The referenced NV block is invalidated.
	NVM_REQ_CANCELED	0x06	The multi block request NvM_WriteAll was cancelled by calling NvM_CancelWriteAll. Or Any single block job request (NvM_ReadBlock, NvM_WriteBlock, NvM_EraseNvBlock, NvM_InvalidateNvBlock and NvM_RestoreBlockDefaults) was cancelled by calling NvM_CancelJobs.
	NVM_REQ_REDUNDANCY_FAILED	0x07	The required redundancy of the referenced NV block is lost.
	NVM_REQ_RESTORED_FROM_ROM	0x08	The referenced NV block has been restored from ROM.
Description:	This is an asynchronous request result returned by the API service NvM_GetErrorStatus. The availability of an asynchronous request result can be additionally signaled via a callback function.		

※ Footnote

1) NVM_REQ_OK

The state of a block when a request was successful

In addition, when InitBlockCallback is called (InitBlockCallback is set) after failing to read, Ram Block considers that Application has handled, and the state of the block is set as NVM_REQ_OK.

2) NVM_REQ_NOT_OK

The state of a block when a request failed

3) NVM_REQ_PENDING

The state of a block with an ongoing request after receiving a request

4) NVM_REQ_INTEGRITY_FAILED

The state of a block when Read/ReadAll failed while reading worked without an error and the cause of failing does not match the CRC value of the read data

5) NVM_REQ_BLOCK_SKIPPED

NVM_REQ_BLOCK_SKIPPED is set for ReadAll or WriteAll, where blocks set as ReadAll/WriteAll progress accordingly while blocks that are not set as ReadAll/WriteAll skip ReadAll/WriteAll.

6) NVM_REQ_NV_INVALIDATED

When NV Block is invalidated (when the job result of Fee is MEMIF_BLOCK_INVALID), the state of a block when a block set in Fee is invalid.

For example, for Ext.EEPROM or Bolero, the block status is NVM_REQ_NV_INVALIDATED in case of reading in an initial state without writing once. (However, NVM_REQ_NV_INVALIDATED does not necessarily mean not writing once.)

7) NVM_REQ_REDUNDANCY_FAILED

This is an error that occurs when both fail in redundant blocks, and this is not separately set on the current platform as it is only defined as a type in the specification without practical use. But as updates cannot be ruled out since it is defined in the specification, handling equally as NVM_REQ_NOT_OK would be a good idea.

8) NVM_REQ_RESTORED_FROM_ROM

The state of a block when failing to read caused the value of Rom block to be saved in Ram block where ROM block is set

In case of failing, even when the result value is NVM_REQ_NV_INVALIDATED or NVM_REQ_INTEGRITY_FAILED,

if ROM block is set and the value of Rom block is saved, it is returned to NVM_REQ_RESTORED_FROM_ROM.

6.2 Macro Constants

None

6.3 Functions

6.3.1 Initialization

<i>Function Name</i>	NvM_Init
----------------------	----------

Syntax:	FUNC(void,NVM_CODE) NvM_Init(void)
Service ID	0x00
Sync/Async	Synchronous
Reentrancy	Non-reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	Service for basic NVRAM Manager initialization.
Preconditions	NA
Configuration Dependency	None

6.3.2 Synchronous Requests

6.3.2.1 NvM_SetDataIndex

Function Name	NvM_SetDataIndex	
Syntax:	FUNC(Std_ReturnType, NVM_CODE) NvM_SetDataIndex(NvM_BlockIdType BlockId, uint8 DataIndex)	
Service ID	0x01	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	BlockId	This uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all information about a single block.
	Data Index	Index position of an NV/ROM block.
Parameters (Inout)	None	
Parameters (Out)	None	

Return Value	Std_ReturnType	E_OK: The index position was set successfully. E_NOT_OK: An error occurred.
Description	The function sets the association of Dataset NV block with its corresponding RAM block by storing the 8 bit DataIndex passed by the application to the index field of the RAM block.	
Preconditions	NvM should be initialized.	
Configuration Dependency	This API is available only if configuration parameter NvMApiConfigClass is not set to NVM_API_CONFIG_CLASS_1	
In Communication with application SW-C	Rte_Call_<P>_SetDataIndex(uint8 DataIndex) <P> : R-Port Name	

6.3.2.2 NvM_GetDataIndex

Function Name	NvM_GetDataIndex	
Syntax:	FUNC(Std_ReturnType, NVM_CODE) NvM_GetDataIndex(NvM_BlockIdType BlockId, P2VAR(uint8, AUTOMATIC, NVM_APPL_DATA) DataIndexPtr)	
Service ID	0x02	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all information about a single block.
Parameters (Inout)	None	
Parameters (Out)	DataIndexPtr	Pointer to store the current dataset index (0 to 255).
Return Value	Std_ReturnType	E_OK: Index position has been retrieved successfully. E_NOT_OK: An error occurred.

Description	This function reads the index (association of NV block with its corresponding RAM block) from the RAM block index field.
Preconditions	NvM should be initialized.
Configuration Dependency	This API is available only if configuration parameter NvMApiConfigClass is not set to NVM_API_CONFIG_CLASS_1
In Communication with application SW-C	Rte_Call_<P>_GetDataIndex(uint8* DataIndexPtr) <P>: R-Port Name

6.3.2.3 NvM_SetBlockProtection

Function Name	NvM_SetBlockProtection	
Syntax:	FUNC(Std_ReturnType, NVM_CODE) NvM_SetBlockProtection(NvM_BlockIdType BlockId, boolean ProtectionEnabled)	
Service ID	0x03	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	BlockId	This uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all information about a single block.
	Protection Enabled	TRUE: Write protection is enabled. FALSE: Write protection is disabled.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: The block was enabled/disabled as requested. E_NOT_OK: An error occurred.
Description	This function enables/disables the write block Protection bit in the RAM block attribute/error/status field. This function is available only if API Configuration Class 3 is enabled.	
Preconditions	NvM should be initialized.	
Configuration Dependency	This API is available only if configuration parameter NvMApiConfigClass is set to NVM_API_CONFIG_CLASS_3	
In Communication with application SW-C	Rte_Call_<P>_SetBlockProtection (Boolean ProtectionEnabled) <P> : R-Port Name	

6.3.2.4 NvM_GetErrorStatus

Function Name	NvM_GetErrorStatus	
Syntax:	FUNC(Std_ReturnType, NVM_CODE) NvM_GetErrorStatus (NvM_BlockIdType BlockId, P2VAR(NvM_RequestResultType, AUTOMATIC, NVM_APPL_DATA) RequestResultPtr)	
Service ID	0x04	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	BlockId	This uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all information about a single block.
Parameters (Inout)	None	
Parameters (Out)	RequestResultPtr	Pointer to store the requested result.
Return Value	Std_ReturnType	E_OK: The block dependent error/status information was read successfully. E_NOT_OK: An error occurred.
Description	Service to read the block dependent error/status information.	
Preconditions	NvM should be initialized.	
Configuration Dependency	None	
In Communication with application SW-C	Rte_Call_<P>_GetErrorStatus (NvM_RequestResultType * RequestResultPtr) <P> : R-Port Name	

6.3.2.5 NvM_GetVersionInfo

Function Name	NvM_GetVersionInfo	
Syntax:	FUNC(void, NVM_CODE) NvM_GetVersionInfo (P2VAR(Std_VersionInfoType, AUTOMATIC, NVM_APPL_DATA) versioninfo)	

<i>Service ID</i>	0x0F	
<i>Sync/Async</i>	Synchronous	
<i>Reentrancy</i>	Reentrant	
<i>Parameters (In)</i>	None	
<i>Parameters (Inout)</i>	None	
<i>Parameters (Out)</i>	None	
<i>Return Value</i>	Versioninfo	Pointer to store the versioninfo of this module.
<i>Description</i>	Service to get the version information of the NvM module.	
<i>Preconditions</i>	NA	
<i>Configuration Dependency</i>	This API is available only if configuration parameter NVM_VERSION_INFO_API is set to STD_ON.	

6.3.2.6 NvM_SetRamBlockStatus

<i>Function Name</i>	NvM_SetRamBlockStatus	
<i>Syntax:</i>	FUNC(Std_ReturnType, NVM_CODE) NvM_SetRamBlockStatus(NvM_BlockIdType BlockId, boolean BlockChanged)	
<i>Service ID</i>	0x05	
<i>Sync/Async</i>	Synchronous	
<i>Reentrancy</i>	Reentrant	
<i>Parameters (In)</i>	BlockId	This uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all information about a single block.
	BlockChanged	TRUE: Validate the RAM block and mark as changed. (FALSE: Invalidate the RAM block and mark as unchanged.
<i>Parameters (Inout)</i>	None	
<i>Parameters (Out)</i>	None	
<i>Return Value</i>	Std_ReturnType	E_OK: The status of the RAM-Block was changed as requested. E_NOT_OK: An error occurred.
<i>Description</i>	This API recalculates the CRC (if configured) for the RAM block data and sets the state of the RAM block to valid/invalid.	

Preconditions	NvM should be initialized.
Configuration Dependency	This API is available only if configuration parameter NVM_SET_RAM_BLOCK_STATUS_API is set to STD_ON.
In Communication with application SW-C	Rte_Call_<P>_SetRamBlockStatus (boolean BlockChanged) <P> : R-Port Name

* Cautions

1. When calling SetRamBlockStatus, Xxx_WriteBlock is always processed normally whether or not the value of BlockChanged is TRUE/FALSE.
2. Only blocks that are VALID/CHANGED are processed at the WriteAll stage. In case of using WriteAll only without WriteBlock, the state of blocks must be notified to NvM by calling SetRamBlockStatus function.

State	Description
INVALID/UNCHANGED	This is a state before reading a block after NvM_Init. Additionally, failing to read a block without error recovery configuration becomes INVALID/UNCHANGED. And finally, calling SetRamBlockStatus(FALSE) transitions to INVALID/UNCHANGED.
VALID/UNCHANGED	Executing ReadBlock or ReadAll functions for the relevant block normally transitions to VALID/UNCHANGED.
VALID/CHANGED	Reading Rom after failing to Read or calling SetRamBlockStatus(TRUE) transitions to VALID/CHANGED.

6.3.2.7 NvM_SetBlockLockStatus

Function Name	NvM_SetBlockLockStatus	
Syntax:	FUNC(void, NVM_CODE) NvM_SetBlockLockStatus(NvM_BlockIdType BlockId, boolean BlockLocked)	
Service ID	0x13	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (In)	BlockId	This uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all information about a single block.
	BlockLocked	TRUE: Mark the RAM block as locked. FALSE: Mark the RAM block as unlocked.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	None	
Description	Service for setting the lock status of a permanent RAM block of an NVRAM block.	
Preconditions	NvM should be initialized.	

Configuration Dependency	This API is available only if configuration parameter NvMApiConfigClass is set to NVM_API_CONFIG_CLASS_3
In Communication with application SW-C	Rte_Call_<P>_ SetBlockLockStatus (boolean BlockLocked) <P> : R-Port Name

6.3.3 Asynchronous Single Block Requests

6.3.3.1 NvM_ReadBlock

Function Name	NvM_ReadBlock	
Syntax:	FUNC(Std_ReturnType, NVM_CODE) NvM_ReadBlock(NvM_BlockIdType BlockId, P2VAR(void, AUTOMATIC, NVM_APPL_DATA) NvM_DstPtr)	
Service ID	0x06	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (In)	BlockId	This uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all information about a single block.
Parameters (Inout)	None	
Parameters (Out)	NvM_DstPtr	Pointer to the RAM data block.
Return Value	Std_ReturnType	E_OK: request has been accepted E_NOT_OK: request has not been accepted
Description	Request updates the job queue with the BlockId, NvM_DstPtr and Service Id to 'Read' the NV/ROM block data to its corresponding RAM block.	
Preconditions	NvM should be initialized.	
Configuration Dependency	This API is available only if configuration parameter NvMApiConfigClass is set to NVM_API_CONFIG_CLASS_2 / NVM_API_CONFIG_CLASS_3.	
In Communication with application SW-C	Rte_Call_<P>_ ReadBlock (void* NvM_DstPtr) <P> : R-Port Name	

6.3.3.2 NvM_WriteBlock

Function Name	NvM_WriteBlock	
Syntax:	FUNC(Std_ReturnType, NVM_CODE) NvM_WriteBlock(NvM_BlockIdType BlockId, P2CONST(void, AUTOMATIC, NVM_APPL_DATA) NvM_SrcPtr)	
Service ID	0x07	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (In)	BlockId	This uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all information about a single block.
	NvM_SrcPtr	Pointer to the RAM data block.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: request has been accepted E_NOT_OK: request has not been accepted
Description	Service to copy the data of the RAM block to its corresponding NV block.	
Preconditions	NvM should be initialized.	
Configuration Dependency	This API is available only if configuration parameter NvMApiConfigClass is set to NVM_API_CONFIG_CLASS_2/ NVM_API_CONFIG_CLASS_3.	
In Communication with application SW-C	Rte_Call_<P>_ WriteBlock (const void* NvM_SrcPtr) <P> : R-Port Name	

6.3.3.3 NvM_RestoreBlockDefaults

Function Name	NvM_RestoreBlockDefaults	
Syntax:	FUNC(Std_ReturnType, NVM_CODE) NvM_RestoreBlockDefaults(NvM_BlockIdType BlockId, P2VAR(void, AUTOMATIC, NVM_APPL_DATA) NvM_DstPtr) 	

Service ID	0x08	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (In)	BlockId	This uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all information about a single block.
Parameters (Inout)	None	
Parameters (Out)	NvM_DstPtr	Pointer to the RAM data block.
Return Value	Std_ReturnType	E_OK: request has been accepted E_NOT_OK: request has not been accepted
Description	Service to restore the default data to its corresponding RAM block.	
Preconditions	NvM should be initialized.	
Configuration Dependency	This API is available only if configuration parameter NvMApiConfigClass is set to NVM_API_CONFIG_CLASS_2/ NVM_API_CONFIG_CLASS_3.	
In Communication with application SW-C	Rte_Call_<P>_RestoreBlockDefaults (void* NvM_ DstPtr) <P> : R-Port Name	

6.3.3.4 NvM_EraseNvBlock

Function Name	NvM_EraseNvBlock	
Syntax:	FUNC(Std_ReturnType, NVM_CODE) NvM_EraseNvBlock(NvM_BlockIdType BlockId)	
Service ID	0x09	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (In)	BlockId	This uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all information about a single block.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: request has been accepted E_NOT_OK: request has not been accepted

<i>Description</i>	Request updates the job queue with the BlockId and Service Id to 'Erase' the NV block data.
<i>Preconditions</i>	NvM should be initialized.
<i>Configuration Dependency</i>	This API is available only if configuration parameter NvMApiConfigClass is set to NVM_API_CONFIG_CLASS_3.
<i>In Communication with application SW-C</i>	Rte_Call_<P>_EraseNvBlock (void) <P> : R-Port Name

6.3.3.5 NvM_CancelWriteAll

<i>Function Name</i>	NvM_CancelWriteAll
<i>Syntax:</i>	FUNC(void, NVM_CODE)NvM_CancelWriteAll(void)
<i>Service ID</i>	0x0A
<i>Sync/Async</i>	Asynchronous
<i>Reentrancy</i>	Non Reentrant
<i>Parameters (In)</i>	None
<i>Parameters (Inout)</i>	None
<i>Parameters (Out)</i>	None
<i>Return Value</i>	None
<i>Description</i>	Cancels a running NvM_WriteAll request.
<i>Preconditions</i>	NvM should be initialized.
<i>Configuration Dependency</i>	None

6.3.3.6 NvM_InvalidateNvBlock

<i>Function Name</i>	NvM_InvalidateNvBlock
<i>Syntax:</i>	FUNC(Std_ReturnType, NVM_CODE) NvM_InvalidateNvBlock(NvM_BlockIdType BlockId)
<i>Service ID</i>	0x0B
<i>Sync/Async</i>	Asynchronous

Reentrancy	Reentrant	
Parameters (In)	BlockId	This uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all information about a single block.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: request has been accepted E_NOT_OK: request has not been accepted
Description	Request updates the job queue with the BlockId and Service Id to 'Invalidate' the NV block data.	
Preconditions	NvM should be initialized.	
Configuration Dependency	This API is available only if configuration parameter NvMApiConfigClass is set to NVM_API_CONFIG_CLASS_3.	
In Communication with application SW-C	Rte_Call_<P>_InvalidateNvBlock (void) <P> : R-Port Name	

6.3.3.7 NvM_CancelJobs

Function Name	NvM_CancelJobs	
Syntax:	FUNC(Std_ReturnType, NVM_CODE) NvM_CancelJobs(NvM_BlockIdType BlockId)	
Service ID	0x10	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (In)	BlockId	This uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all information about a single block.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: request has been accepted E_NOT_OK: request has not been accepted
Description	Service to cancel all jobs pending for a NV block.	
Preconditions	NvM should be initialized.	

<i>Configuration Dependency</i>	None
---------------------------------	------

6.3.4 Asynchronous Multi Block Requests

6.3.4.1 NvM_ReadAll

<i>Function Name</i>	NvM_ReadAll
<i>Syntax:</i>	FUNC(void, NVM_CODE) NvM_ReadAll(void)
<i>Service ID</i>	0x0C
<i>Sync/Async</i>	Asynchronous
<i>Reentrancy</i>	Non Reentrant
<i>Parameters (In)</i>	None
<i>Parameters (Inout)</i>	None
<i>Parameters (Out)</i>	None
<i>Return Value</i>	None
<i>Description</i>	Initiates a multi block read request.
<i>Preconditions</i>	NvM should be initialized.
<i>Configuration Dependency</i>	None

6.3.4.2 NvM_WriteAll

<i>Function Name</i>	NvM_WriteAll
<i>Syntax:</i>	FUNC(void, NVM_CODE) NvM_WriteAll(void)
<i>Service ID</i>	0x0D
<i>Sync/Async</i>	Asynchronous
<i>Reentrancy</i>	Non Reentrant
<i>Parameters (In)</i>	None
<i>Parameters (Inout)</i>	None
<i>Parameters (Out)</i>	None

<i>Return Value</i>	None
<i>Description</i>	Initiates a multi block write request.
<i>Preconditions</i>	NvM should be initialized.
<i>Configuration Dependency</i>	None

6.3.5 Callback Notifications

6.3.5.1 NvM_JobEndNotification

<i>Function Name</i>	NvM_JobEndNotification
<i>Syntax:</i>	void NvM_JobEndNotification(void)
<i>Service ID</i>	0x11
<i>Sync/Async</i>	Synchronous
<i>Reentrancy</i>	Non Reentrant
<i>Parameters (In)</i>	None
<i>Parameters (Inout)</i>	None
<i>Parameters (Out)</i>	None
<i>Return Value</i>	None
<i>Description</i>	Function to be used by the underlying memory abstraction to signal end of job without error.
<i>Preconditions</i>	None
<i>Configuration Dependency</i>	This function is available only if configuration parameter NvMPollingmode is set to STD_OFF.

6.3.5.2 NvM_JobErrorNotification

<i>Function Name</i>	NvM_JobErrorNotification
<i>Syntax:</i>	void NvM_JobErrorNotification(void)
<i>Service ID</i>	0x12
<i>Sync/Async</i>	Synchronous
<i>Reentrancy</i>	Non Reentrant
<i>Parameters (In)</i>	None
<i>Parameters (Inout)</i>	None
<i>Parameters (Out)</i>	None

Return Value	None
Description	Function to be used by the underlying memory abstraction to signal end of job with error.
Preconditions	None
Configuration Dependency	This function is available only if configuration parameter NvMPollingMode is set to STD_OFF.

6.3.6 Scheduled Functions

6.3.6.1 NvM_Mainfunction

Function Name	NvM_MainFunction
Syntax:	FUNC(void, NVM_CODE) NvM_MainFunction(void)
Service ID	0x0E
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	Function performs the processing of the NVRAM Manager jobs. This function has to be called cyclically in every case
Preconditions	NVRAM Manager should be initialized.
Configuration Dependency	None

6.3.7 CddIf

6.3.7.1 NvM_CddGetStatus

Function Name	NvM_CddGetStatus
Syntax:	FUNC(NvM_OpStatusType, NVM_CODE) NvM_CddGetStatus(void)
Service ID	None
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None
Return Value	NvM_OpStatusType: NVM_OPSTATUS_UNINIT:The memory stack has not been initialized. NVM_OPSTATUS_IDLE:The memory stack is currently idle. NVM_OPSTATUS_BUSY:The memory stack is currently busy.
Description	It is a function used when there is a need to check the state of BSW memory modules.
Preconditions	None
Configuration Dependency	None

6.3.7.2 NvM_UserJobFunction

Function Name	UserJobFunction
Syntax:	FUNC(NvM_OpStatusType, NVM_CODE) NvM_UserJobFunction (void)
Service ID	None
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None
Return Value	NvM_OpStatusType:

	<p>NVM_OPSTATUS_IDLE: user job is idle</p> <p>NVM_OPSTATUS_BUSY: user job is busy</p>
Description	<p>When users set NvMUserJobFunction, UserJob function is driven in OsTask_BSW_Mem_Process Task.</p> <p>NVM_OPSTATUS_IDLE must be returned when there is nothing to handle in UserJobFunction.</p> <p>NVM_OPSTATUS_BUSY is also returned when UserJob function needs to be executed. And mainfunctions of memory stacks are not driven. Therefore, UserJob processing must be completed and NVM_OPSTATUS_IDLE needs to be returned as soon as possible.</p> <p>Note: UserJob must be started when the return value of NvM_CddGetStatus is NVM_OPSTATUS_IDLE.</p>
Preconditions	None
Configuration Dependency	This function is available only if configuration parameter NvMUserJobFunction is configured.

6.3.7.3 NvM_UserWdgToggleFunction

Function Name	UserWdgToggleFunction
Syntax:	void NvM_UserWdgToggleFunction (void)
Service ID	None
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	<p>If the user configures NvMUserWdgToggleFunction, user functions are repeatedly performed in mem_EalnitPerform and Mem_FeelnitPerform until initialization is completed. This function should be used only when Reset occurs before memory initialization is completed because External Watchdog is used and Timer configuration time is short. In general, it is recommended to use the Watchdog function provided by the platform.</p>

<i>Preconditions</i>	None
<i>Configuration Dependency</i>	This function is available only if configuration parameter NvMUserWdgToggleFunction is configured.

6.3.8 Notes

6.3.8.1 In Communication with application SW-C

See AUTOSAR BSW Service API Guide.doc for prototypes of RTE-based functions.

7. Generator

7.1 Generator Option

7.1.1 NvM

Option	Description
S	Generating Mode Switch Interface for ECU Mode and Client Server Interface and P-Port for each service in Swcd_Bsw_EcuM.arxml.
P	Generating Port Name according to Autosar NvM SWS 4.2.2 spec.

7.2 Generator Error Message

7.2.1 NvM

7.2.1.1 Error Messages

1) ERR020001

Unexpected Error Found. Please contact Support System.

This error occurs, if the number of fields is not same in the structure that is to be generated in the C Source file. Contact Support System.

2) ERR020002

Unexpected Error Found. This error may be due to the incorrect configuration of the element(s) 'Element Name'. If the error is not resolved, then please contact Support System.

This error occurs, if the structure fields that are to be generated in the C Source file are empty. Contact Support System.

3) ERR020003

'NvM' Component is not present in the input file(s).

This error occurs, if NvM component is not present in any of the input ECU Configuration Description File(s).

4) ERR020004

The reference path is empty for the parameter 'Parameter Name' in the container 'Container Name', having short name 'Short Name'.

This error occurs, if reference path is not provided for the parameter 'parameter name'.

Container Name	Parameter Name
NvMEaRef	NvMNameOfEaBlock

Container Name	Parameter Name
NvMFeeRef	NvMNameOfFeeBlock

5) ERR020005

The parameter 'Parameter Name' in the container 'Container Name' should be configured.

This error occurs, if any of the mandatory configuration parameters mentioned below is not configured in ECU Configuration Description File.

Container Name	Parameter Name
NvMBlockDescriptor	NvMBlockJobPriority
	NvMBlockManagementType
	NvMBlockUseCrc
	NvMBlockUseSyncMechanism
	NvMBlockWritePort
	NvMbswMMultiBlockJobStatusInformation
	NvMMaxNumOfReadRetries
	NvMMaxNumOfWriteRetries
	NvMNvBlockBaseNumber
	NvMNvBlockLength
	NvMNvBlockNum
	NvMNvramBlockIdentifier
	NvMNvramDeviceId
	NvMResistantToChangedSw
	NvMRomBlockNum
	NvMStaticBlockIdCheck
	NvMWriteBlockOnce
	NvMWriteVerification
	NvMWriteVerificationDataSize
	NvMDefaultROMCRCEnabled
NvMCommon	NvMApiConfigClass

Container Name	Parameter Name
	NvMBswMBlockStatusInformation
	NvMCompiledConfigId
	NvMCrcNumOfBytes
	NvMDatasetSelectionBits
	NvMDevErrorDetect
	NvMDrvModeSwitch
	NvMDynamicConfiguration
	NvMJobPrioritization
	NvMMainFunctionCycleTime
	NvMPollingMode
	NvMRepeatMirrorOperations
	NvMSetRamBlockStatusApi
	NvMSizeStandardJobQueue
	NvMVersionInfoApi

6) ERR020006

The value configured for the parameter 'Parameter Name' in the container 'Container Name' should follow the pattern: <Pattern>.

This error occurs, when the parameter 'Parameter Name' is not configured as per the pattern.

Parameter Name	Container Name	Pattern	Example
AR-RELEASE-VERSION	BSW-IMPLEMENTATION	4.[0-9]+.[0-9]+	4.0.3
SW-VERSION		1.[0-9]+.[0-9]+	1.0.0
NvMInitBlockCallback	NvMBlockDescriptor	[a-zA-Z][a-zA-Z0-9W_]*	InitBlockCallback_0
NvMRamBlockDataAddress			RamBlockDataAddress_1
NvMReadRamBlockFromNvCallback			ReadRamBlock_1
NvMRomBlockDataAddress			RomBlockDataAddress_1
NvMSingleBlockCallback			SingleBlockCallback_1

Parameter Name	Container Name	Pattern	Example
NvMWriteRamBlockToNvCallback			WriteRamBlock_1
NvMApiConfigClass	NvMCommon	[a-zA-Z][a-zA-Z0-9W_]*	NVM_API_CONFIG_CLASS_2
NvMMultiBlockCallback			MULTI_BLOCK_CBK

7) ERR020008

Value of the parameter 'NvMBlockManagementType' in the container 'NvMBlockDescriptor' should not be configured as <NVM_BLOCK_DATASET>, since value of the parameter 'Parameter Name' in the container 'NvMCommon' is configured as <Value>.

This error occurs, if the value of the parameter NvMBlockManagementType in the container NvMBlockDescriptor is configured as <NVM_BLOCK_DATASET>, when the below mentioned parameters 'Parameter Name' are configured as <value>.

Parameter Name	Value
NvMApiConfigClass	NVM_API_CONFIG_CLASS_1
NvMDatasetSelectionBits	1

8) ERR020013

The reference path <Reference Path> provided for the parameter 'Parameter Name' in the container 'Container Name', having short name <Short Name> is incorrect.

This error occurs, if incorrect reference is provided for any of the below parameters.

Container Name	Parameter Name
NvMEaRef	NvMNameOfEaBlock
NvMFeeRef	NvMNameOfFeeBlock
NvmDemEventParameterRefs	NVM_E_INTEGRITY_FAILED
	NVM_E_LOSS_OF_REDUNDANCY
	NVM_E_QUEUE_OVERFLOW
	NVM_E_REQ_FAILED
	NVM_E_VERIFY_FAILED
	NVM_E_WRITE_PROTECTED
	NVM_E_WRONG_BLOCK_ID

9) ERR020051

When value configured for the parameter 'NvMDatasetSelectionBits' is <0>, the value of the parameter 'NvMBlockManagementType' should not be configured as <NVM_BLOCK_DATASET/NVM_BLOCK_REDUNDANT> in the container 'NvMBlockDescriptor'.

This error occurs, if the value configured for the parameter NvMDatasetSelectionBits is 0 when the value of the parameter NvMBlockManagementType is configured as

NVM_BLOCK_DATASET /NVM_BLOCK_REDUNDANT in the container NvMBlockDescriptor.

10) ERR020052

Value of the parameter 'NvMNvBlockNum' should be configured as <1>, when the value of the parameter 'NvMBlockManagementType' is configured as <NVM_BLOCK_NATIVE> in the container 'NvMBlockDescriptor'.

This error occurs, if the value of the parameter NvMNvBlockNum is not configured as 1 when the value of the parameter NvMBlockManagementType is configured as NVM_BLOCK_NATIVE in the container NvMBlockDescriptor.

11) ERR020053

Value of the parameter 'NvMNvBlockNum' should be configured as <2>, when the value of the parameter 'NvMBlockManagementType' is configured as <NVM_BLOCK_REDUNDANT> in the container 'NvMBlockDescriptor'.

This error occurs, if the value of the parameter NvMNvBlockNum is not configured as 2 when the value of the parameter NvMBlockManagementType is configured as NVM_BLOCK_REDUNDANT in the container NvMBlockDescriptor.

12) ERR020054

Value of the parameter 'NvMBlockUseSyncMechanism' should be configured as <false/0>, when the value of the parameter 'NvMWriteVerification' is configured as <true/1> in the container 'NvMBlockDescriptor'.

This error occurs, if the value configured for the parameter NvMBlockUseSyncMechanism is <true/1> when the value of the parameter NvMWriteVerification is configured as <true/1> in the container NvMBlockDescriptor.

13) ERR020055

Value configured for the parameter 'NvMRomBlockNum' should range from <0> to <1> when the value of the parameter 'NvMBlockManagementType' is configured as <NVM_BLOCK_NATIVE/NVM_BLOCK_REDUNDANT> in the container 'NvMBlockDescriptor'.

This error occurs, if the value configured for the parameter NvMRomBlockNum is not in the range of 0 to 1 when the value of the parameter NvMBlockManagementType is configured as NVM_BLOCK_NATIVE/ NVM_BLOCK_REDUNDANT in the container NvMBlockDescriptor, for each configured block.

14) ERR020056

The sum of parameters 'NvMRomBlockNum' and 'NvMNvBlockNum' should be less than or equal to <255> when the value of the parameter 'NvMBlockManagementType' is configured as <NVM_BLOCK_DATASET> in the container 'NvMBlockDescriptor'.

This error occurs, if the sum of the value of the parameters NvMRomBlockNum and NvMNvBlockNum is greater than 255 when the value of the parameter NvMBlockManagementType is configured as NVM_BLOCK_DATASET in the container NvMBlockDescriptor for each configured block.

15) ERR020057

Value of the parameters 'NvMReadRamBlockFromNvCallback' and 'NvMWriteRamBlockToNvCallback' should be configured since the value of the parameter 'NvMBlockUseSyncMechanism' in the container is configured in the container 'NvMBlockDescriptor'.

This error occurs, if the value of the parameters NvMReadRamBlockFromNvCallback and NvMWriteRamBlockToNvCallback are not configured when the value of the parameter NvMBlockUseSyncMechanism is configured in the container NvMBlockDescriptor.

16) ERR020058

<Value 1> value is not unique. Function names configured across parameters 'Parameter Name' should be unique across the 'NvMBlockDescriptor'.

This error occurs, if the value configured for the below mentioned parameters is not unique.

Parameter Name	Value1
NvMRamBlockDataAddress	RamBlockDataAdress_2
NvMRomBlockDataAddress	RamBlockDataAdress_2
NvmInitBlockCallback	RamBlockDataAdress_2
NvmSingleBlockCallback	RamBlockDataAdress_2

17) ERR020059

Value of the parameter 'NvMSizeImmediateJobQueue' should be configured and should range from <1> to <255>, if the value of the parameter 'NvMJobPrioritization' is configured as <true/1> in the container 'NvMCommon'.

This error occurs, if the value of the parameter NvMSizeImmediateJobQueue is not configured and is not in the range of 1 to 255 when the value of the parameter NvMJobPrioritization is configured as <true/1> in the container NvMCommon.

18) ERR020060

Value of the parameter 'NvMNvBlockNum' in the container 'NvMBlockDescriptor' should be less than or equal to $2^{NvMDatasetSelectionBits}$ in the container 'NvMCommon'.

This error occurs, if the value of the parameter NvMNvBlockNum in the container NvMBlockDescriptor is greater than $2^{NvMDatasetSelectionBits}$ in the container NvMCommon.

19) ERR020061

Value configured for the parameter 'NvMNvBlockBaseNumber' in the container 'NvMBlockDescriptor' should be equal to the value configured for the parameter 'Parameter Name' in the container 'Container Name' right shifted by 'NvMDatasetSelectionBits' < EaBlockNumber / FeeBlockNumber << NvMDatasetSelectionBits >.

This error occurs, if the value configured for the parameter `NvMNvBlockBaseNumber` in the container `NvMBlockDescriptor` is not equal to the value (right shifted by `NvMDatasetSelectionBits`) configured for the below mentioned parameter.

Parameter Name	Container Name
<code>FeeBlockNumber</code>	<code>FeeBlockConfiguration</code>
<code>EaBlockNumber</code>	<code>EaBlockConfiguration</code>

20) ERR020062

Value of the parameter '`NvMWriteVerificationDataSize`' should be less than or equal to '`NvMNvBlockLength`' and '`NvMNvBlockLength`' should be completely divisible by '`NvMWriteVerificationDataSize`' in the container '`NvMBlockDescriptor`'.

if the value of the parameter `NvMWriteVerificationDataSize` is greater than `NvMNvBlockLength` and `NvMNvBlockLength` is not completely divisible by `NvMWriteVerificationDataSize`.

21) ERR020063

Value of the parameter '`NvMBlockManagementType`' should be `<NVM_BLOCK_REDUNDANT>` and the value of the parameter '`NvMRamBlockDataAddress`' should be configured in the container '`NvMBlockDescriptor`' for the block in which '`NvMNvramBlockIdentifier`' is configured as `<1>`.

This error occurs, if the value configured for the parameter `NvMBlockManagementType` is configured other than `<NVM_BLOCK_REDUNDANT>` and the value of the parameter `NvMRamBlockDataAddress` is not configured in the container `NvMBlockDescriptor` for the block in which `NvMNvramBlockIdentifier` is configured as `<1>`.

22) ERR020064

Value of the parameter '`NvMBlockUseSyncMechanism`' should not be configured as `<true/1>`, when the value of the parameter '`NvMBlockManagementType`' is configured as `<NVM_BLOCK_DATASET>` in the container '`NvMBlockDescriptor`'.

This error occurs, if the value of the parameter '`NvMBlockUseSyncMechanism`' is configured as `<true/1>` when value of the parameter `NvMBlockManagementType` is configured as `<NVM_BLOCK_DATASET>` in the container '`NvMBlockDescriptor`'.

23) ERR020065

Value of the parameter '`NvMRamBlockDataAddress`' should be configured for the block having `BlockId` '`CRC block`' since '`Parameter Name`' is configured as `<true/1>`.

This error occurs, if the Value of the parameter '`NvMRamBlockDataAddress`' is not configured for the block having `BlockId` '`CRC block`' since '`Parameter Name`' is configured as `<true/1>`.

Parameter Name
<code>NvMSelectBlockForReadAll</code>

Parameter Name
NvMSelectBlockForWriteAll

24) ERR020066

Value configured for the parameter NvMNvramDeviceId in the container NvMBlockDescriptor should be equal to the value configured for the parameter 'Parameter Name' in the container 'Container Name'.

This error occurs, if the value configured for the parameter NvMNvramDeviceId in the container NvMBlockDescriptor is not equal to the value configured for the below mentioned parameter.

Parameter Name	Container Name
FeeDeviceIndex	FeeBlockConfiguration
EaDeviceIndex	EaBlockConfiguration

25) ERR020068

Value configured for the parameter 'Parameter Name' should be unique in the container 'Container Name'.

This error occurs, if the value configured for the below mentioned parameters is not unique.

Parameter Name	Container Name
NvMNvramBlockIdentifier	NvMscBlockDeriptor
NvMNameOfEaBlock	NvMEaRef
NvMNameOfFeeBlock	NvMFeeRef
NvMReadAllOrder	NvMscBlockDeriptor
NvMWriteAllOrder	NvMscBlockDeriptor

26) ERR020069

The value configured for the parameter 'Parameter Name' in the container 'Container Name' should be sequential.

This error occurs, if the value configured for the below mentioned parameter is not sequential..

Parameter Name	Container Name
NvMNvramBlockIdentifier	NvMscBlockDeriptor
NvMReadAllOrder	NvMscBlockDeriptor
NvMWriteAllOrder	NvMscBlockDeriptor

27) ERR020070

The value of the parameter 'Parameter Name' in the container 'Container Name' should start with <1>.

This error occurs, if the value configured for the below mentioned parameter does not start with 1..

Parameter Name	Container Name
NvMNvramBlockIdentifier	NvMscBlockDeriptor
NvMReadAllOrder	NvMscBlockDeriptor
NvMWriteAllOrder	NvMscBlockDeriptor

28) ERR020071

Value of the parameter 'NvMCalcRamBlockCrc' should be configured as <true/1>, when the value of the parameter 'NvMBlockUseCrc' is configured as <true/1> in the container 'NvMBlockDescriptor'.

This error occurs, if the value configured for the parameter NvMCalcRamBlockCrc is <false/0> when the value configured for the parameter NvMBlockUseCrc is <true/1> in the container NvMBlockDescriptor.

29) ERR020072

Value of the parameter 'NvMBlockCrcType' should be configured, when the value of the parameter 'NvMBlockUseCrc' is configured as <true/1> in the container 'NvMBlockDescriptor'.

This error occurs, if the value of the parameter NvMBlockCrcType is not configured, when the value of the parameter NvMBlockUseCrc is configured as <true/1> in the container NvMBlockDescriptor.

30) ERR020073

Value of the parameter 'NvMBlockUseCrc' should be configured as <true/1>, when the value of the parameter 'NvMWriteBlockOnce' is configured as <true/1> in the container 'NvMBlockDescriptor'.

This error occurs, if the value of the parameter NvMBlockUseCrc is configured as <false/0>, when the value of the parameter NvMWriteBlockOnce is configured as <true/1> in the container NvMBlockDescriptor.

31) ERR020074

Value of the parameter 'NvMBlockWriteProt' should not be configured as <true/1>, when the value of the parameter 'NvMWriteBlockOnce' is configured as <true/1> in the container 'NvMBlockDescriptor'.

This error occurs, if the value of the parameter NvMBlockWriteProt is configured as <true/1>, when the value of the parameter NvMWriteBlockOnce is configured as <true/1> in the container NvMBlockDescriptor.

32) ERR020075

CRC blocks should be configured since block id <BlockIdentifier> is configured for 'NvMStaticBlockIDCheck' as <value of the parameter NvMStaticBlockIDCheck> and 'NvMBlockUseCrc' as <value of the parameter NvMBlockUseCrc> in the container 'NvMBlockDescriptor'.

This error occurs, if CRC Blocks are not configured when Main Block 'NvMStaticBlockIDCheck' is configured as <true/1,false/0> or 'NvMBlockUseCrc' is configured as <true/1,false/0>, when 'NvMBlockManagementType' is configured as any one of the following 'NVM_BLOCK_NATIVE' or 'NVM_BLOCK_REDUNDANT' or 'NVM_BLOCK_DATASET' in the container 'NvMBlockDescriptor'.

33) ERR020076

Value of the parameter 'NvMRamBlockDataAddress' should be configured, when the value of the parameter 'NvMSelectBlockForReadAll/NvMSelectBlockForWriteAll' is configured <true/1> in the container 'NvMBlockDescriptor'.

This error occurs, if the value configured for the parameter 'NvMRamBlockDataAddress' is not configured, when the value of the parameter 'NvMSelectBlockForReadAll/NvMSelectBlockForWriteAll' is configured as <true/1> in the container 'NvMBlockDescriptor'.

34) ERR020077

Value of the parameter 'NvMBlockWriteProt' should be configured as <false/0>, when the value of the parameter 'NvMSelectBlockForWriteAll' is configured as <true/1> in the container 'NvMBlockDescriptor'.

This error occurs, if the value configured for the parameter 'NvMBlockWriteProt' is configured as <true/1>, when the value of the parameter 'NvMSelectBlockForWriteAll' is configured <true/1> in the container 'NvMBlockDescriptor'.

35) ERR020078

Value of the parameter 'NvMCalcRamBlockCrc' should be configured as <true/1>, when the value of the parameter 'NvMBlockUseCrc' is configured as <true/1> in the container 'NvMBlockDescriptor'.

This error occurs, if the value configured for the parameter 'NvMCalcRamBlockCrc' is configured as <false/0>, when the value of the parameter 'NvMBlockUseCrc' is configured as <true/1> in the container 'NvMBlockDescriptor'.

36) ERR020079

Value of the parameter 'NvMBlockCrcType' should be configured, when the value of the parameter 'NvMBlockUseCrc' is configured as <true/1> in the container 'NvMBlockDescriptor'.

This error occurs, if Value of the parameter 'NvMBlockCrcType' is not configured, when the value of the parameter 'NvMBlockUseCrc' is configured as <true/1> in the container 'NvMBlockDescriptor'.

37) ERR020080

The value of the parameter 'Parameter Name' in the container 'Container Name' should be configured as blank, when the value of the parameter 'NvMNvramBlockIdentifier' is configured as <0> or <1> in the container 'NvMBlockDescriptor'.

This error occurs, if the value configured for the below mentioned parameter does not blank when 'NvMNvramBlockIdentifier' is <0> or <1>.

Parameter Name	Container Name
NvMReadAllOrder	NvMscBlockDeriptor
NvMWriteAllOrder	NvMscBlockDeriptor

7.2.1.2 Warning Messages

1) WRN020003

Parameter 'NvMRomBlockDataAddress' in the container 'NvMBlockDescriptor' should not be configured, since value of the parameter 'NvMRomBlockNum' in the container 'NvMBlockDescriptor' is configured as <0>.

This warning occurs, if the value of the parameter NvMRomBlockDataAddress is configured when the value of the parameter NvMRomBlockNum is configured as <0> in the container NvMBlockDescriptor.

2) WRN020051

Value of the parameter 'NvMRomBlockDataAddress' should be configured since the value of the parameter 'NvMRomBlockNum' is other than <0> in the container 'NvMBlockDescriptor'.

This warning occurs, if the value of the parameter NvMRomBlockDataAddress is not configured when the value of the parameter NvMRomBlockNum is configured other than <0> in the container NvMBlockDescriptor.

3) WRN020053

Value of the parameter 'Parameter Name' is considered as <false/0>, when the value of the parameter 'Parameter Name1' is <NVM_BLOCK_DATASET> in the container 'NvMBlockDescriptor'.

This warning occurs, if the value configured for the parameters 'Parameter Name' is <true/1>, when the value of the parameter 'Parameter Name1' is configured as NVM_BLOCK_DATASET in the container NvMBlockDescriptor and the Generation Tool ignores the value of the parameter 'Parameter Name'.

Parameter Name1	Parameter Name
NvMBlockManagementType	NvMSelectBlockForReadAll
	NvMSelectBlockForWriteAll

7.2.1.3 Information Messages

1) INF020015

AUTOSAR Release version <Version> configured for the parameter 'AR-RELEASE-VERSION' in provided MDT file is not correct. AUTOSAR Release version should be one of the following: 4.0.3.

This information occurs, if the value of the element AR-RELEASE-VERSION present in the BSW Module Description template is configured other than 4.0.3.

2) INF020051

Value of the parameter 'Parameter Name' should not be configured as <true/1> for the block having block id <CRC Block Id>, when the block having block id <Main Block Id> is Main block and the Generation Tool resets the value of the parameter 'Parameter Name' to <false/0>.

This information occurs, if the value of the parameter 'Parameter Name' is configured as <true/1> for the block having block id <CRC Block Id>, when the block having block id <Main Block Id> is Main block and the Generation Tool resets the value of the parameter 'Parameter Name' to <false/0>.

Parameter Name
NvMBlockUseCrc
NvMBlockUseSyncMechanism
NvMStaticBlockIDCheck
NvMWriteVerification
NvMSelectBlockForWriteAll
NvMBlockUseSyncMechanism
NvMStaticBlockIDCheck
NvMWriteVerification
NvMCalcRamBlockCrc

3) INF020052

Value of the parameter 'NvMBlockManagementType' should be configured as <NVM_BLOCK_NATIVE> for the block having block id <CRC Block Id>, when the block having block id <Main Block Id> is Main block and the Generation Tool resets the value of the parameter NvMBlockManagementType to NVM_BLOCK_NATIVE.

This information occurs, if the value of the parameter NvMBlockManagementType is not configured as <NVM_BLOCK_NATIVE> for the block having block id <CRC Block Id>, when the block having block id <Main Block Id> is Main block and the Generation Tool resets the value of the parameter NvMBlockManagementType to <NVM_BLOCK_NATIVE>.

4) INF020053

Value of the parameter 'Parameter Name' should be configured as <0> for the block having block id <CRC block Id>, when the block having block id <Main Block Id> is Main block and the Generation Tool resets the value of the parameter 'Parameter Name' to <0>.

This information occurs, if the value of the parameter 'Parameter Name' is not configured as <0> for the block having block id <CRC Block Id>, when the block having block id <Main Block Id> is Main block and the Generation Tool resets the value of the parameter 'Parameter Name' to <0>.

Parameter Name
NvMMaxNumOfReadRetries
NvMMaxNumOfWriteRetries

5) INF020054

Value of the parameter 'NvMBlockUseCrc' should not be configured as <true/1>, when the value of the parameter NvMBlockJobPriority is configured as <0> in the container 'NvMBlockDescriptor' and the Generation Tool resets the value of the parameter 'NvMBlockUseCrc' to <false/0>.

This information occurs, if the value configured for the parameter NvMBlockUseCrc is <true/1>, when the value of the parameter NvMBlockJobPriority is configured as <0> and the Generation Tool resets the value of the parameter NvMBlockUseCrc to <false/0>.

6) INF020052

Value of the parameter 'NvMBlockManagementType' should be configured as <NVM_BLOCK_NATIVE> for the block having block id <Dataset Block Id>, when the block having block id <Main Block Id> is Main block and the Generation Tool resets the value of the parameter 'NvMBlockManagementType' to <NVM_BLOCK_NATIVE>.

This information occurs, if the value of the parameter 'NvMBlockManagementType' is not configured as <NVM_BLOCK_NATIVE> for the block having block id <Dataset Block Id>, when the block having block id <Main Block Id> is Main block and the Generation Tool resets the value of the parameter 'NvMBlockManagementType' to <NVM_BLOCK_NATIVE>.

8. SWP Error Code

8.1 SWP Error Code List

8.1.1 NVM_E_INTEGRITY_FAILED

ErrorId Symbol	NVM_E_INTEGRITY_FAILED
Description	<p>1. This error occurs when CRC is incorrect after reading wrong data because EEPROM is physically destroyed or SPI communication does not work.</p> <p>2. It occurs when only part of data is written after PowerOff (i.e., Reset) during writing. It also occurs if CRC is incorrect.</p>
Cause of error	H/W, SWP
Platform default Action	NO RESET
Functional impact	As this is an error that can occur during reading, users may

	gather wrong data if there is no error check.
Related module(s)	None
MCU	Common
Error type	Settings, code
How to apply to application	<p>Being an error that can occur during reading, it can be detected by making the state of a block NVM_REQ_INTEGRITY_FAILED before notifying to application through a callback or directly reading the block state on the app.</p> <p>Code: (Rather than handling DEM Report) If Request(Read) of the relevant block fails, it can be handled with corresponding logic (i.e., using default value or writing, etc.).</p> <p>Settings: When setting redundant blocks (two copies), this error is extremely unlikely. (It can occur when EEPROM is broken.)</p>

8.1.2 NVM_E_LOSS_OF_REDUNDANCY

ErrorId Symbol	NVM_E_LOSS_OF_REDUNDANCY
Description	This error occurs when reading redundant block data all fails because EEPROM is physically destroyed or SPI communication does not work.
Cause of error	H/W
Platform default Action	NO RESET
Functional impact	As this is an error that can occur during reading, users may gather wrong data if there is no error check.
Related module(s)	None
MCU	Common
Error type	Code
How to apply to application	<p>Being an error that can occur during reading, it can be detected by making the state of a block NVM_REQ_REDUNDANCY_FAILED before notifying to application through a callback or directly reading the block state on the app.</p> <p>Code: (Rather than handling DEM Report) If Read of the relevant block fails, it can be handled with corresponding logic (i.e., using default value or writing, etc.).</p> <p>※ NVM_E_LOSS_OF_REDUNDANCY occurs only in a block set as redundant. (It can occur when EEPROM is broken.)</p>

8.1.3 NVM_E_QUEUE_OVERFLOW

ErrorId Symbol	NVM_E_QUEUE_OVERFLOW
Description	This error occurs when pending jobs exceed a queue size and cannot be saved anymore.
Cause of error	ASW

Platform default Action	NO RESET
Functional impact	As this is an error that can occur during Read/Write, Read/Write may not work normally or users may gather wrong data if there is no error check.
Related module(s)	None
MCU	Common
Error type	Settings, code
How to apply to application	<p>(Rather than handling DEM Report) E_NOT_OK is returned API (Write/Read) Return value.</p> <p>Settings: This error does not occur if a queue size is set as many as the number of NvM Block.</p> <p>Code: If E_NOT_OK is returned, API must be called again at the next task.</p>

8.1.4 NVM_E_REQ_FAILED

ErrorId Symbol	NVM_E_REQ_FAILED
Description	This error occurs when Read/Write fails because EEPROM is physically destroyed or SPI communication does not work.
Cause of error	H/W
Platform default Action	NO RESET
Functional impact	As this is an error that can occur during Read/Write, Read/Write may not work normally or users may gather wrong data if there is no error check.
Related module(s)	None
MCU	Common
Error type	Code
How to apply to application	<p>Being an error that can occur during Read/Write, it can be detected by making the state of a block NVM_REQ_NOT_OK before notifying to application through a callback or directly reading the block state on the app.</p> <p>Code: (Rather than handling DEM Report) If Read/Write of the relevant block fails, it can be handled with corresponding logic (i.e., using default value or writing, etc.).</p>

8.1.5 NVM_E_VERIFY_FAILED

ErrorId Symbol	NVM_E_VERIFY_FAILED
Description	In case of setting Verify function, if the value matches after writing and then reading again, it is considered a success. Failing here generates this error, and it occurs when Read/Write fails because EEPROM is physically destroyed or SPI communication does not work.
Cause of error	H/W
Platform default Action	NO RESET

Functional impact	As this is an error that can occur during Write, users may gather wrong data if there is no error check and Write cannot work normally.
Related module(s)	None
MCU	Common
Error type	Settings, code
How to apply to application	<p>Being an error that can occur during Write, it can be detected by making the state of a block NVM_REQ_NOT_OK before notifying to application through a callback or directly reading the block state on the app.</p> <p>Settings: Designate the number of retry through NvMMaxNumOfWriteRetries.</p> <p>Code: Thus, (rather than handling DEM Report) if Write of the relevant block fails, it can be handled with corresponding logic (i.e., retry, etc.).</p>

8.1.6 NVM_E_WRITE_PROTECTED

ErrorId Symbol	NVM_E_WRITE_PROTECTED
Description	Through WriteBlock Protect setting on application, this error occurs when requesting Write without command over a block to lift a ban on writing.
Cause of error	ASW
Platform default Action	NO RESET
Functional impact	Write request is not accepted.
Related module(s)	None
MCU	Common
Error type	Code
How to apply to application	Code: Through NvM_SetBlockProtection API, request Write again after lifting Write Protect.

8.1.7 NVM_E_WRONG_BLOCK_ID

ErrorId Symbol	NVM_E_WRONG_BLOCK_ID
Description	<p>1. In case of setting IDCheck function, Write request saves Block ID in addition to data and compares Block ID saved when reading with the set Block ID. This error occurs when Block ID is incorrect after reading wrong data because EEPROM is physically destroyed or SPI communication does not work.</p> <p>2. It occurs because Block ID is incorrect when only part of data is written after PowerOff (i.e., Reset) during writing. Redundant blocks (two copies) can reduce this error.</p>
Cause of error	H/W, SWP
Platform default Action	NO RESET

Functional impact	As this is an error that can occur during reading, users may gather wrong data if there is no error check.
Related module(s)	None
MCU	Common
Error type	Settings, code
How to apply to application	<p>Being an error that can occur during Read, it can be detected by making the state of a block NVM_REQ_NOT_OK before notifying to application through a callback or directly reading the block state on the app.</p> <p>Code: (Rather than handling DEM Report) If Request(Read) of the relevant block fails, it can be handled with corresponding logic (i.e., using default value or writing, etc.).</p> <p>Settings: When setting redundant blocks (two copies), this error is extremely unlikely. (It can occur when EEPROM is broken.)</p>

9. Appendix

9.1 Function-specific Configuration Guide

9.1.1 Redundant Block Configuration (2 copies)

- 1) Generate a block at the NvM module.
- 2) Set Block Management Type as NVM_BLOCK_REDUNDANT and Nv Block Num as 2.
- 3) Generate two Ea / Fee blocks.
- 4) Length as equal, block Number as formula (n : NvMDataSetSelectionBits)
- 5) First Ea/Fee Block Number: $2n * NvMNvBlockBaseNumber$
- 6) Second Ea/Fee Block Number : $2n * NvMNvBlockBaseNumber+1$
- 7) Mapping Reference of NvM Block at the first Ea / Fee Block

9.1.2 CRC Implement

- 1) Set NvMBlockUseCrc and NvMCalcRamBlockCrc of a block that requires CRC as True. Set NvMBlockCrcType among CRC8, CRC16 and CRC32 in line with the design purpose of App.
- 2) Basically, the Underlayer Block (Fee/Ea Block) size needs to be the same as the length of NvM Block, but if CRC is added, the CRC size must be added to NvM Block Length.
- 3) Set an underlayer block size as +1 for CRC8, +2 for CRC16 and +4 for CRC32.
- 4) Immediate Block does not support CRC. (From SWS_NVM721) In other words, if BlockJobPriority is set as 0 in the NvM Block Descriptor configuration, Generator notifies the following information even when CRC is set and does not use CRC function (regardless of JobPriorization of Common Container). Value of the parameter 'NvMBlockUseCrc' should not be configured as <true/1>, when the value of the parameter 'NvMBlockJobPriority' is configured as <0> in the container 'NvMBlockDescriptor' and the Generation Tool resets the value of the parameter 'NvMBlockUseCrc' to <false/0>.

9.1.3 Immediate Block Configuration

- 1) Set NvMJobPrioritization as True in NvM Common Container.
- 2) Set NvMBlockJobPriority of a block to be Immediate as 0.
- 3) Modify setting value to prevent OverFlow in NvMSizeImmediateJobQueue.

※ If Immediate Block is used, the NvM module shall use two queues, one for immediate write jobs (crash data) another for all other jobs (including immediate read/erase jobs, standard jobs). Otherwise the NvM Module shall use one queue and processes all jobs in FCFS order.
NVRAM blocks with immediate priority are not expected to be configured to have a CRC.

9.1.4 ReadAll / WriteAll Configuration

- 1) Set NvMSelectBlockForReadAll / NvMSelectBlockForWriteAll of a block to be ReadAll/WriteAll as True.

ReadAll : A value saved in EEPROM is read to Ram during StartUp.

WriteAll : Ram value is written in EEPROM during ShutDown.

9.1.5 ReadAll / WriteAll Order Configuration

- 1) Configure the parameter NvMReadAllOrderSupport / NvMWriteAllOrderSupport to TRUE to change the order of block processing during ReadAll/Write stage

Path: [NvM \[NvM\]](#) > [NvMCommon \[NvMCommon\]](#)

Navigator

- NvM
 - Block Descriptor [11]
 - NvMCommon
 - NvMUserIncludeFiles
 - NvMDemEventParameterRefs_0

Set Ram Block Status Api*: ☐ false

Size Immediate Job Queue: 20

Size Standard Job Queue*: 20

Main Function Call Cycle: 0.005

Version Info Api*: ☐ false

Read All Order Support: ☒ true

Write All Order Support: ☒ true

Mainfunction Trigger Ref: OsAlarm_BSW_Me

- 2) If NvMReadAllOrderSupport / NvMWriteAllOrderSupport is TRUE, change the NvMReadAllOrder / NvMWriteAllOrder of the blocks user want to configure the order. However, since the block with BlockId of "1" cannot be configured. In case of blocks that do not require an order setting. It is not necessary to configure an order. For blocks that require ordering, order must start with "1" and be set to sequential with other orders.

Path: [NvM \[NvM\]](#) > [NvMBlock PrimaryEventMemory2 \[NvMBlockDescriptor1\]](#)

Navigator

- NvM
 - Block Descriptor [11]
 - NvMBlock_ConfigID
 - NvMBlock_ManagementBlock
 - NvMBlock_EventStatusNvRamBlock
 - NvMBlock_PrimaryEventMemory0
 - NvMBlock_PrimaryEventMemory1

Write Verification Data Size*: 18

DefaultROMCRCEnabled*: ☐ false

Read All Order*: 7

Write All Order*: 1

Target Block Reference 1 [1]

9.2 Cautions during Design

9.2.1 NvM Block Identifier

The NvM Block Ids are Expected to be in a sequential order. [NvM475]

1) NvM Block IDs 0 and 1 are blocks that are used by the NvM module itself and must not be used by App or other BSW modules.

2) Reserved NVRAM block IDs:

0 -> to derive multi block request results via NvM_GetErrorStatus

1 -> redundant NVRAM block which holds the configuration ID : NvMBlock_ConfigID

※ Application users need not additionally configure in deployed projects.

9.2.2 RamBlock Length

The length of both RamBlock and NvBlock must be equal.

9.2.3 Immediate Block

If requesting a write of an immediate block while processing a job (i.e., Write/Read) of a standard block, the job in progress is canceled to execute an immediate block write and then the canceled job is carried out again. If the standard job in progress is a write, it may be canceled while only part of the job is written, then the canceled write job is conducted again after processing an immediate block write. In case of reset during this process, EEPROM data of the job in progress may be left only partially written, and this error can be detected through CRC configuration.

9.2.4 Request Return Value and Checking Block Status

When requesting Read/Write, return value of API must be confirmed. If return value is E_OK, a request is registered normally in the queue of NvM, but if it is E_NOT_OK, an error occurs and the request is not registered. This error mostly occurs when a block requested is pending or a queue is full. Therefore, users must check the status of a block and confirm that it is not pending before requesting. In case of Async request such as Read/Write, it must be confirmed that the job is complete through polling or callbacks. If a reset occurs when the request is not finished, the job cannot be ensured its completion.

9.2.5 ReadAll Time

On the Hyundai Motors standard platform, the ReadAll function at the StartUpTwo stage is conducted, and when it is finished, it goes over to the StartUpThree stage for CAN communication. As the time required here varies depending on the number of ReadAll blocks, if the time specification from power on to CAN communication is exceeded, the number of the blocks must be cut. And as for the internal EEPROM, the virgin StartUpTwo time takes longer than when it is not virgin. This is the time for FeeBlock Init, and it changes depending on the number of Fee blocks. Many virgin Fee blocks may trigger Wdg reset, and this can be fixed by changing NvMMaxNumOfReadRetries configuration or adjusting Wdg time up to StartUpTwo.

9.2.6 WriteAll Time

On the Hyundai Motors standard platform, the WriteAll function is conducted after WdgM Delnit at the Shutdown Sequence stage. DelnitTimeOut can be set in WdgM configuration, and this value

generates Wdg reset when TimeOut occurs after WdgM Delnit. Therefore, WriteAll must be completed within DelnitTimeOut value. As the WriteAll running time varies depending the EEPROM type and the number of NvM blocks, users must check WriteAll time and apply it to DelnitTimeOut value.

9.2.7 NvM API Call-Context

API of NvM can be called through RTE in the application, or API of NvM can be directly called at the CDD. But it must not be called at the ISR (Interrupt Service Routine). This is because the API running time can be flexible according to conditions. It must not be called at the Lower Power Mode either. This is because it is designed to work only at the High Power Mode.

9.2.8 Fee Init of Virgin Internal EEPROM

In the case of the internal EEPROM, the virgin DFlash initialization time takes longer than when it is not virgin. And in the state of Internal EEPROM Virgin, Fee Init (i.e., in factory) must not be interrupted (i.e., reset). Therefore, Fee Init time must be ensured in the initial process. That time can be figured out by measuring the time taken from making DFlash virgin, power on to NvM_ReadAll callout.

9.2.9 Erase All When Changing Memory Layout

To prevent abnormal operation, Internal/External EEPROM must be erased when the memory layout is changed (adding, deleting, changing the length and CRC of NvM Block, etc.).

9.2.10 Mem_Integration_User Implementation

Path : Static_Code\Integration_Code\Integration_Mem\Usercode
File : Mem_Integration_User.h, Mem_Integration_User.c

Users may revise the functions below only.

9.2.10.1 MEM_WRITEALL_FAST_MODE

File : Mem_Integration_User.h


```

052:
053: /******
054:  * START : Only STD_ON or STD_OFF of the macros can be modified by user      *
055:  ******
056:
057: /* STD_ON : When WriteAll is called , mainFunctions of the NvM,Fee,Fls,Eep,Ea
058:  *           are called in the while-loop
059:  * STD_OFF : When WriteAll is called , mainFunctions of the NvM,Fee,Fls,Eep,Ea
060:  *           are called by the GPT or periodic task
061:  */
062:
063: /* CAUTION!!!
064:  *
065:  * WriteAll : Depending on the type of MCU,
066:  *           there may be a timing problems.
067:  * Default : STD_OFF
068:  *
069:  * please contact us.
070:  ******
071:
072:
073: #define MEM_WRITEALL_FAST_MODE      (STD_OFF)
074:
075:

```

STD_ON: The WriteAll running time can be reduced by calling NvM/Fee/Fls/Ea/Eep mainfunctions in the while-loop.

STD_OFF: NvM/Fee/Fls/Ea/Eep mainfunctions are invoked about every 1ms as in a normal operation.

9.2.10.2 User Callbacks

7) Mem_PostFeeInitCallback (Common)

Callbacks called before Fee Init

8) Mem_Cypress_IllegalStateCallback(Cypress amethyst/ artemis)

This is called in case of FEE initialization failure because of changed memory layout and other causes.

“Flash erase all” is the only recovery solution.

9) Mem_Infineon_IllegalStateCallback (Infineon Aurix)

This is called when FEE cannot perform normally. When this callback is called, FEE suspends its operation.

See the FEE manual and handle the matter case by case.

9.2.11 Garbage Collection

In the case of internal EEPROM (Flash), if one sector is full, valid blocks are moved to an empty sector in order to write the next block and then a write request is performed in the available space. Additionally, the existing full sector is deleted. This algorithm is generally called Garbage Collection (GC). Each vendor has a different term. Some vendors use swap or refresh.

GC takes longer than the write job. Hundreds of ms are taken for both moving available blocks and erasing sectors.

[Note]

Running time differs per MCU. See Fee/FIs manuals and MCU DataSheet for more details.

GC is usually conducted when there is no space during write operations.
MCUs below are exceptions.

(1) Aurix

Fee of Aurix MCU can select an execution time of GC.

Container Details - FeeIcxSpecificConfig	
Short Name*:	FeeIcxSpecificConfig
Threshold Value*:	1024
Max Block Count*:	100
Use Erase Suspend*:	<input type="checkbox"/> false
State Var Structure*:	Fee_StateVar
Un Config Block*:	FEE_UNCONFIG_BLOCK_IGNORE
Un Config Blk Overflow Handle*:	FEE_CONTINUE
Gc Restart*:	FEE_GC_RESTART_WRITE
Get Cycle Count Api*:	FEE_GC_RESTART_INIT
Erase All Enable*:	FEE_GC_RESTART_WRITE
Get Prev Data Api*:	<input type="checkbox"/> false

Configuration Gc Restart

FEE_GC_RESTART_INIT : GC can be executed during Fee initialization and writing.

FEE_GC_RESTART_WRITE : GC is executed during writing only.

(2) Cypress

As for Fee of Cypress MCU, GC may occur at the time of initialization.

Reset during block writing could damage data. To recover damaged data (previous data),
Fee performs recycling (the term recycling is used for Cypress.).

[Note]

※ During Fee initialization, recycling may delay the initialization.

※ See Cypress Fee UserGuide for more details.

9.2.12 NvMMainfunctionTriggerRef

If an alarm is not wanted for NvM, the configuration of an alarm for NvMMainfunctionTriggerRef can be deleted. But change request is necessary. If not configured, a periodic SetEvent API callout after Rte start is necessary to drive memory stack mainfunction.

Example)

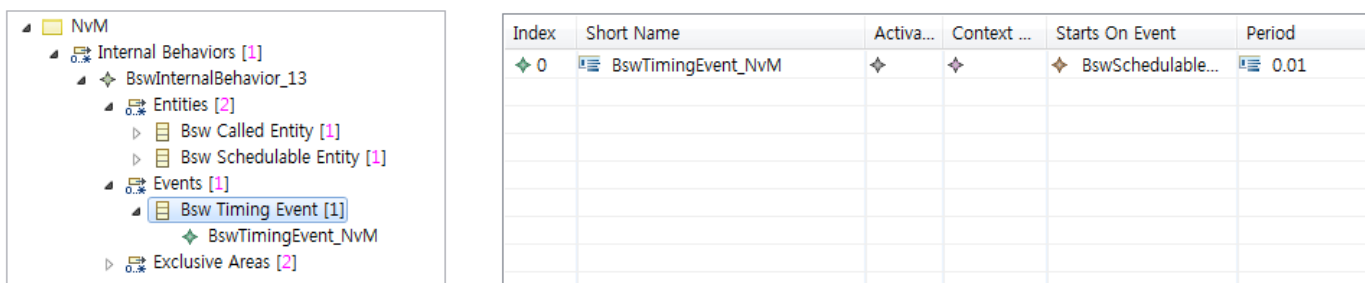
```
if (E_OK != SetEvent(OsTask_BSW_Mem_Process, OsEvent_BSW_Mem_Process))
{
    /* error */
}
```

9.3 Bswmd (Bsw Module Description)

9.3.1 MainFunction Period Configuration

The NvM module must periodically call MainFunction, and this is performed by mapping in TimingEvent.

Set a period in Bsw Timing Event of BswModuleDescription Container of NvM as below.
The period unit is second, and set 0.005 if the MainFunction period received from SRS is 5ms.



The screenshot shows the NvM configuration tree on the left and a table on the right. The tree structure is as follows:

- NvM
 - Internal Behaviors [1]
 - BswInternalBehavior_13
 - Entities [2]
 - Bsw Called Entity [1]
 - Bsw Schedulable Entity [1]
 - Events [1]
 - Bsw Timing Event [1]
 - BswTimingEvent_NvM
 - Exclusive Areas [2]

The table on the right shows the configuration for BswTimingEvent_NvM:

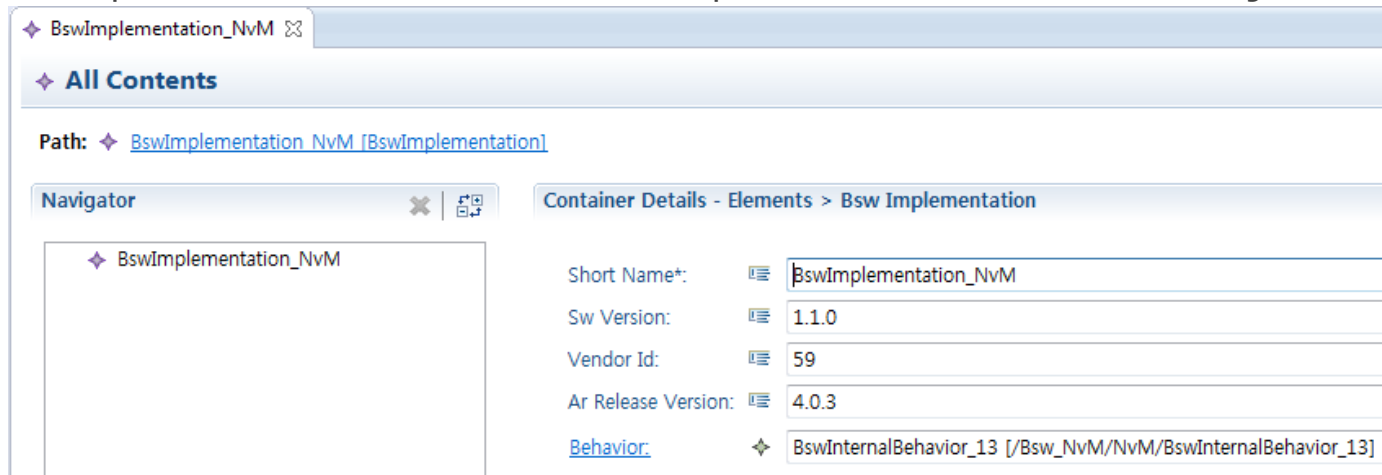
Index	Short Name	Activa...	Context ...	Starts On Event	Period
0	BswTimingEvent_NvM			BswSchedulable...	0.01

MainFunction to be set on Memory Stack is as follows:

1. NvM_MainFunction (both External and Internal EEPROM)

9.3.2 Bsw Module Version Setting

When compiling each module, incorrect version information triggers a compile error.
This requires version information revision in BswImplementation Container as the following Bswmd.



The screenshot shows the BswImplementation_NvM configuration window. The 'All Contents' tab is selected, and the 'Path' is set to 'BswImplementation_NvM [BswImplementation]'. The 'Navigator' pane shows the 'BswImplementation_NvM' element. The 'Container Details - Elements > Bsw Implementation' pane shows the following settings:

- Short Name*: BswImplementation_NvM
- Sw Version: 1.1.0
- Vendor Id: 59
- Ar Release Version: 4.0.3
- Behavior: BswInternalBehavior_13 [/Bsw_NvM/NvM/BswInternalBehavior_13]

9.4 Exclusive Areas

9.4.1 Module-specific SchM Apis

In order to provide data integrity of shared resources, Memory Module uses the scheduler service to enable and to disable data protection.

Following exclusive areas along with scheduler services are used to provide the protection:

Module	SchM APIs
NvM	SchM_Enter_NvM_RAM_INDEX() SchM_Exit_NvM_RAM_INDEX() SchM_Enter_NvM_RAM_STATUS_PROTECTION() SchM_Exit_NvM_RAM_STATUS_PROTECTION()
Fee (Bolero)	None
Fls (Bolero)	SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_00() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_00() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_01() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_01() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_02() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_02() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_03() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_03() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_04() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_04() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_05() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_05() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_06() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_06() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_07() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_07() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_08() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_08() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_09() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_09() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_10() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_10() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_11() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_11() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_12() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_12() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_13() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_13() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_14() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_14() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_15() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_15() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_16() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_16() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_17() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_17() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_18() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_18() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_19() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_19() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_20()

	SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_20() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_21() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_21() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_22() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_22() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_23() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_23() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_24() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_24() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_25() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_25() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_26() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_26() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_27() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_27() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_28() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_28() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_29() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_29() SchM_Enter_Fls_FLS_EXCLUSIVE_AREA_30() SchM_Exit_Fls_FLS_EXCLUSIVE_AREA_30()
--	---

9.4.2 Configuration Method

Add to Exclusive Areas of the module BswModuleDescription Container as follows.

The screenshot shows the NvM configuration interface. The path is: **NvM [BswModuleDescription]** > **BswInternalBehavior 13 [BswInternalBehavior]** > **RAM STATUS PROTECTION**.

Navigator

- NvM
 - Internal Behaviors [1]
 - BswInternalBehavior_13
 - Entities [2]
 - Events [2]
 - Exclusive Areas [2]
 - RAM_INDEX
 - RAM_STATUS_PROTECTION

Container Details - Exclusive Areas

Index	Short Name
0	RAM_INDEX
1	RAM_STATUS_PROTECTION

9.5 Normal and extended runtime preparation of NVRAM blocks

This subchapter is supposed to provide a short summary of normal and extended runtime Preparation of NVRAM blocks. The detailed behavior regarding the handling of NVRAM blocks during start-up is specified in chapter 8.3.3.1. (AUTOSAR_SWS_NVRAMManager.pdf)

Depending on the two configuration parameters `NvMDynamicConfiguration` and `NvMResistantToChangedSw` the NVRAM Manager shall behave in different ways during start-up, i.e. while processing the request `NvM_ReadAll()`.

If `NvMDynamicConfiguration` is set to **FALSE**, the NVRAM Manager shall ignore the stored configuration ID and continue with the normal runtime preparation of NVRAM blocks. In this case the RAM block shall be checked for its validity. If the RAM block content is detected to be invalid the NV block shall be checked for its validity. A NV block which is detected to be valid shall be copied to its assigned RAM block. If an invalid NV Block is detected default data shall be loaded.

If `NvMDynamicConfiguration` is set to **TRUE** and a **configuration ID mismatch** is detected, the extended runtime preparation shall be performed for those NVRAM blocks which are configured with `NvMResistantToChangedSw(FALSE)`. In this case default data shall be loaded independent of the validity of an assigned RAM or NV block.

In other words, when `DynamicConfiguration` in the common container is true, `BlockID #1` is first read during `ReadAll`, and if this value and the set `NvMCompiledConfigId` match, `ReadAll` is executed.

If they do not match and the setting value of `NvMResistantToChangeSw` that can be set for each block is false, it is processed as failure regardless of read success. (If there is default data, the data is copied to `RamBlock`.)

If a block whose setting value of `NvMResistantToChangeSw` is true,

`ReadAll` is conducted regardless of its match with `NvMCompiledConfigId`. But writing `NvMCompiledConfigId` on `BlockID #1` must be performed in Application through either `Write API` or `WriteAll`.

9.6 Notification Interface with Application

9.6.1 SingleBlock Callback

In the NvM module, each block supports `SingleBlock Callback`, and when in use, set in `SingleBlockCallback` of `Block Descriptor` as follows. (See Chap 5.1.2)

```
Rte_Call_NvM_PNJF_{Block}_JobFinished
Block = {ecuc(NvM/NvMBlockDescriptor.SHORT-NAME)}
```

※ See Chap 8.7 to change as above a project whose naming rule is set as `Rte_Call_NvM_PNx_JobFinished`.

In case of the setting above, the created NvM Swcd generates ports and interfaces as follows:

```
PNJF_{Block}
```

```
Block = {ecuc(NvM/NvMBlockDescriptor.SHORT-NAME)}
```

```
ClientServerInterface NvMNotifyJobFinished {
```

```
    JobFinished( IN uint8 ServiceId, IN NvM_RequestResultType JobResult);
```

```
};
```

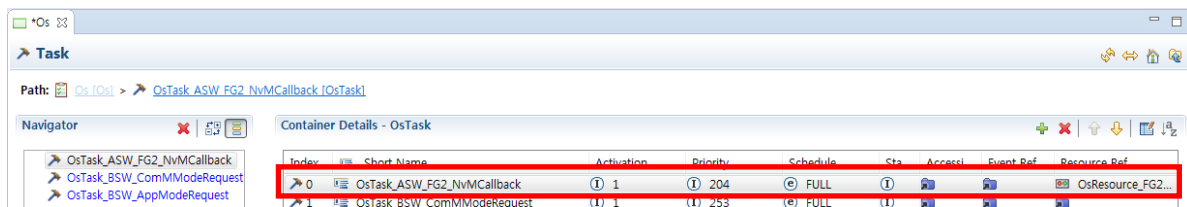
ServiceID is ServiceID of NvM Function (see each function description in Chap 6.3), and NvM_RequestResultType is the result of Request (see Chap 6.1.1).

This can be connected to ApplicationSwComponent in Application. (See AUTOSAR BSW Service API Guide.doc for prototypes of RTE-based functions.)

As for Runnable related to callbacks that are set in Application Sw Component, canBeInvokedConcurrently must be set as FALSE on the basis of NvM SWS [NVM736].

To prevent canBeInvokedConcurrently, a task must be activated to call a callback. Therefore, one task is necessary to use SingleBlockCallback, and this must be directly generated in Application as follows:

1. Generate Application SW Component and register Runnable of SingleBlockCallback.
2. Generate Task in OS.

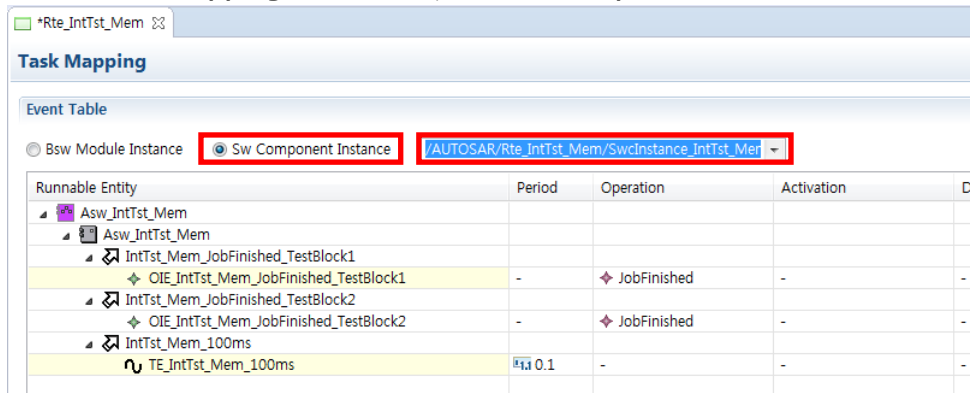


ShortName : OsTask_ASW_FG2_NvMCallback

Priority : Adjust bigger than FG1 and smaller than FG3.

ResourceRef : FG2

3. Under Task Mapping tab of RTE, set SW Component Instance and select the set App SW-C.



4. Task Mapping on Callback Runnable

Event Table

● Bsw Module Instance ● Sw Component Instance /AUTOSAR/Rte_IntTst_Mem/SwcInstance_IntTst_Mer

Runnable Entity	Period	Operation	Activation
Asw_IntTst_Mem			
IntTst_Mem_JobFinished_TestBlock1			
③ OIE_IntTst_Mem_JobFinished_TestBlock1	-	◆ JobFinished	-
IntTst_Mem_JobFinished_TestBlock2			
OIE_IntTst_Mem_JobFinished_TestBlock2	-	◆ JobFinished	-
IntTst_Mem_100ms			
TE_IntTst_Mem_100ms	0.1	-	-

④ Add Remove

OsTask Mapping Table

● Related to Bsw Module ① ● Related to Sw Component ● unMapped ② OsTask_ASW_FG2_NvMCallback

Position	RteEvent	Software Compone	Runnable Entity	Operation	Period	Offset
⑤ 100	◆ OIE_IntTst_Mem...	Asw_IntTst_Mem	IntTst_Mem_Job...	◆ JobFinished	-	

- ① Select unMapped for the first mapping and RelatedToSwComponent after that
- ② Select OsTask_ASW_FG2_NvMCallback
- ③ Select Runnable for Mapping
- ④ Add Click
- ⑤ Check the generation on OsTaskMappingTable

When setting Event related to SingleBlockCallback, set MappedToTaskRef as OsTask_ASW_FG2_NvMCallback as follows (RTE > Sw ComponentInstance > EventToTaskMapping).

Navigator

- Rte_IntTst_Mem
 - Sw Component Instance [1]
 - SwcInstance_IntTst_Mem
 - Event To Task Mapping [3]
 - RteEventToTaskMapping_OIE_I
 - RteEventToTaskMapping_OIE_J
 - RteEventToTaskMapping0

Container Details - RteEventToTaskMapping

Short Name*: RteEventToTaskMapping_OIE_IntTst_Mem_JobFinished_TestBlock1 Edit

Immediate Restart*: false

Position In Task: 100

Event Ref*: OIE_IntTst_Mem_JobFinished_TestBlock1 /ARPackage_IntTst_Mem/Asw_IntTst_Mem/SwcInstance_IntTst_Mem/OIE_IntTst_Mem_JobFinished_TestBlock1 Browse...

Mapped To Task Ref*: OsTask_ASW_FG2_NvMCallback [/Os/OsTask_ASW_FG2_NvMCallback] (/TestProject_RTU_Mem_mpc5607b/t Browse...

※ As SingleBlockCallback Runnable is called from FG2 Task, its running time must be a minimum.

9.6.2 InitBlock Callback

In the NvM module, each block supports InitBlockCallback, and when in use, set in InitBlockCallback of Block Descriptor as follows. (See Chap 5.1.2)

```
Rte_Call_NvM_PNIB_{Block}_InitBlock
Block = {ecuc(NvM/NvMBlockDescriptor.SHORT-NAME)}
```

※ See Chap 8.7 to change as above a project whose naming rule is set as Rte_Call_NvM_PNx_InitBlock.

In case of the setting above, the created NvM Swcd generates ports and interfaces as follows:

```
PNIB_{Block}
Block = {ecuc(NvM/NvMBlockDescriptor.SHORT-NAME)}

ClientServerInterface NvMNotifyInitBlock {
    InitBlock();
};
```

This can be connected to ApplicationSwComponent in Application. (See AUTOSAR BSW Service API Guide.doc for prototypes of RTE-based functions.)

9.7 Edit Cases following Port Name Change

9.7.1 InitBlock, SingleBlock Callback Name Edit

Callback name edit in Ecud_NvM.arxml or Ecud_NvM_IntTst_Mem.arxml files
 Rte_Call_NvM_PNx_InitBlock => Rte_Call_NvM_PNIB_{Block}_InitBlock
 Rte_Call_NvM_PNx_JobFinished => Rte_Call_NvM_PNjf_{Block}_JobFinished
 Block = {ecuc(NvM/NvMBlockDescriptor.SHORT-NAME)}

9.7.2 Generate.py Edit

Add P option when generating Swcd of NvM as the following image.

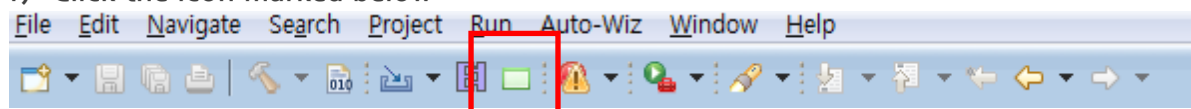
```
# GenerateNvM_S #####
moduleName = 'NvM'
Swcd_Bsw_NvM = os.path.join('swcd', 'Swcd_Bsw_NvM.arxml')

Swcd_Bsw_NvM = env.GenerateBSW(
    target=Swcd_Bsw_NvM,
    source=[
        Ecud_NvM_TC0, Ecud_Fee, Ecud_Dem, Bswmd_NvM,
        # Ecud_NvM_IntTst_Mem,
        # Bswmd_Mem.arxml Ecud_Fls.arxml Ecud_NvM.arxml Ecud_NvM_IntTst_Mem.arxml
    ],
    BSW_GENERATOR_NvM,
    BSWDEFINES=['S', 'P']
)

Alias('Generate' + moduleName + '_S', Swcd_Bsw_NvM)
#####
```

9.7.3 Port Connection Edit under EcucValueCollection Tap


1) Click the icon marked below



2) Click Service and I/O Tap

Ports in yellow are not connected, and ConfigID and diagnosis blocks do not connect ports.

3) Click the + button on ports to connect

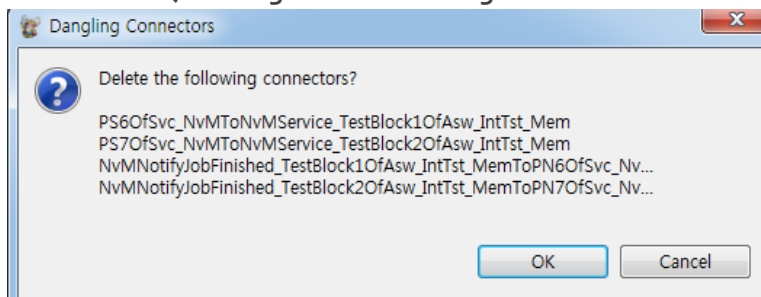


Contents	Context Component	Port Interface	Component Type	Connector
Dem	-	-	-	-
Det	-	-	-	-
EcuM	-	-	-	-
Pm	-	-	-	-
SWC_FIM	-	-	-	-
Svc_IoHwAb	-	-	-	-
Svc_NvM	-	-	-	-
PS_NvMBlock_ConfigID	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-
PS_NvMBlock_DemNonVolatileData	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-
PS_NvMBlock_DemPrimaryEventMemoryEnt	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-
PS_NvMBlock_DemPrimaryEventMemoryEnt	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-
PS_NvMBlock_DemPrimaryEventMemoryEnt	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-
PS_NvMBlock_IntTst_TestBlock1	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-
PS_NvMBlock_IntTst_TestBlock2	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-
PS_NvMBlock_IntTst_TestBlock3	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-

4) Uncheck Respect Naming Rule and check ports to connect

Svc_NvM	-	-	-	-
PS_NvMBlock_ConfigID	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-
PS_NvMBlock_DemNonVolatileData	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-
PS_NvMBlock_DemPrimaryEventMemoryEnt	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-
PS_NvMBlock_DemPrimaryEventMemoryEnt	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-
PS_NvMBlock_DemPrimaryEventMemoryEnt	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-
PS_NvMBlock_IntTst_TestBlock1	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-
NvMService_TestBlock1	Asw_IntTst_Mem	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	PS_NvMBlock_IntTst_TestBlock1OfSv
PS_NvMBlock_IntTst_TestBlock2	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-
NvMService_TestBlock2	Asw_IntTst_Mem	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	PS_NvMBlock_IntTst_TestBlock2OfSv
PS_NvMBlock_IntTst_TestBlock3	Svc_NvM	NvMService [/Svc_NvM/NvMS...	NvM [/Svc_NvM/NvM]	-

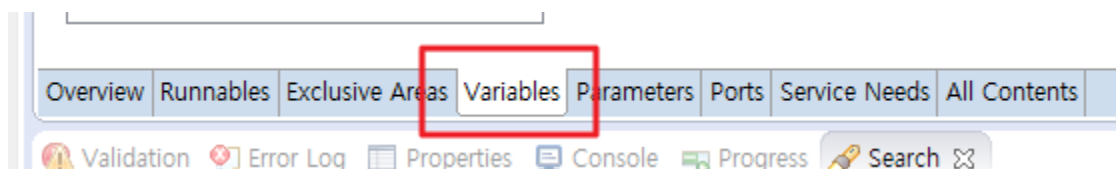
5) Click Assembly Connectors tab on EcuComposition of EcuExtract file and delete existing connectors (Clicking OK in the image below automatically performs.)



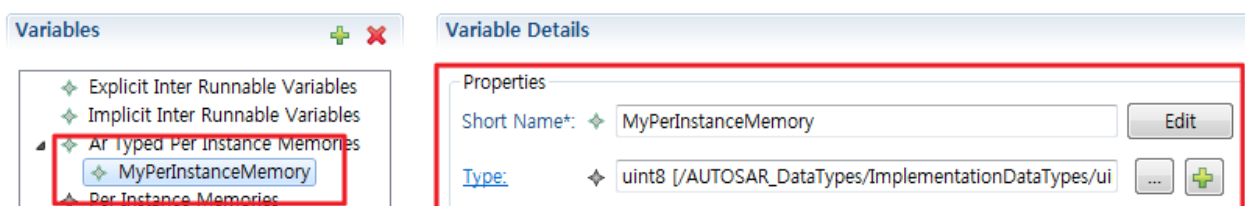
6) Execute EcuExtract again

9.8 PIM (PerInstanceMemory) Configuration Method

9.8.1 ArTypedPerInstanceMemory Addition



1) Move to Variables in Software Component where to use EEPROM Data



2) Input ShortName and Type (Implementation Data Type)

3) If building after saving, API or Rte_Pim_<ShortName>(void) is available right away in

Application, and Return becomes a variable address.

- 4) The variable is created in Rte.c and declared extern in Rte_<SWC>_PerInstanceMemory_CDS.h. If direct access to the variable is necessary as NvM, this can be included to be used.
- 5) The variable form is Rte_G<Variable Type><PerInstanceMemory Name>_<SWC Name>_PIMA. “_PIMA” at the end is ArTypedPerInstanceMemory.
- 6) A primitive variable type includes c: character, s: short, l:long, f:float, d: double, and u is attached to the front for unsigned, s for signed and uc for unsigned char. aa for an array type and st for a structure/union type.

ex) Rte_GstPerInstanceMemory_Rec_ApplicationSwComponentType_0_PIMC)

9.8.2 In case of using ReadAll/WriteAll with PIM (PerInstanceMemory)

- 1) In NvM, the name of a variable declared extern in Rte_<SWC>_PerInstanceMemory_CDS.h is input at Ram Block Data Address of Block Descriptor.

Nv Block Num*:	<input type="text" value="1"/>
Nvram Block Identifier*:	<input type="text" value="6"/>
Nvram Device Id*:	<input type="text" value="0"/>
Ram Block Data Address:	<input type="text" value="Rte_GucMyPerInstanceName_ApplicationSwComponentType0_PIMA"/>
Resistant To Changed Sw*:	<input type="checkbox"/> false
Rom Block Num*:	<input type="text" value="0"/>
Select Block For Read All*:	<input type="checkbox"/> false
Select Block For Write All*:	<input type="checkbox"/> false

- 2) For Header File Inclusion, Rte_<SWC>_PerInstanceMemory_CDS.h is added to NvM > NvMCommon > NvMUserIncludeFiles as below.

