

SCOPE OF APPLICATION All Project/Engineering	HYUNDAI AutoEver	SHT/SHTS 1 / 74
Responsibility: 클래식오토사팀	AUTOSAR NvM User Manual	DOC. NO
AUTOSAR NvM User Manual		

Document Change Histroy				
Date (YYYY-MM-DD)	Ver.	Editor	Chap	내용(개정 전 -> 개정 후)
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	• 형식변경 및 설정 Defalut값 변경 • ReadAll Time 추가 • WriteAll Time 추가
2016-05-31	1.2	CY Song	Chap 4.3 Chap 9.1.2	• Change Log 추가 • 4)번 주의사항 추가
2016-06-28	1.3	CY Song	Chap 4.3.1.1	• Change Log 추가
2016-07-20	1.4	CY Song	Chap 4.3 Chap 9.6.3	• Change Log 삭제 • Callback 설정시 주의사항 추가
2016-08-08	1.5	CY Song	Chap 5.1.1 Chap 5.2.1 Chap 9.5	• NvMDatasetSelectionBits, NvMDynamicConfiguration, NvMJobPrioritization 항목 수정 • FeeSetModeSupported 항목 수정 (value) • 요약 설명 추가
2016-08-12	1.6	CY Song	Chap 5.1.2 Chap 9.1.3 Chap 9.2.9	• 2)NvMBlockJobPriority Std Block 설명 수정 • Immediate Block 설명 추가 • Chap 8.2.9 추가
2016-09-22	1.7	CY Song	Chap 4.3.1.1 Chap 5.1.1	• Change Log 추가 • 3) 설명 추가
2016-10-14	1.8	CY Song	Chap 4.3.1.1 Chap 5.1.1 Chap 9.2.10	• Change Log 추가 • Dynamic Configuration Category 변경 • Virgin Internal EEPROM의 Fee Init 시 추가
2016-11-02	1.9	CY Song	Chap 4.3.1.1 Chap 9.2.2 Chap 9.6	• Change Log 추가 • RH850 Internal EEPROM Size 계산방법 추가 • Application Callback 설정방법 추가
2016-12-06	1.10	CY Song	Chap 4.3.1.1 Chap 5.1.1	• Change Log 추가 • NvMMultiBlockCallback 항목을 Changeable 로 수정
2017-06-30	1.11	YJ Yun	Chap 4.3.1.1 Cahp 9.8	• Change Log 추가 • External EEPROM size 관련 내용 삭제 • Document layout 수정
2017-09-27	1.12	YJ Yun	Chap 4.3.1.1	• Change Log 추가
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	• Change Log 추가 • Internal EEPROM size 수정 • Memory Layout 변경시 Erase All • Mem_Integration_User 추가

26th Edition Date: 18, 08, 2022	File Name NvM_UM.pdf	Creation SH Park 2022/06/15	Check JH Cho 2022/06/15	Approval JH Jung 2022/06/15
Document Management System				

2019-05-09	1.14	YJ Yun	Chap 8 Chap 9.2.11	<ul style="list-style-type: none"> • Dem Event Description 추가 • Garbage Collection 주의사항 추가 • EEPROM 사이즈 계산식을 Application Guide로 이 동함
2019-03-28	1.15	YJ Yun	Chap 4.3.1.1	<ul style="list-style-type: none"> • ChangeLog 추가
2019-08-20	1.16	YJ Yun	Chap 4.3.1.1	<ul style="list-style-type: none"> • ChangeLog 추가 • Appendix NvMMainfunctionTriggerRef 추가
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"> • ChangeLog 추가 • CddIf 추가 • NvMUserJobFunction 설명 추가
2019-10-17	1.18	YJ Yun	Chap 5 Chap 4.3.1.1	<ul style="list-style-type: none"> • 설정 카테고리 수정 • ChangeLog 추가
2020-12-31	1.5.1.0	YJ Yun	Chap 5 Chap 4.3.1.1	<ul style="list-style-type: none"> • 설정 카테고리 수정 • ChangeLog 추가
2021-01-20	1.5.2.0	YJ Yun	Chap 4.3.1.1	<ul style="list-style-type: none"> • ChangeLog 추가
2021-02-01	1.5.3.0	YJ Yun	Chap 4.3.1.1	<ul style="list-style-type: none"> • ChangeLog 추가
2021-12-30	1.5.4.0	JH Lim	Chap 4.3.1.1	<ul style="list-style-type: none"> • ChangeLog 추가
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 추가/삭제 ERR020076, ERR020078, ERR020079 추가 WRN020052, WRN020054 삭제 • ChangeLog 추가 • Fee configuration guide 추가 • SetRamBlockStatus 함수 설명 추가
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"> • ChangeLog 추가 • ReadAll/WriteAll Order Supporting 기능 설명 추 가 • Generator validation message 추가/수정 - ERR020068, ERR020069, ERR020070 수정 - ERR020080 추가 • ReadAll/WriteAll Order Supporting 설정 예시 추 가
2022-08-19	1.6.2.0	YJ Yun	Chap 4.3.1.1	<ul style="list-style-type: none"> • ChangeLog 추가
2022-12-01	1.6.2.1	YJ Yun	Chap 4.3.1.1	<ul style="list-style-type: none"> • ChangeLog 추가
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"> • ChangeLog 추가 • Common Container에 NvMUserWdgToggleFunction 추가 • NvMUserWdgToggleFunction 설명 추가

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	18
4.3.1.3 Deviation.....	18
5. CONFIGURATION GUIDE	19
5.1 NvM 모듈	19
5.1.1 Common Container	19
5.1.2 NvMBlockDescriptor 설정.....	21
5.1.3 NvMDemEventParameterRefs 설정.....	23
5.2 FEE 모듈 (Internal EEPROM)	23
5.2.1 FeeGeneral Container	23
5.2.2 BlockConfiguration 설정.....	24
5.2.3 FeeIcxSpecificConfig (Only Aurix).....	24
5.2.4 Fee Information	25
5.2.5 Redundant Block 분산 배치 (Only Chorus/Bolero).....	25
5.3 FLS 모듈 (Internal EEPROM)	25
5.3.1 FlsGeneral Container	25
5.3.2 FlsConfigSet	25
5.3.3 FlsDemEventParameterRefs	26
5.3.4 FlsSectorList.....	26
5.4 System Configuration	27
5.4.1 ApplicationSwComponentType 설정	27
5.4.2 CompositionSwComponentType 설정	28

6. APPLICATION PROGRAMMING INTERFACE (API)	29
6.1 Type Definitions	29
6.1.1 NvM_RequestResultType	29
6.2 Macro Constants	30
6.3 Functions	30
6.3.1 Initialization	30
6.3.2 Synchronous Requests	31
6.3.2.1 NvM_SetDataIndex	31
6.3.2.2 NvM_GetDataIndex	32
6.3.2.3 NvM_SetBlockProtection	32
6.3.2.4 NvM_GetErrorStatus	33
6.3.2.5 NvM_GetVersionInfo	34
6.3.2.6 NvM_SetRamBlockStatus	35
6.3.2.7 NvM_SetBlockLockStatus	36
6.3.3 Asynchronous Single Block Requests	36
6.3.3.1 NvM_ReadBlock	36
6.3.3.2 NvM_WriteBlock	37
6.3.3.3 NvM_RestoreBlockDefaults	38
6.3.3.4 NvM_EraseNvBlock	38
6.3.3.5 NvM_CancelWriteAll	39
6.3.3.6 NvM_InvalidateNvBlock	40
6.3.3.7 NvM_CancelJobs	40
6.3.4 Asynchronous Multi Block Requests	41
6.3.4.1 NvM_ReadAll	41
6.3.4.2 NvM_WriteAll	41
6.3.5 Callback Notifications	42
6.3.5.1 NvM_JobEndNotification	42
6.3.5.2 NvM_JobErrorNotification	42
6.3.6 Scheduled Functions	43
6.3.6.1 NvM_Mainfunction	43
6.3.7 CddIf	44
6.3.7.1 NvM_CddGetStatus	44
6.3.7.2 NvM_UserJobFunction	44
6.3.8 참고사항	46
6.3.8.1 In Communication with application SW-C	46
7. GENERATOR	47
7.1 Generator Option	47
7.1.1 NvM	47
7.2 Generator Error Message	47
7.2.1 NvM	47
7.2.1.1 Error Messages	47
7.2.1.2 Warning Messages	55
7.2.1.3 Information Messages	56
8. SWP ERROR CODE	57

8.1	SWP Error Code List	57
8.1.1	NVM_E_INTEGRITY_FAILED	57
8.1.2	NVM_E_LOSS_OF_REDUNDANCY	58
8.1.3	NVM_E_QUEUE_OVERFLOW	58
8.1.4	NVM_E_REQ_FAILED	59
8.1.5	NVM_E_VERIFY_FAILED	59
8.1.6	NVM_E_WRITE_PROTECTED	60
8.1.7	NVM_E_WRONG_BLOCK_ID	60
9.	APPENDIX	61
9.1	기능별 설정 Guide	61
9.1.1	Redundant Block 설정(2 copies)	61
9.1.2	CRC Implement	61
9.1.3	Immediate Block 설정	61
9.1.4	ReadAll / WriteAll 설정	61
9.1.5	ReadAll / WriteAll Order Supporting 설정	61
9.2	설계시 유의사항	62
9.2.1	NvM Block Identifier	62
9.2.2	RamBlock Length	62
9.2.3	Immediate Block	63
9.2.4	Request 시 return 값 및 Block 상태 확인 여부	63
9.2.5	ReadAll Time	63
9.2.6	WriteAll Time	63
9.2.7	NvM API Call-Context	63
9.2.8	Virgin Internal EEPROM 의 Fee Init	63
9.2.9	Memory Layout 변경시 Erase All	63
9.2.10	Mem_Integration_User Implementation	64
9.2.10.1	MEM_WRITEALL_FAST_MODE	64
9.2.10.2	User Callbacks	64
9.2.11	Garbage Collection	64
9.2.12	NvMMainfunctionTriggerRef	65
9.3	Bswmd (Bsw Module Description)	66
9.3.1	MainFunction 주기 설정	66
9.3.2	Bsw 모듈 version 설정	66
9.4	Exclusive Areas	66
9.4.1	모듈별 SchM Apis	66
9.4.2	설정방법	68
9.5	Normal and extended runtime preparation of NVRAM blocks	68
9.6	Notification Interface with Application	69
9.6.1	SingleBlock Callback	69
9.6.2	InitBlock Callback	71
9.7	Port Name 변경에 따른 수정건	71
9.7.1	InitBlock, SingleBlock Callback Name 수정	71
9.7.2	generate.py 수정	71

9.7.3	EcucValueCollection Tap 에서 Port 연결 수정	72
9.8	PIM (PerInstanceMemory) 설정방법	73
9.8.1	ArTypedPerInstanceMemory 추가.....	73
9.8.2	PIM (PerInstanceMemory) 을 이용하여 ReadAll/WriteAll 을 사용할 경우	73

1. Overview

본 문서는 Autosar 표준 SRS/SWS 를 기반으로 작성 되었으며. 모듈 사용시 보다 자세한 기능적인 설명이 필요한 경우 아래 Reference 문서를 참고한다. 또한 Fee, Fls 의 자세한 설정은 각 MCU 에서 지원하는 UserManual, Integration Manual 을 참고한다.

설정관련 Category 의 해석은 다음과 같다.

- Changeable (C): User 에 의해서 설정 가능한 항목
- Fixed (F): User 에 의한 변경이 불가능한 항목
- NotSupported (N): 사용되지 않는 항목

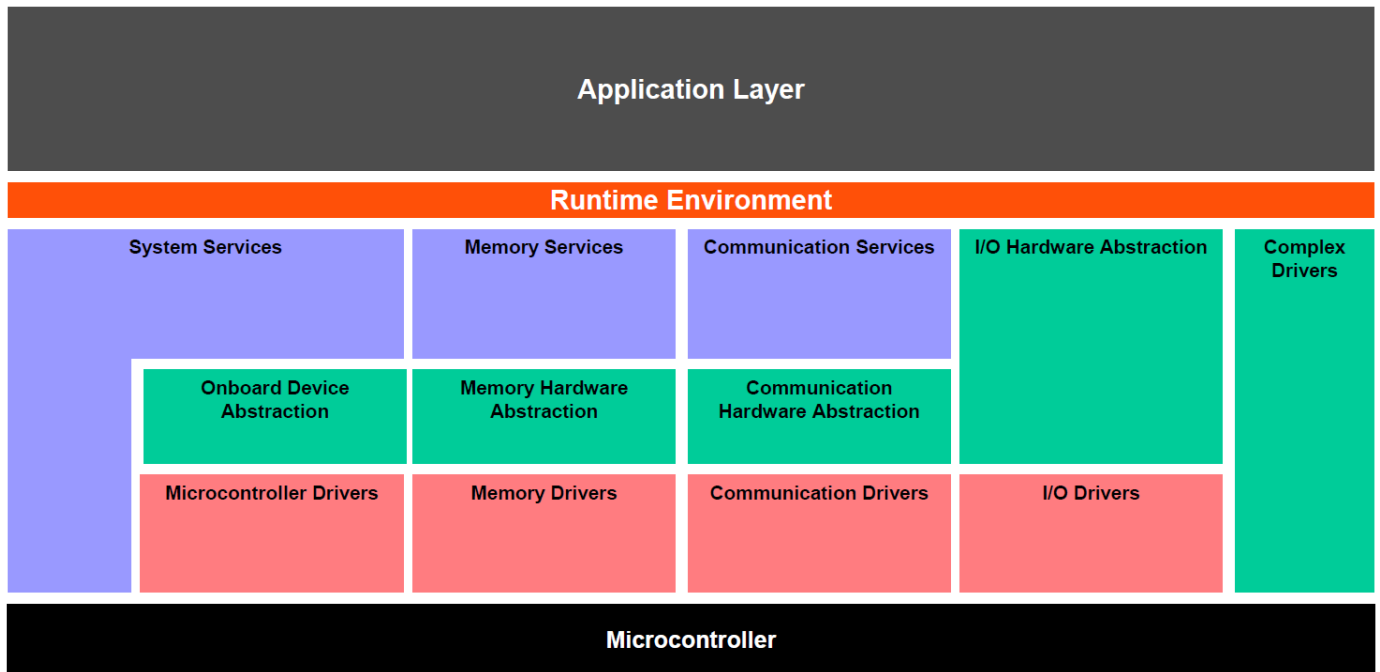
2. Reference

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

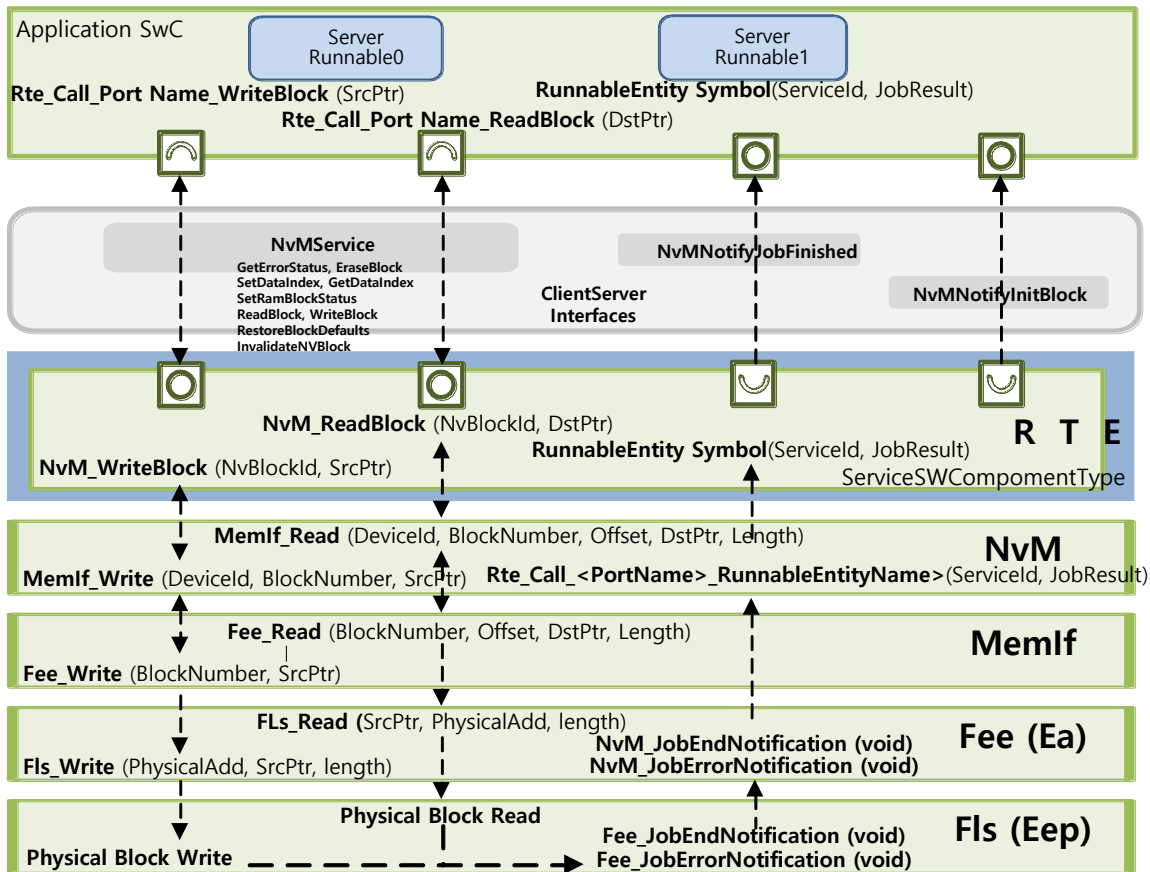
3.1 Overview of Software Layers

AUTOSAR 플랫폼의 Layered Architecture 는 아래와 같다. AUTOSAR 플랫폼은, Service Layer, ECU Abstraction Layer, Complex Device Drivers 및 Microcontroller Abstraction Layer 로 구분될 수 있다.



3.2 AUTOSAR Memory Stack

MemoryStack 모듈은 NvM / MemIf / Ea / Eep / Fee / Fls 모듈을 의미한다. 여기서 Fee 와 Fls 모듈은 Mcal 에 속하는 모듈로 기본적인 사항은 각 MCU 제조사의 UserManual, Integration Manual 을 참조한다. EEPROM 을 사용하기 위하여 기본적으로 NvM, MemIf 모듈이 필요하며, Internal EEPROM 을 사용하기 위해서는 Fee / Fls 모듈, External EEPROM 을 사용하기 위해서는 Ea / Eep 모듈이 추가로 필요하다. EEPROM 을 사용하기 위한 Autosar layer 및 각 모듈간의 interface 는 다음과 같다.



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

이 Chapter에서는, NvM Product에 대한 release 관련 내용을 제공하는데 목적이 있으며, NvM Software product release version에 대한, 제한사항 및 특이사항을 기술하고 있다.

4.2 Scope of the release

이 문서에 대한 모든 내용은, 다음의 NvM 모듈에 한정한다.

Module	Autosar version	SWS version	Module version
NvM	4.0.3	3.2.0	1.7.0

※ Module version은 각 모듈의 BswModule Description(Bswmd)파일의 Sw version을 의미한다.

4.3 Module release notes

4.3.1 NvM

4.3.1.1 Change Log

➤ Version 1.7.0.0 (2022-12-20)

- 신규 기능

■ 초기화 시 NvMUserWdgToggleFunction Callout 추가

원인	External Watchdog을 사용하는 경우, 메모리 초기화 시 메모리 사이즈가 크거나 초기화 로직의 수행시간이 길어 Reset이 발생할 수 있다. 이런 경우를 방지하기 위해 사용자가 External Watchdog을 적절한 주기로 trigger시킬 수 있도록 User Watchdog Toggle callout을 추가한다.
동작 영향	사용자가 NvMUserWdgToggleFunction를 설정하면 초기화가 완료되기 전까지 Mem_EalnitPerform, Mem_FeelnitPerform에서 User 함수가 반복적으로 수행된다.
설정 영향	NvMCommon/ NvMUserWdgToggleFunction
ASW 조치 사항	사용자가 NvMUserWdgToggleFunction 설정 시 호출되는 함수를 구현해야 한다.

※ 해당 기능은 External Watchdog을 사용하고 Timer 설정 시간이 짧아, 메모리 초기화 완료 전 Reset이 발생하는 경우에만 사용해야 한다. 일반적으로 플랫폼에서 제공하는 Watchdog 기능 사용을 권장한다.

- 개선 사항

■ N/A

➤ Version 1.6.2.1 (2022-12-01)

- Task

■ User Manual 수정

원인	영문 매뉴얼 추가
동작 영향	없음
설정 영향	없음
ASW 조치 사항	없음

➤ Version 1.6.2.0 (2022-08-20)

- 신규 기능

■ N/A

- 개선 사항

■ UNECE Cyber Security 법규 대응을 위한 보안 코딩 추가 개선

원인	UNECE Cyber Security 법규 대응을 위한 보안 코딩 추가 개선
동작 영향	없음
설정 영향	없음
ASW 조치 사항	없음

➤ Version 1.6.1.0 (2022-06-15)

- 신규 기능

■ ReadAll / WriteAll 동작 시 Block 처리 순서를 사용자가 설정하도록 기능 추가

원인	ReadAll / WriteAll 동작 시 처리되는 Block 순서를 사용자가 설정 가능하도록 기능 제공
동작 영향	해당 기능이 Enable 된 경우 사용자가 정의한 Block ID 순서대로 Read/Write 기능 수행
설정 영향	NvMReadAllOrderSupport, NvMWriteAllOrderSupport, NvMReadAllOrder, NvMWriteAllOrder
ASW 조치 사항	5.1.1 Common Container, 5.1.2 NvMBlockDescriptor 설정 참조

- 개선 사항

■ N/A

➤ Version 1.6.0.0 (2022-03-28)

- 신규 기능

■ N/A

- 개선 사항

■ Fee 설정 가이드 추가

원인	사용자 Technical Review 수행 시간 개선 필요
동작 영향	UM 에 Technical Review 항목을 반영. Bolero/Chorus MCAL Fee 의 설정 가이드를 추가.
설정 영향	없음.
ASW 조치 사항	5.2.5 Redundant Block 분산 배치 (Only Chorus/Bolero) 참조

■ Generator Validation Rule 개선

원인	사용자 Technical Review 시간 개선 필요
동작 영향	Generator Validation Rule 에 Technical Review 항목을 반영. 1. CRC 적용시 설정 항목. 2. ReadAll/WriteAll 사용시 RamBlockAddress 설정 항목.
설정 영향	없음.
ASW 조치 사항	없음.

■ SetRamBlockStatus 함수 개선

원인	AUTOSAR 4.0.3 사양에 따라 NvMSetRamBlockStatusApi 설정이 TRUE일 경우, SetRamBlockStatus(TRUE)를 호출 해야만 WriteBlock이 수행됨. VALID/UNCHANGED 일 경우는 실제 Write 동작을 수행하지 않고 OK만 반환함.
동작 영향	AUTOSAR 4.3.0 을 반영하여 NvMSetRamBlockStatusApi 설정이 TRUE 일 경우, SetRamBlockStatus 호출 여부와 상관없이 WriteBlock 이 수행되도록 수정함.
설정 영향	없음
ASW 조치 사항	6.3.2.6 NvM_SetRamBlockStatus 참조.

➤ Version 1.5.4.0 (2021-12-30)

- 신규 기능

■ N/A

- 개선 사항

■ UNECE Cyber Security 법규 대응을 위한 보안 코딩 개선

원인	UNECE Cyber Security 법규 대응을 위한 보안 코딩 개선
동작 영향	없음
설정 영향	없음
ASW 조치 사항	없음

➤ Version 1.5.3.0 (2021-02-01)

- 신규 기능

■ N/A

- 개선 사항

■ Redundant block 의 return value 수정

원인	Redundant block의 orginial, copy block의 읽기가 실패했을 경우 return value의 기준을 정함. * Non-AUTOSAR Specification
동작 영향	Redundant block 의 orginial, copy block 의 읽기가 실패했을 경우 original block(first block)의 job result 를 해당 block 의 return value 로 반환함. 두 블록 중 하나의 block 이 virgin 이라면 다른 block 의 job result 를 반환함
설정 영향	없음
ASW 조치 사항	없음

➤ Version 1.5.2.0 (2021-01-20)

- 신규 기능

■ N/A

- 개선 사항

■ MISRA Rule 적용

원인	MISRA Rule 적용
동작 영향	없음
설정 영향	없음
ASW 조치 사항	없음

➤ Version 1.5.1.0 (2020-12-31)

- 신규 기능

■ N/A

- 개선 사항

■ MISRA Rule 적용

원인	MISRA Rule 적용
동작 영향	없음
설정 영향	없음
ASW 조치 사항	없음

■ Dem Event 발생 조건 변경

원인	Redundant Block이 모두 Virgin(Invalidated) 상태 일 때 Dem Event NVM_E_LOSS_OF_REDUNDANCY 발생
동작 영향	Redundant Block 이 모두 Virgin(Invalidated) 상태 일 때 Dem Event NVM_E_LOSS_OF_REDUNDANCY 가 발생하지 않도록 수정. 한번도 쓰여지지 않은 상태이므로 불필요한 Dem Event 가 발생하지 않도록 변경함
설정 영향	없음
ASW 조치 사항	없음

➤ Version 1.5.0.0 (2019-10-17)

- 신규 기능

■ N/A

- 개선 사항

■ Parameter Definition File Category 수정

원인	소스 코드 오픈
동작 영향	없음
설정 영향	유저가 설정 가능한 설정에 한하여 권한 변경
ASW 조치 사항	Category 가 Changeable 로 변경된 설정 수정 가능

➤ Version 1.5.0 (2019-09-26)

- 신규 기능

■ NvM_CddGetStatus 추가

원인	Memory stack 의 상태 확인
동작 영향	없음
설정 영향	없음
ASW 조치 사항	BSW 메모리 모듈들의 상태를 확인할 필요가 있을 경우 사용되는 함수이다.

■ NvMUserJobFunction Callout 추가

원인	User job callout 추가
동작 영향	없음
설정 영향	NvMCommon/NvMUserJobFunction
ASW 조치 사항	유저가 NvMUserJobFunction 를 설정하면 OsTask_BSW_Mem_Process Task 에서 UserJob 함수가 구동된다.

- 개선 사항

■ N/A

➤ Version 1.4.2 (2019-08-30)

- 신규 기능

■ N/A

- 개선 사항

■ NvMMMainfunctionTriggerRef 의 Multiplicity 를 0..1 로 변경

원인	NvMMMainfunctionTriggerRef 의 Multiplicity 를 0..1 로 변경
동작 영향	없음
설정 영향	NvMMMainfunctionTriggerRef
ASW 조치 사항	Alarm 사용을 원치 않는 경우 NvMMMainfunctionTriggerRef 의 설정을 제거할 수 있다. 변경 요청 필요하며, 미설정시 Memory stack mainfunction 구동을 위한 SetEvent API 호출이 필요하다. Appendix 참조

■ 설정 변경없이 생성코드가 변경되는 이슈

원인	설정 변경이 없는데도 생성 코드 NvM_Cfg.c 의 include 문의 위치가 변경됨
동작 영향	없음
설정 영향	없음
ASW 조치 사항	없음

➤ Version 1.4.1 (2019-04-04)

- 신규 기능

■ N/A

- 개선 사항

■ Write Verification 기능 오류 수정

원인	WriteAll 단계에서 Write Verification 이 true 로 설정된 NvM block 은 Write 한 후 다시 Read 하여 두 값을 비교한다. 정상적으로 쓰여져 있는지 검증하는 기능이다. 만약 Read 가 실패했을 경우 WriteAll 이 중단된다. 이로 인해 Shutdown /Reset sequence 가 WriteAll 단계에서 멈출 수 있다.
동작 영향	없음
설정 영향	없음
ASW 조치 사항	없음

■ 매뉴얼 유저 가이드 수정(Appendix)

원인	- . Garbage Collection 주의 사항 추가 - . DFlash size 계산식 SAG 문서로 이동
동작 영향	없음
설정 영향	없음
ASW 조치 사항	Aurix : 주의 사항 참고하여 MCAL Fls 설정 검토

➤ Version 1.4.0 (2018-01-31)

- 신규 기능

■ N/A

- 개선 사항

■ 설계시 유의 사항 보강

원인	chorus MCU 의 Internal EEPROM Size 계산 공식 추가. Memory layout 변경시 조치사항. 사용자 구현이 필요한 Mem_Integration_User Code 설명 추가
동작 영향	없음
설정 영향	없음
ASW 조치 사항	Appendix 설계시 유의 사항 참조

■ [Aurix TCxx] SPI DMA 제약사항 적용

원인	SPI 에서 DMA 사용시 아래와 같은 제약 사항이 존재함. “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)” 이를 위해 NVM_RAM_VAR_CLEARED_ALIGNED_ADDR_UNSPECIFIED 메모리 섹션을 추가함
동작 영향	없음
설정 영향	없음
ASW 조치 사항	없음

■ Task 최적화

원인	기존 GPT 으로 trigger 되던 구조를 Alarm 으로 변경하여 설정을 단순화함. 또한 단순한 Task 구조를 통해 수행 속도를 최적화함..
동작 영향	GPT 사용시 ,5ms 에서 NvM_Mainfunction 을 할당하지 않음.
설정 영향	없음
ASW 조치 사항	없음

■ Dem 용 NvBlock 의 port 생성 방지

원인	Dem 을 위한 Block 추가시 NvM sw-component 에 port 가 생성되어 RTE 에 추가 설정 필요. 불필요한 추가 설정으므로, 이를 방지 하기 위해 Dem 용 Block 에 대해 Port 등이 생성되지 않도록 수정함.
동작 영향	없음
설정 영향	없음
ASW 조치 사항	없음

■ Rom Block 오류 수정

원인	Preconditions: 1. NvMRomBlockDataAddress 이 설정됨 2. NvMNvBlockLength > 32 Preconditions 를 만족하는 Block 읽기가 실패할 경우 Rom Data 의 32 bytes 만 default 값으로 사용되고 나머지 byte 들은 복사 되지 않는 현상이 발생하지 않도록 수정.
동작 영향	없음
설정 영향	없음
ASW 조치 사항	없음

➤ Version 1.3.5 (2017-06-30)

- 신규 기능

■ N/A

- 개선 사항

■ Compile warning 제거

원인	Compile warning 제거
동작 영향	없음
설정 영향	없음
ASW 조치 사항	없음

■ Read/Write 용 임시 버퍼 크기조정

원인	Ea module 과 버퍼 공용화를 위하여 버퍼 크기를 8 bytes 증가 시킴. 기능적인 수정없음.
동작 영향	없음
설정 영향	없음

ASW 조치 사항	없음
-----------	----

➤ 1.3.4 (2016-12-06)

- 신규 기능

■ N/A

- 개선 사항

■ MultiBlockCallback 을 CDD 에서 사용할 수 있도록 기능개선

원인	CDD 에서 MultiBlockCallback 사용 요청
동작 영향	없음
설정 영향	없음
ASW 조치 사항	없음

➤ Version 1.3.3 (2016-11-02)

- 신규 기능

■ N/A

- 개선 사항

■ UserManual (RH850 Internal EEPROM Size 계산방법, Callback Task 설정방법) 추가

원인 : 설정시 주의점 추가

동작영향 : 없음

설정영향 : 없음.

ASW 조치 필요 사항 : 업데이트 내용 (Chap 8.6) 참고

■ ImmediateBlock 사용시 문제점 개선

원인 : Redundant block Write 도중 ImmediateBlock Write 요청시 Cancel 을 하며.

이때 오동작 가능성 발생.

동작영향 : 없음.

설정영향 : 없음.

ASW 조치 필요 사항 : 없음.

➤ Version 1.3.2 (2016-10-17)

- 신규 기능

■ N/A

- 개선 사항

■ Library 의존성 개선

원인 : User 설정 편리성 개선

동작영향 : 없음

설정영향 : NvMBlock_ConfigID(BlockID1 번)의 ReadAll/WriteAll Option 을 True 로 설정

ASW 조치 필요 사항 : 상동

➤ Version 1.3.1 (2016-09-22)

- 신규 기능

■ N/A

- 개선 사항

■ Queue Clear 시간 개선

원인 : Queue Size 를 255 로 설정시, Queue clear 할 때 시간 지연

동작영향 : 없음

설정영향 : 없음

ASW 조치 필요 사항 : 없음

4.3.1.2 Limitations

- 다른 Memory Stack 모듈(Fee, Fls, Ea, Eep)의 MainFunction 과 Preemption 이 일어나지 않도록 같은 Task 에 Mapping 해야 한다.

4.3.1.3 Deviation

- The VARIANT-LINK-TIME 은 지원하지 않는다. (Requirement NVM727)
- Explicit Sync Mechanism 기능은 지원되지 않는다. (SWS - Chap 7.2.2.17)
- NvM 모듈에서는 Sync Mechanism 기능을 설정할 수 있으며, 4.0 에서 추가된 NvSWC 와 통신할때 사용할때 이 기능을 사용한다. (Ram 에 Mirror Buffer 를 두고 NvData Interface 를 이용하여 통신하며 이때 NvM 에서는 이 Buffer 의 Data 를 읽고 쓴다.)
- AUTOSAR Debugging 기능은 지원되지 않는다. (SWS - Chap 7.7)
- Swcd 생성시 PortName 을 AUTOSAR_SWS_NVRAMManager 4.2.2 Specification 에 따라 생성함.
- AUTOSAR_SWS_NVRAMManager 4.3.0 Specification 에 따라 RamBlockStatusrk 의 상태가 VALID/UNCHANGED 일 때도 NvM_WriteBlock 이 정상적으로 수행되도록 수정하였다.

5. Configuration Guide

5.1 NvM 모듈

5.1.1 Common Container

다음 설정을 참고한다.

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) SRS 에서 DataSet Type Block Usage 가 No 인경우 : 1

SRS 에서 DataSet Type Block Usage 가 Yes 인경우 : DataSet Type Block 들의 NvBlockNum 값중 최대값에 따라 달라지며, NvBlockNum 개수는 Application 에서 설계를 해야한다.

NvMDatasetSelectionBits 의 값에 따라 DataSet Type 으로 설정된 Block 에서 설정할 수 있는 최대 NvBlock(Fee/Ea Block)의 개수가 달라진다. (모든 DataSet Type Block 은 NvBlock 을 최대 2ⁿ개 까지 설정할 수 있다. n : NvMDatasetSelectionBits)

- Ex) 이 값이 2 라면 모든 DataSet Block 이 가질수 있는 최대 NvBlock 의 개수는 4
- Ex) 이 값이 3 이 라면 DataSet Block 이 가질수 있는 최대 NvBlock 의 개수는 8

이 설정값에 따라 모든 Fee/Ea Block 의 BlockNumber 값이 달라진다. (Chap 5.2.2)

기본 공식 : $2^n * NvMNvBlockBaseNumber + Index$

(n : NvMDatasetSelectionBits, Index : 0 부터 순차적임, NvBlock(Fee/Ea Block) 개수만큼)

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) SRS 의 Immediate Block Usage 가 Yes 인 경우 (Application 에서 Immediate Block 을 사용한다면), NvMJobPrioritization 항목을 True 로 설정

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
- NvM Manger 에서 요청된 Request(Read/Write job 등)를 저장하는 개수
 - 설정된 값 이상으로 Request(Read/Write)가 queue 에 쌓이면 QueueOverFlow Dem Error 발생. 이때는 값을 늘려줘야 함.
 - Immeidate Block, Standard Block 각각 설정된 Block 의 개수만큼 Application 에서 설정
- 4) 각 Block 에서 RamBlockDataAddress/RomBlockDataAddress 를 설정하면, NvM 생성파일에서 그 주소를 참조하며, 그 주소(버퍼)가 extern 으로 선언되어 있지 않아서 컴파일 에러가 발생한다. 따라서 RamBlockDataAddress/RomBlockDataAddress 설정시, 그 주소(버퍼)가 extern 으로 선언된 Header File 을 NvM 에서 Include 를 시켜야 하며, 그 Header File Name 을 NvMUserIncludeFiles 에 추가시킨다.
- 5) SRS Information 의 DynamicConfiguration Usage 항목에 따라 설정.

주의 : 만약 SRS 에 해당 항목이 없다면 해당 MCU 는 DynamicConfiguration 를 지원하지 않는다.

DynamicConfiguration 에 대한 설명은 다음과 같다.

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. (Extended/normal runtime 을 관련하여 8.5 참조)

- 6) MultiBlock Request (ex, ReadAll) 호출 이후에 Job 이 끝났을시 (ex. ReadAll 완료), 설정된 Callback 이 호출된다.
- 7) 설정한 NvMUserJobFunction 함수를 OsTask_BSW_Mem_Process Task 에서 호출한다.
- 8) ReadAll / WriteAll 동작 시 처리되는 Block 순서를 사용자가 설정하는 기능으로 Disable 시 AUTOSAR Specification 기준으로 동작한다.
- 9) 사용자가 NvMUserWdgToggleFunction 를 설정하면 초기화가 완료되기 전까지 Mem_EalnitPerform, Mem_FeelnitPerform 에서 User 함수가 반복적으로 수행된다.
해당 기능은 External Watchdog 을 사용하고 Timer 설정 시간이 짧아, 메모리 초기화 완료 전 Reset 이 발생하는 경우에만 사용해야 한다. 일반적으로 플랫폼에서 제공하는 Watchdog 기능 사용을 권장한다.

5.1.2 NvMBlockDescriptor 설정

다음 설정을 참고한다.

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
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

NvMWriteAllOrder¹⁷⁾

User Defined

C

- 1) CRC 사용시
 - NvMBlockCrcType : App 에서 원하는대로 설정.
 - NvMBlockUseCrc, NvMCalcRamBlockCrc : True 로 설정. (사용하지 않을시 False 로 설정)
 - Fee / Ea Block 의 size 는 NvMNvBlockLength + CRC Byte 로 설정해 주어야 한다.
- 2) NvMBlockJobPriority
 - Immediate Block 일 때 : 0 으로 설정
 - Standard Block 일 때 : (Block 간 우선순위를 고려하여) 0 이외의 값으로 설정
- 3) NvMBlockManagementType
 - NVM_BLOCK_NATIVE : one copy
 - NVM_BLOCK_REDUNDANT : two copy
 - NVM_BLOCK_DATASET : an array of equally sized data blocks.
- 4) NvMNvBlockBaseNumber
 - NvMNvramBlockIdentifier 와 동일하게 설정
- 5) NvMNvBlockLength
 - App 의 설계에 따라 설정 (하위 모듈 Block – 연결되는 Fee/Ea Block –Length 와 같아야함)
- 6) NvMNvBlockNum
 - NVM_BLOCK_NATIVE : 1 로 설정
 - NVM_BLOCK_REDUNDANT : 2 로 설정
 - NVM_BLOCK_DATASET : App 의 설계에 따라 설정.
- 7) NvMNvramBlockIdentifier
 - 다른 Block ID 와 Sequential 하도록 설정해야함. (From Autosar 사양)
- 8) NvMNvramDeviceld
 - 연결되는 하위 모듈 (Fls/Eep)의 Device ID
 - In / Ex EEPROM 을 하나만 사용시는 0 으로 설정
 - In / Ex EEPROM 을 동시에 쓸 경우에는 Internal EEPROM 에 저장되는 Block 은 0, External EEPROM 에 저장되는 Block 은 1 로 설정
- 9) NvMRamBlockDataAddress
 - ReadAll / WriteAll 사용시는 필히 설정
 - ReadAll 시 설정된 RamBlock 에 NvBlock 값을 읽어오고, WriteAll 시 설정된 RamBlock 값을 NvBlock 에 쓴다.
 - 설정된 RamBlock 이 NvM_Cfg.c 에 Include 될 수 있도록 extern 으로 선언된 파일을 NvM Common Container 의 UserIncludeFile 에 추가해야 함.
 - NvM 모듈에서는 설정된 RamBlock 의 시작주소와 설정된 length 만 가지고 업데이트를 하기 때문에 RamBlock 이 NvBlock 보다 length 가 작을시에는 다른 변수의 ram 영역을 침범할 수 있다. 따라서 RamBlock 과 NvBlock 의 length 는 꼭 같게 맞춰준다.
- 10) NvMSelectBlockForReadAll / NvMSelectBlockForWriteAll
 - ReadAll : StartUp 시에 EEPROM 에 저장되어 있는 값을 Ram 으로 읽어옴.
 - WriteAll : ShutDown 시에 Ram 값을 EEPROM 에 써줌.
WriteAll 사용시에는 WriteAll Time 을 측정하여 WdgDelnit 이후 WdgReset 까지 걸리는 시간에 반영해야 한다.
- 11) NvMSingleBlockCallback
 - 사용시에는 “Rte_Call_NvM_PNJF_{Block}_JobFinished”로 설정 (Chap 8.7 참조)
Block = {ecuc(NvM/NvMBlockDescriptor.SHORT-NAME)}
- 12) NvMTargetBlockReference
 - Internal EEPROM 에 쓰는 Block 이라면 Fee 선택
 - External EEPROM 에 쓰는 Block 이라면 Ea 선택
- 13) NvMNameOfFeeBlock/ NvMNameOfEaBlock
 - NvM 설정에 맞게 Fee / Ea Block 을 생성한 후 그 첫번째 Block 을 연결시켜준다.

14) NvMInitBlockCallback

- Read Fail 시에 InitBlockCallback 이 설정되어 있으면, Callback 을 호출한다. Application 에서는 그 Callback 내에서 설계의도에 맞게 Ram Block 값을 처리해 준다. RamBlock 은 Application 에서 처리했다고 생각하여, NvM 은 InitBlockCallback 이 호출된 후에는 Block 의 상태를 NVM_REQ_OK 로 설정한다.
- 사용시에는 “Rte_Call_NvM_PNIB_{Block}_InitBlock”로 설정 (Chap 8.7 참조)
Block = {ecuc(NvM/NvMBlockDescriptor.SHORT-NAME)}

15) NvMResistantToChangedSw

- NvMDynamicConfiguration 설정이 TRUE 일때, 의미가 있음.
- 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

- RomBlock 을 설정하지 않을 경우 값을 항상 0 으로 설정해야 한다.

17) NvMReadAllOrder / NvMWriteAllOrder

- 순서 설정이 필요없는 Block 의 경우 Order 를 설정하지 않아도 된다.
- 순서 설정이 필요한 Block 의 경우 Order 는 <1>부터 시작해야 하고 다른 Order 와 Sequential 하도록 설정해야 한다.
- Block ID 가 <0>이거나 <1>인 경우 Order 는 설정하지 않아야 한다.

5.1.3 NvMDemEventParameterRefs 설정

다음 설정을 참고한다.

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 모듈 (Internal EEPROM)

FEE 는 Mcal 모듈이며, 설정관련 사항은 MCU 제조사의 Fee User Manual / Fee Integration Manual 을 우선적으로 참고한다.

5.2.1 FeeGeneral Container

다음 설정을 참고한다.

Parameter Name	Value	Category
FeeDevErrorDetect	True	C
FeeIndex	0	F
FeeNvmJobEndNotification	NvM_JobEndNotification	F

Parameter Name	Value	Category
FeeNvmJobErrorNotification	NvM_JobErrorNotification	F
FeePollingMode	False	N
FeeSetModeSupported	True	F
FeeVersionInfoApi	True	C
¹⁾ FeeVirtualPageSize	User Defined	C

1) FeeVirtualPageSize:

Mcu dependant 한 항목 (MCU 제조사의 Fee User/Integration Manual 참고)

5.2.2 BlockConfiguration 설정

NvMBlock 의 NvBlock 개수 (NvMNvBlockNum 설정값) 만큼 Fee Block 을 추가해 주어야 한다.
 예를 들어 NvMNvBlockNum 값이 2(Redundant Block)라면 Fee Block 을 2 개를 만들어 주어야 한다.
 같은 NvMBlock 에 연결되는 Fee Block 이라면 FeeBlockNumber 를 뺀 나머지 설정값은 같아야 하며,
 두번째 Block 의 FeeBlockNumber 은 첫번째 Block 의 FeeBlockNumber 에 +1 (Index)를 해주면 된다.

다음 설정을 참고한다.

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

기본 공식 : $2^n * \text{NvMNvBlockBaseNumber} + \text{Index}$ (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

연결되는 NvMNvBlockLength 와 같게 설정

5.2.3 FeeIcxSpecificConfig (Only Aurix)

Parameter Name	Value	Category
FeeMaxBlockCount ¹⁾	User Defined	C

1) FeeMaxBlockCount

설정된 Block 의 개수를 적어준다.

※ 언급되지 않은 설정은 User 가 변경하면 안되는 설정임.

5.2.4 Fee Information

1) Aurix

- Internal EEPROM Virgin 상태에서 Fee Init 과정시(i.e. in factory) 방해(i.e reset)를 받아서는 안된다. (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.) - 자세한 내용은 Fee User 's Manual 참고

5.2.5 Redundant Block 분산 배치 (Only Chorus/Bolero)

Redundant NvM Block 에 연결되는 2 개의 Fee Block 은 ClusterGroup0 와 ClusterGroup1 에 분산배치하는 것이 한 곳에 배치하는 것보다 DFLASH 수명면에서 유리하다.

5.3 FLS 모듈 (Internal EEPROM)

FLS 는 Mcal 모듈이며, 설정관련 사항은 MCU 제조사의 FIs User Manual / Integration Manual 을 우선적으로 참고한다.

5.3.1 FIsGeneral Container

다음 설정을 참고한다.

Parameter Name	Value	Category
FIsAcLoadOnJobStar	False	C
FIsBaseAddress	0	C
FIsCancelApi	False	C
FIsCompareApi	False	C
FIsDevErrorDetect	User Defined	C
FIsDriverIndex	0	F
FIsGetJobResultApi	True	F
FIsGetStatusApi	True	F
FIsSetModeApi	True	F
FIsTotalSize	User Defined	C
FIsUseInterrupts	False	N
FIsVersionInfoApi	False	C

1) FIsTotalSize

Mcu dependant 한 항목 (Mcu 제조사에서 제공하는 FIs User/Integration Manual 참고)

5.3.2 FIsConfigSet

다음 설정을 참고한다.

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 한 항목 (MCU 제조사의 Fls User/Integration Manual 참고)

5.3.3 FlsDemEventParameterRefs

다음 설정을 참고한다.

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

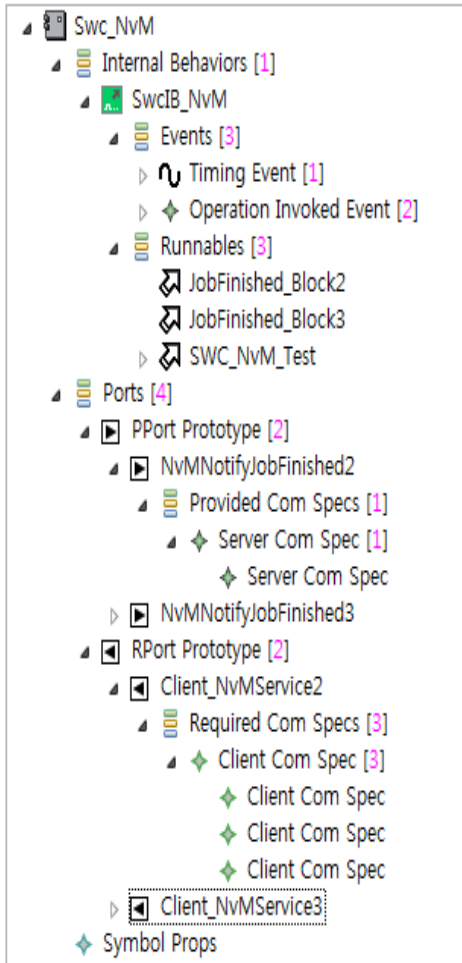
다음 설정을 참고한다.

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 한 항목 (MCU 제조사의 Fls User/Integration Manual 참고)

5.4 System Configuration

5.4.1 ApplicationSwComponentType 설정



☞ Pport, Rport 를 1 개씩 생성

PPort

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

Provided Interface 에 NvMNotifyJobFinished 맵핑
ServerComSpec 생성 및 Operation 에 JobFinished 맵핑
Queue 에 1 기입

RPort

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

Required Interface 항목에 NvMService 맵핑
ClientComSpec 생성
각각 Operation 항목에 필요한 API Mapping
(GetErrorStatus , ReadBlock, WriteBlock 등)

☞ SwcInternalBehavior

Runnable Entity 생성

Short name : JobFinished_Block<Block ID>

Symbol : JobFinished_Block<Block ID>

CanBeInvokedConcurrently = false

☞ Events / OperationInvokedEvent 생성

Short name : OperationInvokedEvent_JobFinished_<Block ID>

StartOnEvent 항목에 JobFinished_Block<Block ID> 맵핑

Operation / ROperation In Atomic Swc Instance Ref 생성

Context P Port : NvMNotifyJobFinished<Block ID>

Target Provided Operation : JobFinished

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

Short name : SWC_NvM_Test (NvM Api 를 호출할 함수)

Symbol : SWC_NvM_Test

Server call points/3 개의 SynchronousServerCallPoint 생성

short name : SynchronousServerCallPoint_GetErrorStatus_<Block ID>

Operation / ROperation In Atomic Swc Instance Ref 생성

Context R Port : Client_NvMService<Block ID>

Target Required Operation : GetErrorStatus

short name : SynchronousServerCallPoint_ReadBlock_<Block ID>

Operation / ROperation In Atomic Swc Instance Ref 생성

Context R Port : Client_NvMService<Block ID>

Target Required Operation : ReadBlock

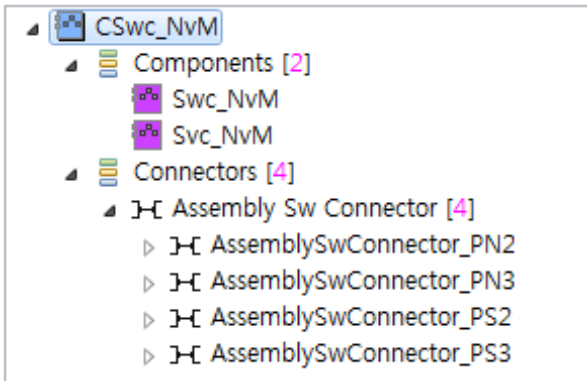
short name : SynchronousServerCallPoint_WriteBlock_<Block ID>

Operation / ROperation In Atomic Swc Instance Ref 생성

Context R Port : Client_NvMService<Block ID>

Target Required Operation : WriteBlock

5.4.2 CompositionSwComponentType 설정



AssemblySwConnector 생성

AssemblySwConnector 에 각각 Provider, Requester 하나씩 생성

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

※ ClientServer Interface 에 대한 설정 사항은 AUTOSAR BSW Service API Guide.doc 문서를 참조한다.

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 signalled via a callback function.		

※ 보충설명

1) NVM_REQ_OK

Request 가 성공했을시의 Block 의 상태.

또한 read 실패후, (InitBlockCallback 이 설정되어 있고), InitBlockCallback 을 호출하면, Ram Block 은 Application 에서 처리했다고 생각하여, Block 의 상태는 NVM_REQ_OK 로 설정됨.

2) NVM_REQ_NOT_OK

Request 가 실패하였을시에 Block 의 상태.

3) NVM_REQ_PENDING

Request 를 받은 상태로, Request 가 진행중인 Block 의 상태.

4) NVM_REQ_INTEGRITY_FAILED

Read/ReadAll 시에 fail 이 났으며, 읽기 동작은 이상없이 진행되었으나, 그 fail 원인이 읽은 Data 값의 CRC 값이 맞지 않았을 때 발생하는 Block 의 상태.

5) NVM_REQ_BLOCK_SKIPPED

ReadAll 이나 WriteAll 시에 ReadAll/WriteAll 로 설정된 Block 은 그에 맞는 동작을 진행하며, ReadAll/WriteAll 이 설정되지 않은 Block 은 ReadAll/WriteAll 을 Skip 했단 의미로 NVM_REQ_BLOCK_SKIPPED 가 설정됨.

6) NVM_REQ_NV_INVALIDATED

NV Block 이 invalidated 일때 (Fee 의 Job Result 결과가 MEMIF_BLOCK_INVALID 일때)로, Fee 에 설정한 Block 이 Invalid 일 때 발생하는 Block 의 상태.

예를들어, Ext.EEPROM 이나 Bolero 의 경우 한번도 write 하지 않은 초기 상태에서 Read 를 하는 경우에는 Block 상태가 NVM_REQ_NV_INVALIDATED 임 (단, NVM_REQ_NV_INVALIDATED 상태가 Write 를 한번도 안한 상태만 의미하지는 않음.)

7) NVM_REQ_REDUNDANCY_FAILED

Redundant Block 에서 둘 다 실패했을 시에 발생하는 Error 로, 스펙에서 Type 으로 정의 만 되어있고, 사용적인 측면에서는 나와있지 않아 현재 플랫폼 상에서는 따로 설정되지 않습니다. 하지만 스펙에 나와있는 만큼 추후에 업데이트될 가능성을 배제할 수 없기에 NVM_REQ_NOT_OK 와 동일하게 처리해 주시는 게 좋을 것 같습니다.

8) NVM_REQ_RESTORED_FROM_ROM

ROM Block 이 설정되어 있을시, Read 실패로 Rom Block 의 값이 Ram Block 에 저장이 되었을 때, 설정되는 Block 의 상태임.

실패시 결과값이 NVM_REQ_NV_INVALIDATED 나 NVM_REQ_INTEGRITY_FAILED 일지라도, ROM Block 이 설정되어 Rom Block 의 값이 저장이 되었다면, NVM_REQ_RESTORED_FROM_ROM 으로 Return 된다.

6.2 Macro Constants

None

6.3 Functions

6.3.1 Initialization

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

6.3.2.6 NvM_SetRamBlockStatus

Function Name	NvM_SetRamBlockStatus	
Syntax:	FUNC(Std_ReturnType, NVM_CODE) NvM_SetRamBlockStatus(NvM_BlockIdType BlockId, boolean BlockChanged)	
Service ID	0x05	
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.
	BlockChanged	TRUE: Validate the RAM block and mark as changed. (FALSE: Invalidate the RAM block and mark as unchanged.
Parameters (Inout)	None	
Parameters (Out)	None	
Return Value	Std_ReturnType	E_OK: The status of the RAM-Block was changed as requested. E_NOT_OK: An error occurred.
Description	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	

* 사용시 주의 사항

1. SetRamBlockStatus 호출시 BlockChanged의 값이 TRUE/FALSE와 상관없이 Xxx_WriteBlock은 항상 정상 처리된다.
2. VALID/CHANGED 상태의 Block만 WriteAll 단계에서 처리된다. 만약 WriteBlock 없이 WriteAll만 사용할 경우 SetRamBlockStatus 함수를 호출하여 Block의 상태를 NvM에게 알려야 한다.

상태	설명
INVALID/UNCHANGED	NvM_Init 후 Block을 Read하기 전 상태이다. 그리고 에러 복구 설정이 없는 Block의 Read가 실패했을 경우 INVALID/UNCHANGED 상태가 된다. 마지막으로 - SetRamBlockStatus(FALSE)를 호출했을 경우도 INVALID/UNCHANGED 상태로 천이한다.
VALID/UNCHANGED	해당 Block에 대해 ReadBlock 또는 ReadAll 기능이 정상적으로 수행했을 경우 VALID/UNCHANGED 상태로 천이된다.
VALID/CHANGED	Read 실패로 인해 Rom의 내용을 읽어 왔거나 SetRamBlockStatus(TRUE)를 호출했을 경우 VALID/CHANGED 상태로 천이한다.

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
Description	Request updates the job queue with the BlockId and Service Id to 'Erase' 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>_EraseNvBlock (void) <P> : R-Port Name	

6.3.3.5 NvM_CancelWriteAll

Function Name	NvM_CancelWriteAll
Syntax:	FUNC(void, NVM_CODE)NvM_CancelWriteAll(void)
Service ID	0x0A
Sync/Async	Asynchronous
Reentrancy	Non Reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None

Return Value	None
Description	Cancels a running NvM_WriteAll request.
Preconditions	NvM should be initialized.
Configuration Dependency	None

6.3.3.6 NvM_InvalidateNvBlock

Function Name	NvM_InvalidateNvBlock	
Syntax:	FUNC(Std_ReturnType, NVM_CODE) NvM_InvalidateNvBlock(NvM_BlockIdType BlockId)	
Service ID	0x0B	
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	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.	
Configuration Dependency	None	

6.3.4 Asynchronous Multi Block Requests

6.3.4.1 NvM_ReadAll

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

6.3.4.2 NvM_WriteAll

Function Name	NvM_WriteAll
Syntax:	FUNC(void, NVM_CODE) NvM_WriteAll(void)

Service ID	0x0D
Sync/Async	Asynchronous
Reentrancy	Non Reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	Initiates a multi block write request.
Preconditions	NvM should be initialized.
Configuration Dependency	None

6.3.5 Callback Notifications

6.3.5.1 NvM_JobEndNotification

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

6.3.5.2 NvM_JobErrorNotification

Function Name	NvM_JobErrorNotification
Syntax:	void NvM_JobErrorNotification(void)

Service ID	0x12
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	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	BSW 메모리 모듈들의 상태를 확인할 필요가 있을 경우 사용되는 함수이다.
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: NVM_OPSTATUS_IDLE: user job is idle NVM_OPSTATUS_BUSY: user job is busy
Description	<p>유저가 NvMUserJobFunction를 설정하면 OsTask_BSW_Mem_Process Task에서 UserJob 함수가 구동된다.</p> <p>UserJobFunction에 처리해야 할 일이 없을 경우 NVM_OPSTATUS_IDLE를 리턴해야 한다. 또한 UserJob기능을 수행해야 할 경우 NVM_OPSTATUS_BUSY를 리턴한다. 이때 Memory stack의 mainfunction들이 구동되지 않는다. 따라서 빠른 시간 내에 UserJob 처리를 완료하고 NVM_OPSTATUS_IDLE을 리턴해야 한다.</p> <p>Note: 먼저 NvM_CddGetStatus의 리턴값이 NVM_OPSTATUS_IDLE일 경우 UserJob을 시작해야 한다.</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>사용자가 NvMUserWdgToggleFunction를 설정하면 초기화가 완료되기 전까지 Mem_EalnitPerform, Mem_FeelnitPerform에서 User 함수가 반복적으로 수행된다.</p> <p>해당 기능은 External Watchdog을 사용하고 Timer 설정 시간이 짧아, 메모리 초기화 완료 전 Reset이 발생하는 경우에만 사용해야</p>

	한다. 일반적으로 플랫폼에서 제공하는 Watchdog 기능을 사용을 권장한다.
Preconditions	None
Configuration Dependency	This function is available only if configuration parameter NvMUserWdgToggleFunction is configured.

6.3.8 참고사항

6.3.8.1 In Communication with application SW-C

RTE 기반 생성된 함수의 프로토타입에 대한 사항은 AUTOSAR BSW Service API Guide.doc 문서 참조.

7. Generator

7.1 Generator Option

7.1.1 NvM

Option	Description
S	ECU Mode 에 대한 Mode Switch Interface 와 서비스 별 Client Server Interface 및 P-Port 를 Swcd_Bsw_EcuM.xml 에 생성한다.
P	Autosar NvM SWS 4.2.2 spec 에 따라 Port Name 을 생성한다.

7.2 Generator Error Message

7.2.1 NvM

7.2.1.1 Error Messages

1) ERR020001

Unexpected Error Found. Please contact AUTOEVER AUTOSAR 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 AUTOEVER AUTOSAR 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 AUTOEVER AUTOSAR Support System.

This error occurs, if the structure fields that are to be generated in the C Source file are empty. Contact AUTOEVER AUTOSAR 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
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
	NvMNvramDeviceld
	NvMResistantToChangedSw
	NvMRomBlockNum
	NvMStaticBlockIdCheck
	NvMWriteBlockOnce
	NvMWriteVerification
	NvMWriteVerificationDataSize
	NvMDefaultROMCRCEnabled
NvMCommon	NvMApiConfigClass
	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
NvMRamBlockDataAdress			RamBlockDataAdress_1
NvMReadRamBlockFromNvCallback			ReadRamBlock_1

Parameter Name	Container Name	Pattern	Example
NvMRomBlockDataAddress			RomBlockDataAddress_1
NvMSingleBlockCallback			SingleBlockCallback_1
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	

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	RamBlockDataAddress_2
NvMRomBlockDataAddress	RamBlockDataAddress_2
NvmlInitBlockCallback	RamBlockDataAddress_2
NvmSingleBlockCallback	RamBlockDataAddress_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
FeeBlockNumber	FeeBlockConfiguration
EaBlockNumber	EaBlockConfiguration

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
NvMSelectBlockForReadAll
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	1. EEPROM 이 물리적으로 파손이 되거나 SPI 통신이 되지 않아 잘못된 Data 를 읽어오는 경우에 CRC 가 맞지 않아 발생한다. 2. Write 도중 PowerOff (ie. Reset) 되어 일부 Data 만 Write 가 되었을시 발생한다. 또한 CRC 가 맞지 않았을 경우도 발생한다.
문제 발생 원인	H/W, SWP
Platform default Action	NO RESET
기능적 영향	Read 시에 발생할 수 있는 Error 로 Error Check 가 없다면 User 가 잘못된 Data 를 얻을 수 있다.
타 모듈 연관성	없음
MCU	공통
문제 유형	설정, 코드

Application 적용 가능 대책	<p>Read 시에 발생할 수 있는 Error 로 발생했을 경우에 해당 Block 의 상태를 NVM_REQ_INTEGRITY_FAILED 로 만들어준 다음 Application 에 Callback 을 통하여 알려주거나, App 에서 Block 의 상태를 직접 읽어서 알 수 있다.</p> <p>코드: (DEM Report 처리하기 보다는) 해당 Block 의 Request(Read) 가 실패했을 경우, 대응하는 로직(i.e default 값 사용 or write 등)으로 처리가능하다.</p> <p>설정: Redundant Block (Two Copies) 설정 시, 해당 Error 발생 가능성이 현저히 적다. (EEPROM 이 깨졌을때 발생할 수 있음)</p>
----------------------	--

8.1.2 NVM_E_LOSS_OF_REDUNDANCY

ErrorId Symbol	NVM_E_LOSS_OF_REDUNDANCY
Description	EEPROM 이 물리적으로 파손이 되거나 SPI 통신이 되지 않아 Redundant Block 의 Data 읽기가 모두 실패했을 경우에 발생한다.
문제 발생 원인	H/W
Platform default Action	NO RESET
기능적 영향	Read 시에 발생할 수 있는 Error 로 Error Check 가 없다면 User 가 잘못된 Data 를 얻을 수 있다.
타 모듈 연관성	없음
MCU	공통
문제 유형	코드
Application 적용 가능 대책	<p>Read 시에 발생할 수 있는 Error 로 발생했을 경우에 해당 Block 의 상태를 NVM_REQ_REDUNDANCY_FAILED 로 만들어준 다음 Application 에 Callback 을 통하여 알려주거나, App 에서 Block 의 상태를 직접 읽어서 알 수 있다.</p> <p>코드: (DEM Report 처리하기 보다는) 해당 Block 의 Read 실패했을 경우, 대응하는 로직(i.e default 값 사용 or write 등)으로 처리가능하다.</p> <p>※ Redundant 로 설정한 Block 에서만 NVM_E_LOSS_OF_REDUNDANCY 가 발생함. (EEPROM 이 깨졌을 때 발생할 수 있음)</p>

8.1.3 NVM_E_QUEUE_OVERFLOW

ErrorId Symbol	NVM_E_QUEUE_OVERFLOW
Description	Pending Job 이 Queue Size 보다 많아서 더 이상 저장할 수 없을때 발생한다.
문제 발생 원인	ASW
Platform default Action	NO RESET
기능적 영향	Read/Write 시에 발생할 수 있는 Error 로 Error Check 가 없다면 Read/Write 가 정상동작 하지 않거나, User 가 잘못된 Data 를 얻을 수 있다.
타 모듈 연관성	없음
MCU	공통
문제 유형	설정, 코드
Application 적용 가능 대책	(DEM Report 처리하기 보다는) API (Write/Read) Return 값으로 E_NOT_OK 가 return 된다.

설정: Queue Size 를 NvM Block 개수 만큼 설정하면 발생하지 않는다.

코드: E_NOT_OK 가 반환되면 다음 Task 에서 API 를 재호출해야 한다.

8.1.4 NVM_E_REQ_FAILED

ErrorId Symbol	NVM_E_REQ_FAILED
Description	EEPROM 이 물리적으로 파손이 되거나 SPI 통신이 되지 않아 읽기/쓰기가 실패할 경우 발생한다.
문제 발생 원인	H/W
Platform default Action	NO RESET
기능적 영향	Read/Write 시에 발생할 수 있는 Error 로 Error Check 가 없다면 Read/Write 가 정상 동작하지 않거나, User 가 잘못된 Data 를 얻을 수 있다.
타 모듈 연관성	없음
MCU	공통
문제 유형	코드
Application 적용 가능 대책	Read/Write 발생할 수 있는 Error 로 발생했을 경우에 해당 Block 의 상태를 NVM_REQ_NOT_OK 로 만들어준 다음 Application 에 Callback 통하여 알려주거나, App 에서 Block 의 상태를 직접 읽어서 알 수 있다. 코드: (DEM Report 처리하기 보다는) 해당 Block 의 Read/Write 가 실패했을 경우, 대응하는 로직(i.e default 값 사용 or write 등)으로 처리 가능하다.

8.1.5 NVM_E_VERIFY_FAILED

ErrorId Symbol	NVM_E_VERIFY_FAILED
Description	Verify 기능 설정시, Write 이후에 다시 읽어서 값이 일치하면 성공으로 판단한다. 이때 실패하면 발생하는 Error 로 EEPROM 이 물리적으로 파손이 되거나 SPI 통신이 되지않아 읽거나 쓰기가 실패했을 경우 발생한다.
문제 발생 원인	H/W
Platform default Action	NO RESET
기능적 영향	Write 시에 발생할 수 있는 Error 로 Error Check 가 없다면 write 가 정상동작을 하지 않아서, User 가 잘못된 Data 를 얻을 수 있다.
타 모듈 연관성	없음
MCU	공통
문제 유형	설정, 코드
Application 적용 가능 대책	Write 시에 발생할 수 있는 Error 로 발생했을 경우에 해당 Block 의 상태를 NVM_REQ_NOT_OK 로 만들어준 다음 Application 에 Callback 을 통하여 알려주거나, App 에서 Block 의 상태를 직접 읽어서 알 수 있다. 설정: NvMMaxNumOfWriteRetries 을 통해 재시도 횟수를 지정한다.

코드 : 따라서, (DEM Report 처리하기 보다는) 해당 Block 의 Write 가 실패했을 경우, 대응하는 로직(i.e retry 등)으로 처리가능하다.

8.1.6 NVM_E_WRITE_PROTECTED

ErrorId Symbol	NVM_E_WRITE_PROTECTED
Description	Application 에서 WriteBlock Protect 설정을 통해, 쓰기금지된 Block 에 대하여 해제 명령 없이 Write 요청시 발생한다.
문제 발생 원인	ASW
Platform default Action	NO RESET
기능적 영향	Write 요청이 Accept 되지 않는다.
타 모듈 연관성	없음
MCU	공통
문제 유형	코드
Application 적용 가능 대책	코드: NvM_SetBlockProtection API 를 통해, Write Protect 를 해제 후 Write 를 재 요청한다.

8.1.7 NVM_E_WRONG_BLOCK_ID

ErrorId Symbol	NVM_E_WRONG_BLOCK_ID
Description	<p>1. IDCheck 기능 설정시, Write 요청시 Block ID 를 데이터에 덧붙여 저장하고 읽을 때 저장된 Block ID 와 설정한 Block Id 를 비교한다. EEPROM 이 물리적으로 파손이 되거나 SPI 통신이 되지 않아 잘못된 Data 를 읽어와서, Block ID 가 맞지 않아 발생한다.</p> <p>2. Write 도중 PowerOff (ie. Reset) 되어 일부 Data 만 Write 가 되었을 경우 Block ID 가 맞지 않아 발생한다. Redundant Block (Two Copies)으로 발생 빈도를 줄일 수 있다.</p>
문제 발생 원인	H/W, SWP
Platform default Action	NO RESET
기능적 영향	Read 시에 발생할 수 있는 Error 로 Error Check 가 없다면 User 가 잘못된 Data 를 얻을 수 있다.
타 모듈 연관성	없음
MCU	공통
문제 유형	설정, 코드
Application 적용 가능 대책	<p>Read 시에 발생할 수 있는 Error 로 발생했을 경우에 해당 Block 의 상태를 NVM_REQ_NOT_OK 로 만들어준 다음 Application 에 Callback 통하여 알려주거나, App 에서 Block 의 상태를 직접 읽어서 알 수 있다.</p> <p>코드: (DEM Report 처리하기 보다는) 해당 Block 의 Request(Read) 가 실패시 대응하는 로직(i.e default 값 사용 or write 등)으로 처리 가능하다.</p> <p>설정: Redundant Block (Two Copies) 설정 시, 해당 Error 발생 가능성이 현저히 적다. (EEPROM 이 깨졌을때 발생할 수 있음)</p>

9. Appendix

9.1 기능별 설정 Guide

9.1.1 Redundant Block 설정(2 copies)

- 1) NvM 모듈에서 Block 을 생성 후.
- 2) Block Management Type 을 NVM_BLOCK_REDUNDANT 로, Nv Block Num 을 2 로 설정.
- 3) Ea / Fee Block 을 2 개 생성.
- 4) Length 는 동일하게, block Number 는 공식대로 (n : NvMDatasetSelectionBits)
- 5) 첫번째 Ea/Fee Block Number : $2n * \text{NvMNvBlockBaseNumber}$
- 6) 두번째 Ea/Fee Block Number : $2n * \text{NvMNvBlockBaseNumber} + 1$
- 7) NvM Block 의 Reference 를 첫번째 Ea / Fee Block 에 Mapping

9.1.2 CRC Implement

- 1) CRC 로 검사해야할 Block 의 NvMBlockUseCrc, NvMCalcRamBlockCrc 를 True 로 설정한다. NvMBlockCrcType 는 CRC8, CRC16, CRC32 중 App 의 설계의도에 맞게 설정한다.
- 2) Underlayer Block (Fee/Ea Block) size 는 기본적으로 NvM Block 의 Length 와 같아야 하나, CRC 를 추가했다면, NvM Block Length 에 CRC size 를 추가해 주어야 한다.
- 3) CRC8 일 때 Underlayer Block size 는 +1, CRC16 일 때 Length 는 +2, CRC32 일 때 Length 는 +4 로 설정한다.
- 4) Immediate Block 은 CRC 를 지원하지 않는다. (From SWS_NVM721) 즉, NvM Block Descriptor 설정에서 BlockJobPriority 를 0 으로 설정하면 CRC 를 설정했다 하더라도, Generator 에서 다음과 같은 Info 를 알려주며, CRC 기능을 사용하지 않는다. (Common Container 의 JobPriorization 여부와 무관) 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 설정

- 1) NvM Common Container 에서 NvMJobPrioritization 항목을 True 로 설정
- 2) Immediate 로 설정할 Block 의 NvMBlockJobPriority 항목을 0 으로 설정
- 3) NvMSizeImmediateJobQueue 에 Overflow 가 나지 않도록 설정값 수정

※ 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 설정

- 1) ReadAll/WriteAll 로 설정할 Block 의 NvMSelectBlockForReadAll / NvMSelectBlockForWriteAll 항목을 True 로 설정

ReadAll : StartUp 시에 EEPROM 에 저장되어 있는 값을 Ram 으로 읽어온다.

WriteAll : ShutDown 시에 Ram 값을 EEPROM 에 쓴다.

9.1.5 ReadAll / WriteAll Order Supporting 설정

- 1) ReadAll /WriteAll 동작 시 Block 처리 순서를 사용자가 설정하려면 NvMReadAllOrderSupport / NvMWriteAllOrderSupport 항목을 True 로 설정

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

Navigator

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

Set Ram Block Status Api*: ☐ false

Size Immediate Job Queue*:

Size Standard Job Queue*:

Main Function Call Cycle*:

Version Info Api*: ☐ false

Read All Order Support*: ☒ true

Write All Order Support*: ☒ true

Mainfunction Trigger Ref: [OsAlarm_BSW_Me](#)

- 2) NvMReadAllOrderSupport / NvMWriteAllOrderSupport 가 True 인 경우, 순서를 설정하려는 Block 들의 NvMReadAllOrder / NvMWriteAllOrder 항목에 Order 값을 설정한다.
단, Block ID 가 “1”인 Block 은 Config Block 으로 사용자가 순서를 설정할 수 없으므로 Order 는 설정 하지 않는다.
순서 설정이 필요없는 Block 의 경우 Order 를 설정하지 않아도 된다.
순서 설정이 필요한 Block 의 경우 Order 는 “1”부터 시작해야 하고 다른 Order 와 Sequential 하도록 설정한다.

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*:

DefaultROMCRCEEnabled*: ☐ false

Read All Order*:

Write All Order*:

Target Block Reference 1 [1]

9.2 설계시 유의사항

9.2.1 NvM Block Identifier

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

1) NvM Block Id 0, 1 번 Block 은 NvM Module 자체에서 사용하는 Block 으로 App 이나 다른 Bsw 모듈에서 사용하면 안된다.

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 User 들은 배포된 프로젝트에서 추가 설정할 필요가 없다.

9.2.2 RamBlock Length

RamBlock 의 length 는 NvBlock 의 length 와 같게 설정해야 한다.

9.2.3 Immediate Block

Standard 속성 block 의 job(i.e Write/Read)을 처리중 Immediate 속성을 가진 Block 의 Write Request 를 요청하면 처리중인 Job 을 cancel 하고 Immediate block Write 를 수행후에 다시 cancel 된 Job 을 수행한다. 이때 처리중인 Standard Job 이 Write 일 경우에는 일부만 쓰여진 상태에서 cancel 을 할 경우가 생기며, Immediate Block Write 를 처리후에 다시 cancel 된 Write 작업을 수행한다. 이 사이에서 reset 이 된다면 처리중인 job 의 EEPROM Data 는 일부만 쓰여진 상태로 남을 수 있으며, 이는 CRC 설정을 통하여 Error 를 검지할 수 있다.

9.2.4 Request 시 return 값 및 Block 상태 확인 여부

Read/Write 등 Request 요청시에는 API 의 Return 값을 확인하여야 한다. return 값이 E_OK 일때는 NvM 의 Queue 에 Request 가 정상적으로 등록이 되었으며, E_NOT_OK 일 경우에는 Error 가 발생하여 Queue 에 등록되지 않는다. 이 Error 의 대부분은 Request 의 Block 이 Pending 상태이거나, Queue 가 Full 인 상태이다. 따라서 User 는 Request 전에 Block 의 상태를 확인하여 Pending 이 아님을 확인 후에 Request 를 요청해야 한다. Read/Write 등 Async Request 를 요청하였을 경우에는 Polling 이나 callback 을 통하여 Job 이 완료되었음을 확인해야 한다. 만약 Request 가 끝나지 않은 상태에서 Reset 이 일어날 경우에는 그 동작이 완료됨을 보장할 수 없다.

9.2.5 ReadAll Time

현대차 표준플랫폼에서는 StartUpTwo 단계에서 ReadAll 기능을 수행후 끝나면, StartUpThree 단계로 넘어가서 Can 통신이 이루어진다. 이때까지 걸리는 시간은 ReadAll Block 의 개수에 따라 달라지므로, 전원인가 후에 Can 통신까지의 시간 Spec 을 초과한다면, ReadAll 개수를 줄여야 한다.

또한 Internal EEPROM 의 경우 Virgin 상태에서 StartUpTwo 시간은 Virgin 이 아닌 상태서와 비교하여 오래 걸린다. 이는 FeeBlock 을 Init 하는데 걸리는 시간이며, 이는 Fee 의 Block 개수에 따라 달라진다. Virgin 상태에서 Fee Block 의 개수가 많으면 Wdg Reset 이 걸릴수 있으며, 이는 NvMMaxNumOfReadRetries 설정 변경이나, StartUpTwo 때까지의 Wdg Time 을 조정하여 해결할 수 있다.

9.2.6 WriteAll Time

현대차 표준플랫폼에서는 Shutdown Sequence 단계에서 WdgM Delnit 이후에 WriteAll 기능을 수행한다. WdgM 설정에는 DelnitTimeOut 을 설정할 수 있으며, 이 값은 WdgM Delnit 이후 TimeOut 이 발생하면 Wdg Reset 을 발생시킨다. 그래서 DelnitTimeOut 값 이내에 WriteAll 이 수행완료가 되어야 한다. 이 WriteAll 의 수행시간은 EEPROM 종류와 NvM Block 의 개수에 따라 달라지므로, User 는 WriteAll 시간을 확인하여, DelnitTimeOut 값에 반영을 해야한다.

9.2.7 NvM API Call-Context

Application 에서 RTE 를 통하여 NvM 의 API 를 호출하거나, CDD 에서 NvM 의 API 를 직접 호출할 수 있다. 이 때, ISR (Interrupt Service Routine)에서 호출해서는 안된다. 이는 조건에 따라 API 의 수행시간이 유동적일 수 있기 때문이다. 또한 Low Power Mode 에서 호출해서는 안된다. 이는 High Power Mode 에서만 동작하도록 설계되어 있기 때문이다.

9.2.8 Virgin Internal EEPROM 의 Fee Init

Internal EEPROM 의 경우 Virgin 상태에서 DFlash 초기화 시간이 Virgin 이 아닌 상태서와 비교하여 오래 걸린다. 이때, Internal EEPROM Virgin 상태에서 Fee Init 과정시(i.e. in factory) 방해(i.e reset)를 받아서는 안된다. 따라서 초기 공정에서 Fee Init 하는 시간을 보장을 해 주어야 한다. 그 시간은 DFlash 를 Virgin 상태로 만들고, Power On 부터 NvM_ReadAll 이 호출될 때 까지를 측정하여 알 수 있다.

9.2.9 Memory Layout 변경시 Erase All

비정상 동작을 방지하기 위해 Memory Layout 이 변경(NvM Block 의 추가/삭제/Length/ CRC 변경등) 되었을 경우 Internal/External EEPROM 을 erase 해야 한다.

9.2.10 Mem_Integration_User Implementation

Path : Static_Code\Integration_Code\Integration_Mem\Usercode

File : Mem_Integration_User.h, Mem_Integration_User.c

아래 기능에 한하여 사용자가 수정할 수 있다.

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:while-loop 에서 NvM/Fee/Fls/Ea/Eep Mainfunction 을 호출하여 WriteAll 수행시간을 줄일 수 있다.

STD_OFF:normal operation 과 같이 NvM/Fee/Fls/Ea/Eep Mainfunction 을 약 1ms 마다 호출한다.

9.2.10.2 User Callbacks

7) Mem_PostFeeInitCallback (Common)

Fee Init 전에 호출되는 Callback 이다.

8) Mem_Cypress_IllegalStateCallback(Cypress amethyst/ artemis)

Memory Layout 변경등의 원인으로 FEE 의 초기화가 실패할 경우 호출된다.

“Flash erase all” 이외에 다른 복구 방법은 없다.

9) Mem_Infineon_IllegalStateCallback (Infineon Aurix)

Fee 가 정상적인 동작을 할 수 없을 경우에 호출된다. 이 Callback 이 호출되면 Fee 는 동작을 중지한다.

Fee 매뉴얼을 참조하여 case 별로 대응해야 한다.

9.2.11 Garbage Collection

Internal EEPROM(Flash)의 경우 하나의 sector 가 가득찰 경우, 다음 block 을 쓰기 위해 유효한 block 들을 비어 있는 sector 로 옮긴 후 여유 공간에 요청 받은 쓰기를 수행한다. 또한 기존의 가득찬 sector 를 삭제하게 된다. 이와 같은 알고리즘을 일반적으로 Garbage Collection(GC)이라고 한다. (Vendor 마다 용어의 차이가 있다.

Swap 또는 Refresh 라는 용어를 쓰는 vendor 도 있다.

GC 가 수행될 경우 기존 쓰기 작업보다 많은 시간이 걸리게 된다. 유효한 Block 들을 옮기는 작업과 Sector Erase 작업이 수행되기 때문에 수백 ms 의 시간이 소요된다.

[Note]

수행 시간은 MCU 마다 다르다. 자세한 사항은 Fee/FIs 매뉴얼 및 MCU DataSheet 를 참고해야 한다.

GC 가 수행되는 시점은 보통 쓰기 동작 중에 공간이 없을 때 발생한다.

아래 MCU 는 예외적인 경우이다.

(1) Aurix

Aurix MCU 의 Fee 는 GC 의 수행 시점을 설정으로 선택할 수 있다.

설정 Gc Restart

FEE_GC_RESTART_INIT : Fee 초기화 및 Write 시에 GC가 수행될 수 있다.

FEE_GC_RESTART_WRITE : Write시에만 GC가 수행된다.

(2) Cypress

Cypress MCU의 Fee의 경우 초기화 시점에 GC가 발생할 가능성이 있다.

Block을 쓰는 도중에 Reset이 발생되면 Data가 손상될 수 있다. Fee는 손상된 Data를 복구(이전 데이터)하기 위해서 Recycling (Cypress에서는 Recycling이라는 용어를 사용한다.) 수행한다.

[Note]

※ Fee 초기화시 Recycling이 일어날 경우 초기화 시간이 지연될 수 있다.

※ 자세한 내용은 Cypress Fee UserGuide를 참고 해야 한다.

9.2.12 NvMMainfunctionTriggerRef

NvM 등에 Alarm 사용을 원치 않는 경우 NvMMainfunctionTriggerRef 의 Alarm 설정을 제거할 수 있다. 단, 변경 요청을 해야 한다. 미설정시 Memory stack mainfunction 구동을 위해, Rte start 이후 주기적으로 SetEvent API 호출이 필요하다.

예)

```
if (E_OK != SetEvent(OsTask_BSW_Mem_Process, OsEvent_BSW_Mem_Process))
{
```

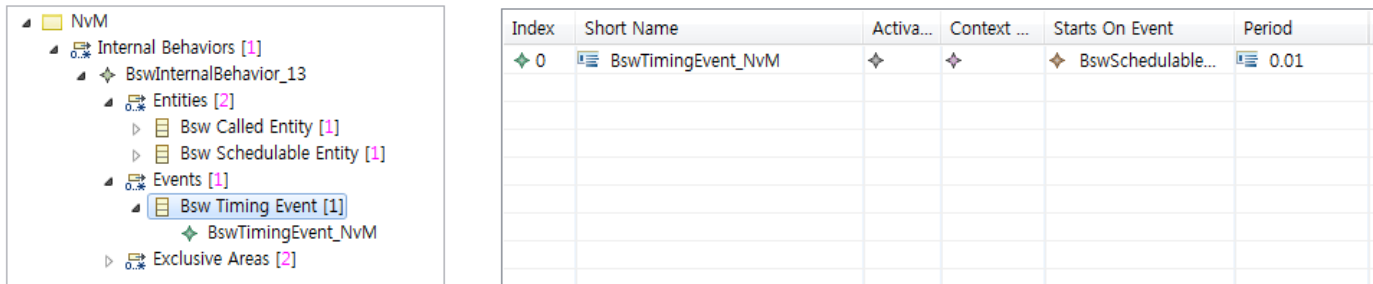
```
/* error */
}
```

9.3 Bswmd (Bsw Module Description)

9.3.1 MainFunction 주기 설정

NvM 모듈은 주기적으로 MainFunction 을 호출해야 하며 이를 TimingEvent 에 Mapping 하여 주기적으로 호출한다.

아래 그림과 같이 NvM 의 BswModuleDescription Container 에 Bsw Timing Event 에서 주기를 설정한다. Period 의 단위는 second 이며 SRS 에서 받은 MainFunction 주기가 5ms 일 경우 0.005 로 설정한다.



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

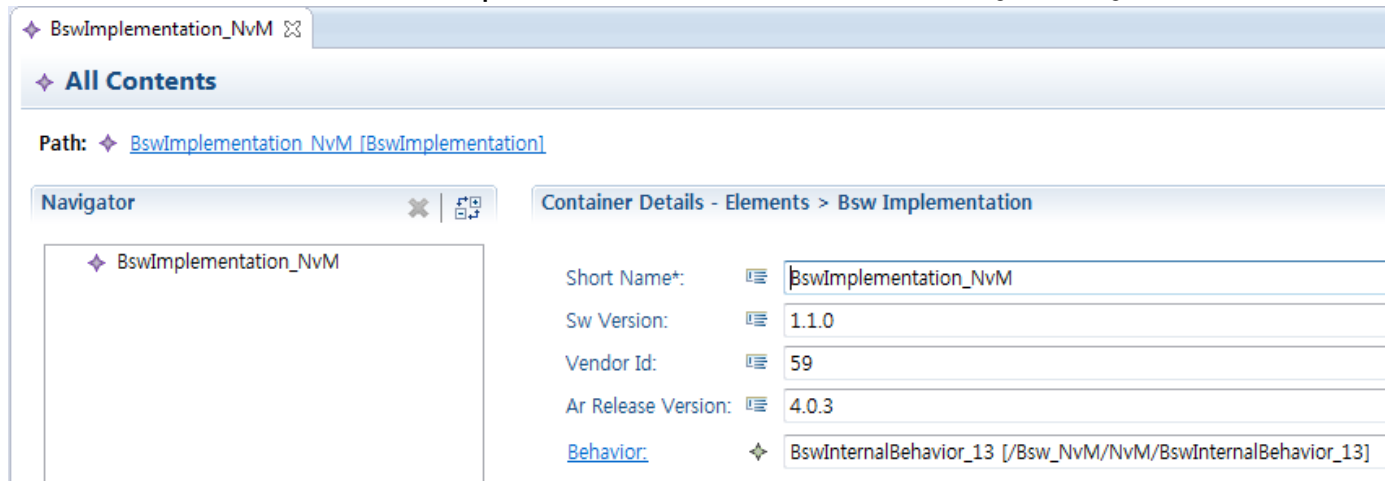
Memory Stack 에서 설정해야할 MainFunction 은 다음과 같다.

1. NvM_MainFunction (External, Internal EEPROM 모두)

9.3.2 Bsw 모듈 version 설정

각 모듈을 컴파일할 때, version 정보가 맞지 않으면 Compile 에서 Error 를 발생시킨다.

이때는 Bswmd 의 다음과 같이 BswImplementation Container 에서 version 정보를 수정해야 한다.



Container Details - Elements > Bsw Implementation	
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 모듈별 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()

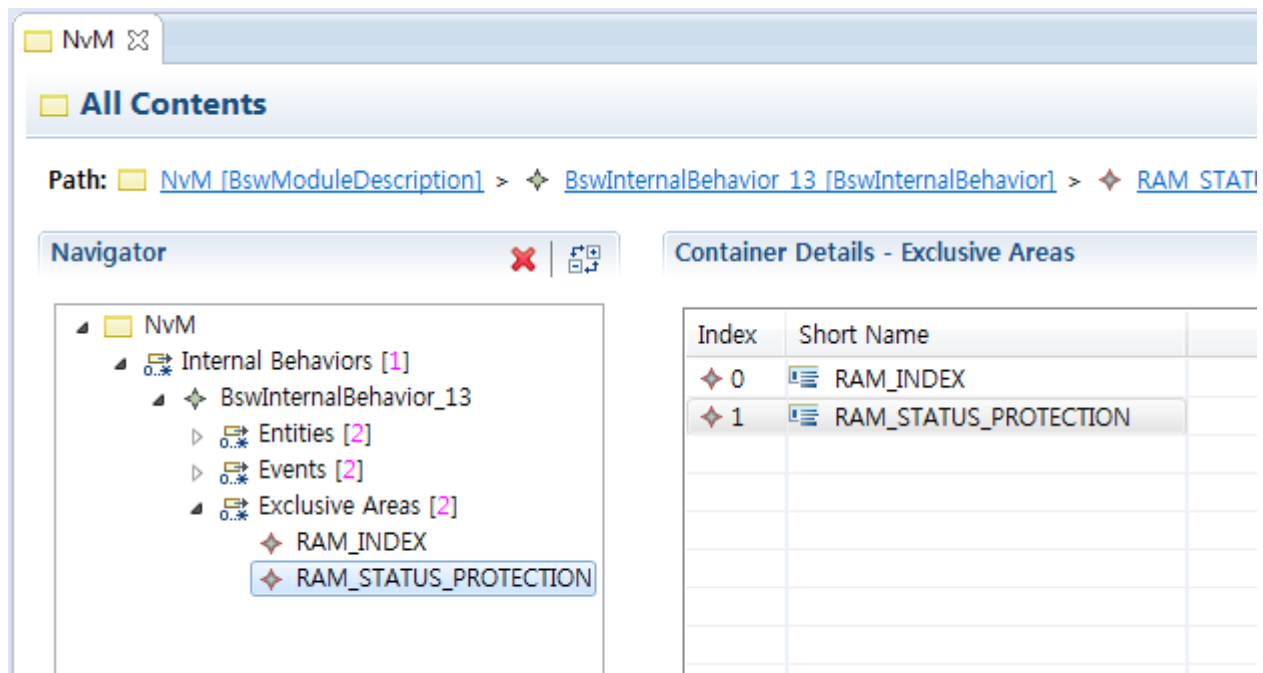
```

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 설정방법

모듈 BswModuleDescription Container 의 Exclusive Areas 에 다음과 같이 추가



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.

즉, common container 에 있는 `DynamicConfiguraition` 이 true 일 때, `ReadAll` 시에 `BlockID` 1 번을 먼저 읽어서 그 값과 설정된 `NvMCompiledConfigId` 이 일치하면 `ReadAll` 을 진행한다.

일치하지 않을 경우에는 Block 별로 설정할 수 있는 `NvMResistantToChangeSw` 의 설정값이 false 일 경우 읽기 성공여부에 상관없이 실패로 처리한다. (Default Data 가 있다면, `RamBlock` 에 그 Data 를 copy 한다.) `NvMResistantToChangeSw` 의 설정값이 true 인 Block 은 `NvMCompiledConfigId` 일치 여부와 상관없이 `ReadAll` 을 진행한다. 단, `BlockID` 1 번에 `NvMCompiledConfigId` 를 쓰는 것은 `Write Api` 나 `WriteAll` 을 통하여 Application 에서 해야한다.

9.6 Notification Interface with Application

9.6.1 SingleBlock Callback

NvM 모듈에서는 각 Block 마다 `SingleBlock Callback` 을 지원하며, 사용시에는 `Block Descriptor` 의 `SingleBlockCallback` 에 다음과 같이 설정한다. (Chap 5.1.2 참조)

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

※ Naming 규칙이 `Rte_Call_NvM_PNx_JobFinished` 로 되어 있는 프로젝트를 위와 같이 변경하려면 Chap 8.7 을 참고한다.

위와 같이 설정시에는 생성된 NvM Swcd 에서 다음과 같이 포트와 Interface 가 생성된다.

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

```
ClientServerInterface NvMNotifyJobFinished {
    JobFinished( IN uint8 ServiceId, IN NvM_RequestResultType JobResult);
};
```

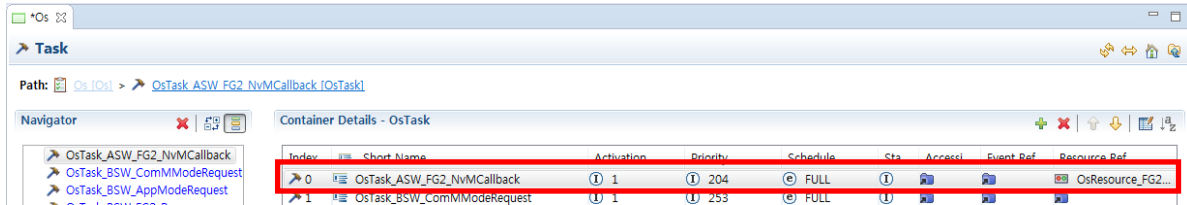
`ServiceId` 는 NvM Function 의 `ServiceID` 이며 (Chap 6.3 의 각 함수 설명 참조), `NvM_RequestResultType` 은 요청된 Request 의 결과이다 (Chap 6.1.1 참조).

Application 에서는 이를 `ApplicationSwComponent` 에 연결시켜 사용할 수 있다. (RTE 기반 생성된 함수의 프로토타입에 대한 사항은 AUTOSAR BSW Service API Guide.doc 문서 참조)

Application Sw Component 에 설정하는 Callback 관련 runnable 은 NvM SWS[NVM736]에 의거 `canBeInvokedConcurrently` 를 **FALSE** 로 설정해야 한다.

이에 동시에 발생하는 상황(canBeInvokedConcurrently)을 막기 위하여 Task 를 Activate 하여 Task 에서 Callback 을 호출하도록 한다. 따라서 SingleBlockCallback 을 사용하기 위해서는 하나의 Task 가 필요하며 이는 Application 에서 다음과 같이 직접 생성을 해주어야 한다.

1. Application SW Component 를 생성하고 SingleBlockCallback 의 Runnable 을 등록한다.
2. OS 에서 Task 생성

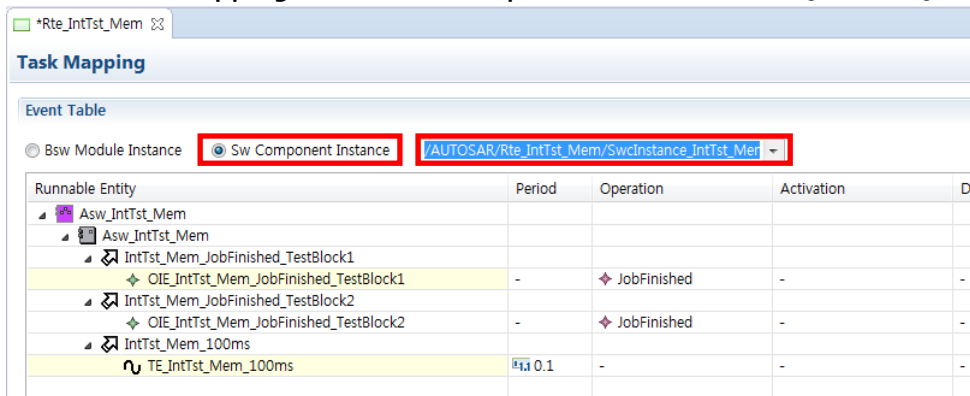


ShortName : OsTask_ASW_FG2_NvMCallback

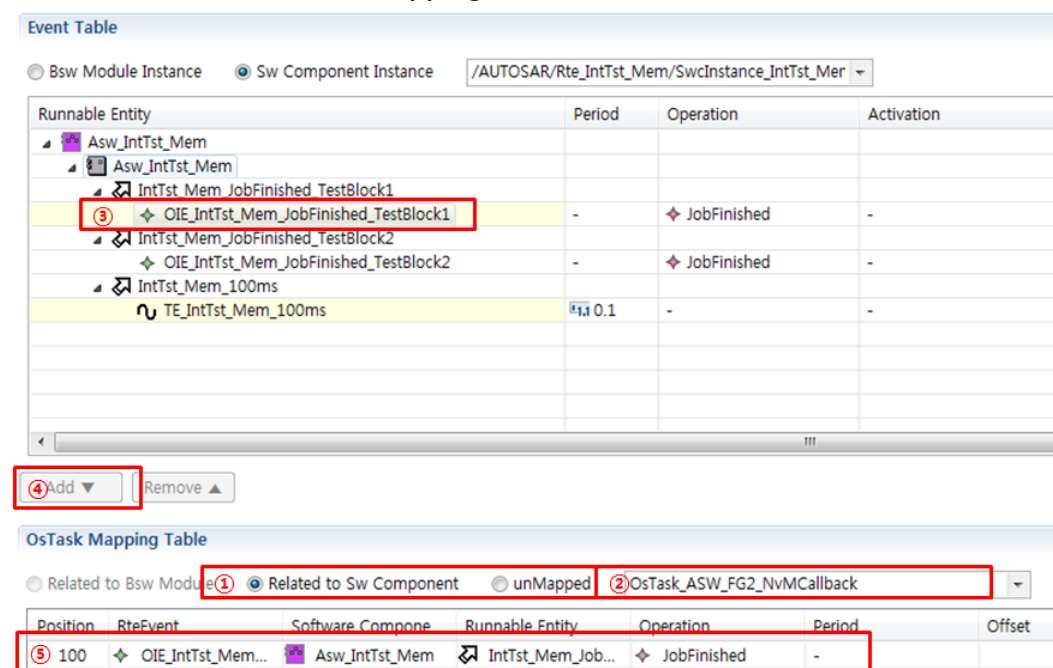
Priority : FG1 보다 크고 FG3 보다 작게 조절

ResourceRef : FG2

3. RTE 의 Task Mapping 탭에서 SW Component Instance 를 설정하고 설정한 App SW-C 를 선택



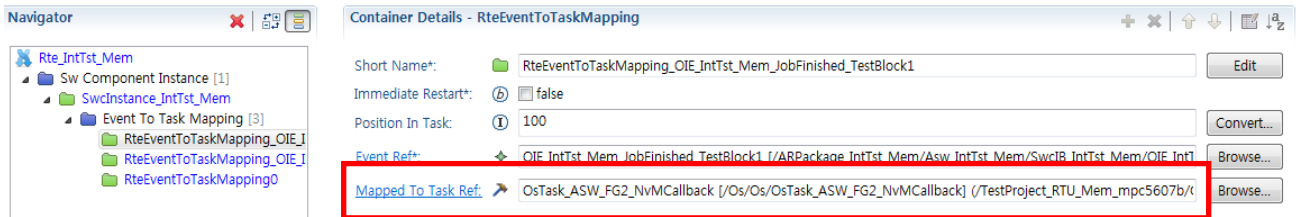
4. Callback Runnable 예 Task Mapping



- ① 최초 mapping 할때는 unMapped 로 선택, 이후에는 RelatedToSwComponent 선택
- ② OsTask_ASW_FG2_NvMCallback 선택
- ③ Mapping 할 runnable 선택
- ④ Add Click

⑤ OsTaskMappingTable 에 생성 확인

SingleBlockCallback 관련 Event 설정시 다음과 같이 (RTE > Sw ComponentInstance > EventToTaskMapping) MappedToTaskRef 를 OsTask_ASW_FG2_NvMCallback 로 설정한다.



※ SingleBlockCallback Runnable 은 FG2 의 Task 에서 호출되기 때문에 그 수행시간은 최소로 해야한다.

9.6.2 InitBlock Callback

NvM 모듈에서는 각 Block 마다 InitBlockCallback 을 지원하며, 사용시에는 Block Descriptor 의 InitBlockCallback 에 다음과 같이 설정한다. (Chap 5.1.2 참조)

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

※ Naming 규칙이 Rte_Call_NvM_PNx_InitBlock 로 되어 있는 프로젝트를 위와 같이 변경하려면 Chap 8.7 을 참고한다.

위와 같이 설정시에는 생성된 NvM Swcd 에서 다음과 같이 포트와 Interface 가 생성된다.

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

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

Application 에서는 이를 ApplicationSwComponent 에 연결시켜 사용할 수 있다. (RTE 기반 생성된 함수의 프로토타입에 대한 사항은 AUTOSAR BSW Service API Guide.doc 문서 참조)

9.7 Port Name 변경에 따른 수정건

9.7.1 InitBlock, SingleBlock Callback Name 수정

```
Ecud_NvM.arxml 또는 Ecud_NvM_IntTst_Mem.arxml 파일에서 Callback 명 수정
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 수정

다음 그림과 같이 NvM 의 Swcd 을 생성할때, P 옵션 추가

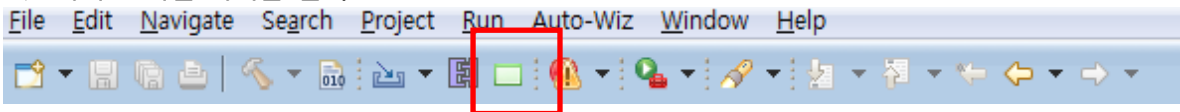

```
# GenerateNvM_S #####
moduleName = 'NvM'
Swcd_Bsw_NvM = os.path.join('swcd', 'Swcd_Bsw_NvM.arxml')

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 Ecu
    ],
    BSW_GENERATOR=NvM,
    BSWDEFINES=['S', 'P']
)

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

9.7.3 EcucValueCollection Tap 에서 Port 연결 수정

- 1) 하기 표시된 아이콘 클릭



- 2) Service and I/O Tap 클릭

노란색이 연결되어 있지 않은 port 들이며, ConfigID 와 진단관련 Block 들은 Port 를 연결해 주지 않는다.

- 3) 연결하려는 Port 에 '+' button 클릭

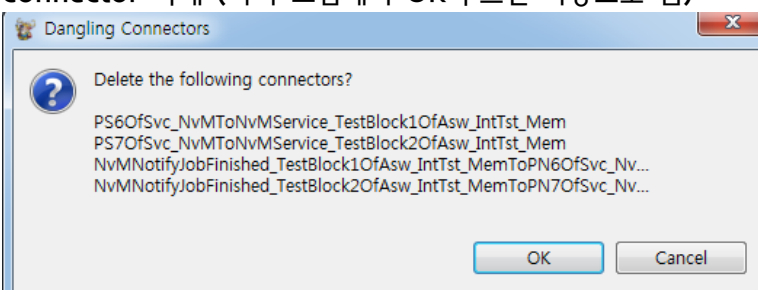
Automatic Connection

Contents	Context Component	Port Interface	Component Type	Connector
Dem	-	-	-	-
Det	-	-	-	-
EcuM	-	-	-	-
Pm	-	-	-	-
SWC_FIM	-	-	-	-
Svc_JoHwAb	-	-	-	-
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) Respect Naming Rule 체크 해제후 연결하려는 Port 체크

Contents	Context Component	Port Interface	Component Type	Connector
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]	-

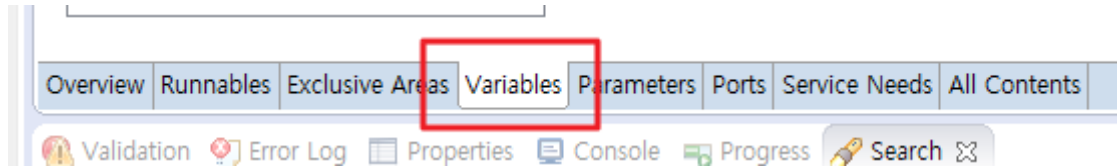
- 5) EcuExtract 파일의 EcuCompisition 에서 Assembly Connectors 탭을 클릭하여 기존에 연결되었던 connector 삭제 (하기 그림에서 OK 누르면 자동으로 됨)



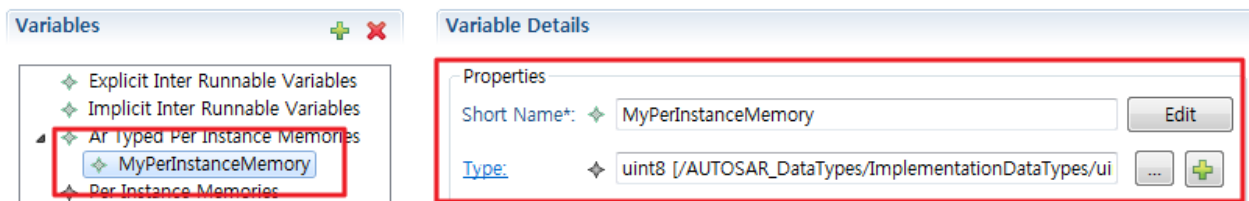
6) EcuExtract 재실행

9.8 PIM (PerInstanceMemory) 설정방법

9.8.1 ArTypedPerInstanceMemory 추가



- 1) EEPROM Data 를 사용하려는 Software Component 에서 Variables 로 이동

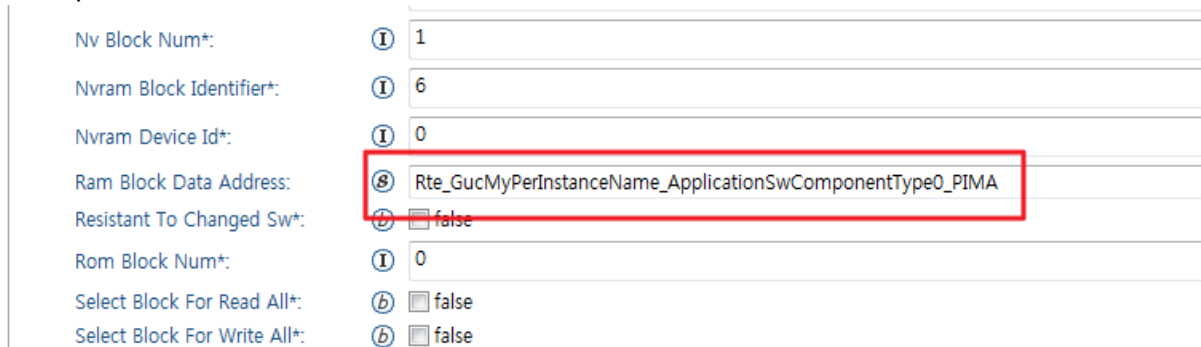


- 2) ShortName 과 Type(Implementation Data Type)을 입력
- 3) 저장 후 Build 하면, 애플리케이션에서는 Rte_Pim_<ShortName>(void) 이라는 API 를 바로 사용할 수 있고, Return 이 변수 주소가 됩니다
- 4) 이 때 해당 변수는 Rte.c 에 생성되는데, Rte_<SWC>_PerInstanceMemory_CDS.h 에 extern 변수 선언이 됩니다. NvM 과 같이 변수에 대한 직접 접근이 필요한 경우 이를 include 하여 사용할 수 있습니다.
- 5) 변수의 형식은 Rte_G<변수타입><PerInstanceMemory 이름>_<SWC 이름>_PIMA 입니다.마지막의 _PIMA 는 ArTypedPerInstanceMemory 입니다.
- 6) 변수 타입 Primitive 형 일 때, c: character, s: short, l:long, f:float, d: double 이며, unsigned 일 때는 앞에 u, signed 일 때는 s 가 붙습니다. unsigned char 일 경우, uc 입니다. Array 타입은 aa, Structure/Union 타입은 st 입니다.

ex) Rte_GstPerInstanceMemory_Rec_ApplicationSwComponentType_0_PIMC)

9.8.2 PIM (PerInstanceMemory) 을 이용하여 ReadAll/WriteAll 을 사용할 경우

- 1) NvM 에서는 Rte_<SWC>_PerInstanceMemory_CDS.h 에 extern 으로 선언된 변수의 이름을 Block Descriptor 의 Ram Block Data Address 에 입력



- 2) Header File Inclusion 을 위해 아래와 같이 NvM > NvMCommon > NvMUserIncludeFiles 에 Rte_<SWC>_PerInstanceMemory_CDS.h 를 추가

Navigator

- NvM
 - Block Descriptor [7]
 - NvMCommon
 - NvMUserIncludeFiles
 - NvMDemEventParameterRefs_0

Container Details - NvMUserIncludeFiles

Short Name*: NvMUserIncludeFiles

User Include File*: Rte_<SWC>_PerInstanceMemory_CDS.h