| SCOPE OF APPLICATION<br>All Project/Engineering | HYUNDAI<br>AutoEver | SHT/SHTS<br>1 / 22 |
|---|---|---|
| Responsibility:<br>클래식오토사팀 | AUTOSAR CtrlRam User Manual | DOC. NO |
| AUTOSAR Controlled RAM User Manual | | |

| Document Change History | | | | |
|---|---|---|---|---|
| Date (YYYY-MM-DD) | Ver. | Editor | Chap | 내용(개정 전 -> 개정 후) |
| 2015-05-11 | 1.0.0 | ChangYong Song<br>JongHyun Baek | All | • Initial Creation (Draft – Will be updated in later)<br>• Reviewed by SangYong Han |
| 2016-05-18 | | HyungHo Lee | 3,4,5,6 | • Minor-modification(Font, Location, Bold)<br>• Add some descriptions. (Section 3)<br>• Put some texts in xenon controlled ram user manual into Part 6.3 |
| 2016-08-09 | 1.0.1 | HyungHo Lee | 4.3 | Release 1.0.1 |
| 2016-10-14 | 1.0.2 | HyungHo Lee | 3,4 | Release 1.0.2 |
| 2017-02-09 | 1.0.3 | HyungHo Lee | 3,4 | Release 1.0.3 |
| 2017-05-15 | 1.0.4 | HyungHo Lee | 3.2.1<br>3.2.2<br>4.3<br>5.1 | • Release 1.0.4<br>Update the following descriptions. (Add Constraints) |
| 2017-09-06 | 1.0.5 | HyungHo Lee | 4.3 | Release 1.0.5 |
| 2019-10-16 | 1.0.5.0 | YongHyun Han | 4.3<br>5.1 | • Release 1.0.5.0<br>• Change category of some parameters 'Fixed' -> 'Changeable'<br>Change file structure (Moving lib folder to delivery) |
| 2021-08-04 | 1.0.6.1 | YongHyun Han | All<br>5.1.1<br>4.3.2,<br>8.3 | • Apply new document template<br>• Add caution of RAM area overlap<br>• Add new MCU support (S32K, CYTXXX) |
| 2022-01-03 | 1.0.7.0 | YongHyun Han | 4.3 | • Release 1.0.7.0 |
| 2022-08-09 | 1.0.8.0 | YongHyun Han | 4.3 | • Release 1.0.8.0 |

| | | | | |
|---|---|---|---|---|
| Edition Date : 2022-08-09 | File Name: | **Creation** | **Check** | **Approval** |
| Document Management System | CtrlRam_UM.docx | **YongHyun Han** | **HoiMin Kim** | **JiHoon Jung** |

# 목차

# 1. Overview

본 문서는 Controlled RAM사용을 위해 AUTOSAR 플랫폼 사용할 때, 사용자가 파라미터 설정 또는 시스템 설계를 할 때 주의하거나 참고할 사항을 제공한다. 자세한 사항은 Reference 문서를 참고한다.

설정관련 Category의 해석은 다음과 같다.

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

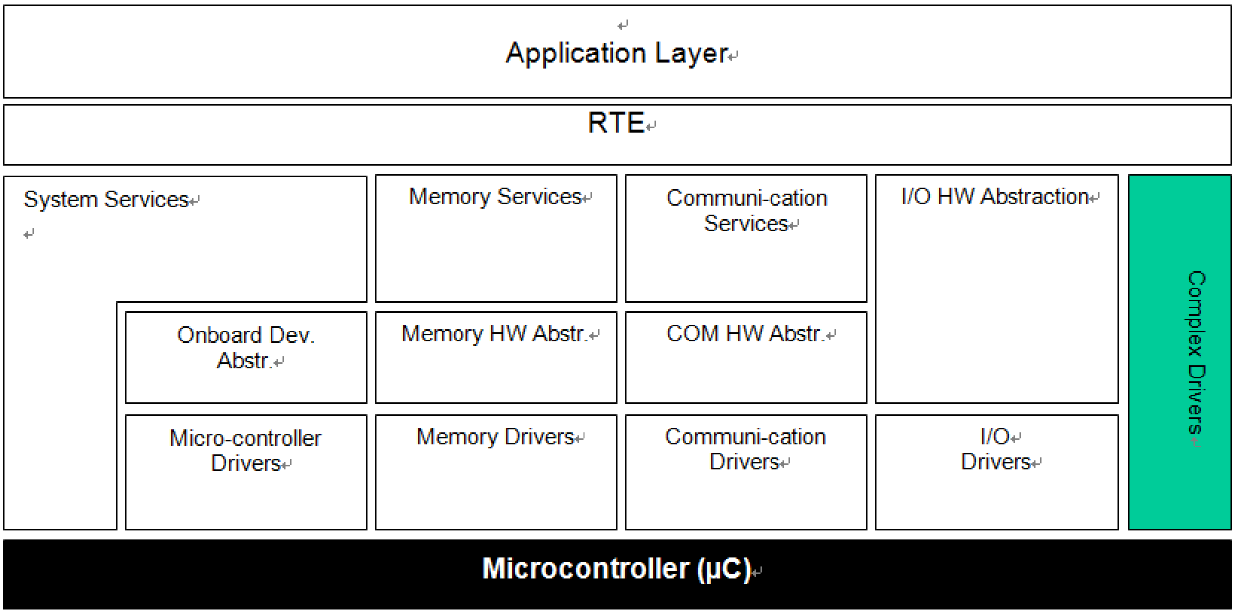
# 2. Reference

| Sl. No. | Title | Version |
|---|---|---|
| 1. | Microcontroller(MPC560xB, Tc2x, RH850 F1x) Reference Manual | The Latest Version |
| 2. | Microcontroller(MPC560xB, Tc2x, RH850 F1x) Data Sheet | The Latest Version |
| | | |
| | | |

# 3. AUTOSAR System

## 3.1 Overview of Software Layers

CtrlRam module is designed as a CDD(Complex Device Driver) in Autosar Platform. Therefore, an application user can't request to change or modify Controlled RAM Module just for Autosar standard specifications or other reasons, which means that all application logic should be designed in accorance with Controlled RAM Module's structure and service. So, If application logic is not matched with functions provided by Controlled Ram Module, an user must modify the application logic, not request to change(or update) Controlled RAM's structure or internal design.



## 3.2 Controlled RAM

The Controlled RAM area is a dedicated RAM area for storing data that will not be lost when a warm reset occurs. The controlled RAM data must be updated through dedicated functions explained in API interface. In order to prevent Controlled RAM area from unsuitable modifications, checks must be performed on its content.

The initialization and the data integrity survey routines are used in order to prevent controlled RAM area from unsuitable modifications. Checks must be performed on request while the software is running and at reset for recovering data after a power down.
A recovery mechanism can be used for avoiding loss of all the Controlled RAM data in case of a reset during Controlled RAM variable update. In this case only the last up-date is lost. This recovery mechanism is defined for the complete Controlled RAM only and is selected by the SWI.

The Controlled RAM functionality provides:

- a routine for initialization of the Controlled RAM area
- a recovery mechanism for preventing data loss in case of reset during Controlled RAM data update
- a routine for data integrity survey of the Controlled RAM area
- dedicated routines for updating the Controlled RAM variables and structures:
  - byte update, word update, long update, structure update

Note:
Asynchronous reset will possibly corrupt SRAM if it asserts during a read or write operation to SRAM in the SPC/MPC560xB MCU. Because SRAM initialization must be needed in this case, the controlled RAM is lost.
Please refer to the reference manual and data sheet of MCU

### 3.2.1 Controlled RAM data initialization

The RTSW can define routines for updating the Controlled RAM data at initialization time in case this area was not valid after a reset. By default if the Controlled RAM area is not valid after a reset, the Controlled RAM area is cleared.

Note:
It MUST not use the Controlled RAM data update routines to update Controlled RAM data during initialization callback (the security mechanism is recomputed after all initializations).

### 3.2.2 General constraint of Controlled RAM

- No Controlled RAM data updates are allowed in any interrupt service routines. Therefore, The job like this is only allowed in task-base.

- After requesting to low power mode or high power mode, It is not recommended to try to update(read, write) ram datas because system environments are changed while dealing with the process of shifting to low power mode or high power mode.

- Controlled RAM APIs are not allowed to use while system is in low power mode(including service like low power task).

- the ongoing Job using Controlled RAM's API must not be again performed or requested before the previous job is finished.

- If a recovery mechanism is used, there must be at least one of CONTROLLED_RAM_CONF and

LargestDataTypeSize must take a size of the largest data type defined in the Controlled RAM.

- If a part of whole controlled ram datas is unexpectedly broken or loss, an user should request reset mode to BswM because the broken or loss data is recovered only after a reset occurs. An user can get the information that Ctrl Ram's state is wrong through DEM Event.
- Whenever CtrlRam's main function is called, Check if datas are broken or loss, and all interrupt is disabled while the checking-job is executed. The time taken to finish checking the entire CtrlRam data depends on system clock and data type size that an user configured.
  (Refer to blocking time below)

  Ex) Interrupt Blocking Time : about 4us

  Test Environment - System clock : 64Mhz
  Data Type Size(the largest) : 100 byte

# 4. Product Release Notes

## 4.1 Overview

이 Chapter에서는, 현대오토에버 CtrlRam Module에 대한 release 관련 내용을 제공하는데 목적이 있으며, CtrlRam Module release version에 대한, 제한사항 및 특이사항을 기술하고 있다.

## 4.2 Scope of the release

이 문서에 대한 모든 내용은, 다음의 현대오토에버 CtrlRam 모듈에 한정한다.

| Module | Autosar version | SWS version | Module version |
|---|---|---|---|
| CtrlRam | - | - | 1.0.8 |

## 4.3 Module release notes

### 4.3.1 Change Log

#### 4.3.1.1 Version 1.0.8.0

- ➢ 개선 사항
  - ■ UNECE Cyber Security 법규 대응 보안 코딩 개선

| | |
|---|---|
| • 원인 | • UNECE Cyber Security 법규 대응 필요 |
| • 동작 영향 | • 없음 |
| • 설정 영향 | • 없음 |
| • ASW 조치 필요 사항 | • 없음 |

#### 4.3.1.2 Version 1.0.7.0

- ➢ 개선 사항
  - ■ UNECE Cyber Security 법규 대응 보안 코딩 개선

| | |
|---|---|
| • 원인 | • UNECE Cyber Security 법규 위반 사항 발생 |
| • 동작 영향 | • 없음 |
| • 설정 영향 | • 없음 |
| • ASW 조치 필요 사항 | • 없음 |

#### 4.3.1.3 Version 1.0.6.1

- ➢ 개선 사항

■ 신규 MCU 지원

| | |
|---|---|
| • 원인 | • 신규 MCU 지원 (S32K, CYTXXX) |
| • 동작 영향 | • 없음 |
| • 설정 영향 | • 없음 |
| • ASW 조치 필요 사항 | • 없음 |

■ RAM 영역 중첩 확인

| | |
|---|---|
| • 원인 | • CtrlRam RAM 영역과 타 모듈 RAM 영역 중첩 회피 필요 |
| • 동작 영향 | • 없음 |
| • 설정 영향 | • 없음 |
| • ASW 조치 필요 사항 | • 없음 |

### 4.3.1.4 Version 1.0.6.0

➢ 개선 사항

■ 정적 분석 수행

| | |
|---|---|
| • 원인 | • 정적 분석 위반사항에 대해 수정 또는 정당화 필요 |
| • 동작 영향 | • 없음 |
| • 설정 영향 | • 없음 |
| • ASW 조치 필요 사항 | • 없음 |

### Version 1.0.5.0

➢ 개선 사항

■ 파일 구조 변경 및 parameter category 변경

| | |
|---|---|
| • 원인 | • Code Open 을 위해 파일 구조 변경 및 parameter 변경 필요 |
| • 동작 영향 | • 없음 |
| • 설정 영향 | • 없음 |
| • ASW 조치 필요 사항 | • 없음 |

■ CtrlDevErrorDetect 설정에 따른 의존성 제거

| | |
|---|---|
| • 원인 | • CtrlDevErrorDetect 파라미터 False 시에 컴파일 에러 발생 |
| • 동작 영향 | • 없음 |
| • 설정 영향 | • 없음 |
| • ASW 조치 필요 사항 | • 없음 |

### 4.3.1.5 Version 1.0.5

➢ 개선 사항

■ Compile Warning 수정

| | |
|---|---|
| • 원인 | • Compile Warning 발생(W767) |
| • 동작 영향 | • 없음 |
| • 설정 영향 | • 없음 |
| • ASW 조치 필요 사항 | • 없음 |

### 4.3.1.6 Version 1.0.4

➢ 개선 사항

■ MainFunction 의 Exclusive 제어 방법 개선

■ User manual 업데이트 (General constraint of Controlled RAM section)

| | |
|---|---|
| • 원인 | • MainFunction 의 Exclusive 제어 방법 개선 |
| • 동작 영향 | • 없음 |
| • 설정 영향 | • RTE CtrlRam Exclusive Area Impl 설정 변경 |
| • ASW 조치 필요 사항 | • 없음 |

### 4.3.1.7 Version 1.0.3

➢ 개선 사항

■ User Manual 업데이트

| | |
|---|---|
| • 원인 | • Callback configuration type (Fixed → Changeable) |
| • 동작 영향 | • 없음 |
| • 설정 영향 | • Application User 가 Callback 설정 변경 가능하도록 수정 |
| • ASW 조치 필요 사항 | • 없음 |

### 4.3.1.8 Version 1.0.2

➢ 신규 기능

■ N/A

➢ 개선 사항

■ Rearrange memory section

■ Update User manual

### 4.3.1.9 Version 1.0.1

➢ 신규 기능

■ N/A

➢ 개선 사항

■ Update User manual

#### 4.3.1.10 Version 1.0.0

- 신규 기능
  - First Release

- 개선 사항
  - N/A

### 4.3.2 Limitations

- Refer to Chapter 3.2
- If the CtrlRam memory section overlaps with other modules such as FBL and HSM, the CtrlRam area may be cleared or corructed unintentionally. So CtrlRam memory section must not overlap with other modules's RAM area.

### 4.3.3 Deviation

None

# 5. Configuration Guide

## 5.1 CtrlRam 모듈

### 5.1.1 CtrlRamGeneral

다음 설정을 참고한다.

| Parameter Name | Value | Categor |
|---|---|---|
| 1)Platform Architecture | BIG_ENDIAN_PLATFORM | C |
| 2)Recovery | true | C |
| 3)Verify At Init | true | C |
| 4)Dev Error Detect | true | C |

1) Bolero, Chorus : BIG_ENDIAN_PLATFORM
TC2x : LITTLE_ENDIAN_PLATFORM
RH850 : LITTLE_ENDIAN_PLATFORM
S32K, CYTXXX : according to AIRCR.ENDIANNESS bit (default LITTLE_ENDIAN_PLATFORM)

2) A recovery of the Controlled RAM data will be attempted when a reset occures during data update in this area.

## 5.1.2 Config Set

| Parameter Name | Value | Category |
|---|---|---|
| Data Type Size | Unsigned word | C |
| Callback | Text | C |

### 5.1.2.1 LargestDataTypeSize

| Attribute Name | LargestDataTypeSize |
|---|---|
| Description | Size of the largest data type defined in the Controlled RAM. [This information will be used in case the Recovery mechanism is enabled. |
| Type | Uint16 |
| Range | 1 – 65535 |
| Default Value | - |

### 5.1.2.2 CallBack

| Attribute Name | CallBack |
|---|---|
| Description | Callback routine - called at startup in order to inform the ASW if the Controlled RAM area was not valid after Reset. The ASW can use this routine to reinitialize the data in the Controlled RAM. Set this to NULL if no callback routine is used. |
| Type | String |
| Range | NULL or Function Name |
| Default Value | NULL |

## 5.1.3 CtrlRamDemEventParameterRefs

| Attribute Name | CtrlRamDemEventParameterRefs |
|---|---|
| Description | Notify DEM of CtrlRam's wrong state. |
| Type | CTRLRAM_E_WRONG_STATE |
| Dem Event Parameter Ref | User-Defined |
| Default Value | NULL |

# 6. Application Programming Interface (API)

## 6.1 Type Definitions

### 6.1.1 CtrlRam_StatusType

| | | | |
|---|---|---|---|
| Type: | uint8 | | |
| Range | CRAM_OK | 0x00 | Controlled RAM area was valid after reset |
| | CRAM_LOST | 0x01 | Controlled RAM area was not valid after reset |
| Description: | This is a request result returned by the API service Ctrlram_GetStatus. | | |

## 6.2 Macro Constants

None

## 6.3 Functions

### 6.3.1 Internal Functions

#### 6.3.1.1 Initialization/Deinitialization Related API

| | |
|---|---|
| *Function Name* | CtrlRam_Init |
| *Syntax* | FUNC(void, CTRLRAM_CODE) CtrlRam_Init (void) |
| *Service ID* | |
| *Sync/Async* | Synchronous |
| *Reentrancy* | |
| *Parameters (In)* | None |
| *Parameters (Inout)* | None |
| *Parameters (Out)* | None |
| *Return Value* | None |
| *Description* | This routine initializes the Controlled RAM area. |
| *Preconditions* | None |
| *Configuration Dependency* | |

#### 6.3.1.2 Scheduled Functions

| | |
|---|---|
| Function Name | CtrlRam_MainFunction |
| Syntax | FUNC(void, CTRLRAM_CODE) CtrlRam_MainFunction(void) |
| Service ID | |

| | |
|---|---|
| **Sync/Async** | Synchronous |
| **Reentrancy** | |
| **Parameters (In)** | None |
| **Parameters (Inout)** | None |
| **Parameters (Out)** | None |
| **Return Value** | None |
| **Description** | This routine surveys the Controlled RAM area integrity. A reset is performed in case the Controlled RAM area is corrupted. |
| **Preconditions** | None |
| **Configuration Dependency** | None |

## 6.3.2 APIs available for application

### 6.3.2.1 ByteWrite

| | | |
|---|---|---|
| **Function Name** | bsramctl_ByteWrite | |
| **Syntax** | FUNC(Std_ReturnType, CTRLRAM_CODE)<br>bsramctl_ByteWrite (uint8 *lpub_PtrByte, uint8 lub_Value) | |
| **Service ID** | | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Non Reentrant | |
| **Parameters (In)** | PtrBlock | Address of the variable to be updated |
| | Value | New Value |
| **Parameters (Inout)** | None | |
| **Parameters (Out)** | None | |
| **Return Value** | Std_ReturnType | E_OK:<br>E_NOT_OK: |
| **Description** | These routines must be used for updating a single byte in the Controlled RAM. The byte can also be a part of a structure defined in the controlled RAM area.<br>There is no check performed for the given address to be in the controlled RAM.<br>The update is done in a section protected against OS preemption.<br>No Controlled RAM data updates are allowed in interrupt service routines (no check is performed). | |
| **Preconditions** | CtrlRam should be initialized. | |
| **Configuration Dependency** | | |

| | | |
|---|---|---|
| **In Communication with application SW-C** | Rte_Call_<P>_ByteWrite (uint8 *lpub_PtrByte, uint8 lub_Value)<br><P> : R-Port Name | |

### 6.3.2.2 WordWrite

| | | |
|---|---|---|
| **Function Name** | bsramctl_WordWrite | |
| **Syntax** | FUNC(Std_ReturnType, CTRLRAM_CODE)<br>bsramctl_WordWrite (uint16 * lpuw_PtrWord, uint16 luw_Value) | |
| **Service ID** | | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Non Reentrant | |
| **Parameters (In)** | PtrBlock | Address of the variable to be updated |
| | Value | New Value |
| **Parameters (Inout)** | None | |
| **Parameters (Out)** | None | |
| **Return Value** | Std_ReturnType | E_OK:<br>E_NOT_OK: |
| **Description** | These routines must be used for updating a word in the Controlled RAM. The word can also be a part of a structure defined in the controlled RAM area.<br>There is no check performed for the given address to be in the controlled RAM.<br>The update is done in a section protected against OS preemption.<br>No Controlled RAM data updates are allowed in interrupt service routines (no check is performed). | |
| **Preconditions** | CtrlRam should be initialized. | |
| **Configuration Dependency** | | |
| **In Communication with application SW-C** | Rte_Call_<P>_ WordWrite (uint16 * lpuw_PtrWord, uint16 luw_Value)<br><P> : R-Port Name | |

### 6.3.2.3 LongWrite

| | | |
|---|---|---|
| **Function Name** | bsramctl_LongWrite | |
| **Syntax** | FUNC(Std_ReturnType, CTRLRAM_CODE)<br>bsramctl_LongWrite (uint32 * lpul_PtrLong, uint32 lul_Value) | |
| **Service ID** | | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Non Reentrant | |
| **Parameters (In)** | PtrBlock | Address of the variable to be updated |

| | | |
|---|---|---|
| | Value | New Value |
| **Parameters (Inout)** | None | |
| **Parameters (Out)** | None | |
| **Return Value** | Std_ReturnType | E_OK:<br>E_NOT_OK: |
| **Description** | These routines must be used for updating a long in the Controlled RAM. The long can also be a part of a structure defined in the controlled RAM area.<br>There is no check performed for the given address to be in the controlled RAM.<br>The update is done in a section protected against OS preemption.<br>No Controlled RAM data updates are allowed in interrupt service routines (no check is performed). | |
| **Preconditions** | CtrlRam should be initialized. | |
| **Configuration Dependency** | | |
| **In Communication with application SW-C** | Rte_Call_<P>_ LongWrite (uint16 * lpuw_PtrWord, uint16 luw_Value)<br><P> : R-Port Name | |

### 6.3.2.4 StructWrite

| | | |
|---|---|---|
| **Function Name** | bsramctl_StructWrite | |
| **Syntax** | FUNC(Std_ReturnType, CTRLRAM_CODE)<br>bsramctl_StructWrite (uint8 * lpub_Struct, const uint8 * lpub_Src, uint8 lub_Size) | |
| **Service ID** | | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Non Reentrant | |
| **Parameters (In)** | PtrBlock | Address of the structure to be updated |
| | PtrSrc | Address of the source structure containing new data |
| | Length | Structure size in bytes |
| **Parameters (Inout)** | None | |
| **Parameters (Out)** | None | |
| **Return Value** | Std_ReturnType | E_OK:<br>E_NOT_OK: |
| **Description** | This routine must be used for updating a structure or an array defined in the Controlled RAM area.<br>There is no check performed for the given address to be in the controlled RAM. | |

| | |
|---|---|
| | The update is done in a section protected against OS preemption.<br>No Controlled RAM data updates are allowed in interrupt service routines (no check is performed). |
| **Preconditions** | CtrlRam should be initialized. |
| **Configuration Dependency** | |
| **In Communication with application SW-C** | Rte_Call_<P>_ StructWrite (uint8 * lpub_Struct, const uint8 * lpub_Src, uint8 lub_Size)<br><P> : R-Port Name |

### 6.3.2.5 GetStatus

| | | |
|---|---|---|
| **Function Name** | Ctrlram_GetStatus | |
| **Syntax** | FUNC(Std_ReturnType, CTRLRAM_CODE) Ctrlram_GetStatus (P2VAR(CtrlRam_StatusType, AUTOMATIC , CTRLRAM_APPL_DATA) StatusPtr) | |
| **Service ID** | | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | | |
| **Parameters (In)** | None | |
| **Parameters (Inout)** | None | |
| **Parameters (Out)** | StatusPtr | Pointer to store the requested result. |
| **Return Value** | None | |
| **Description** | Service to read the CtrlRam Status information. | |
| **Preconditions** | None | |
| **Configuration Dependency** | None | |
| **In Communication with application SW-C** | Rte_Call_<P>_ GetStatus (P2VAR(CtrlRam_StatusType, AUTOMATIC , CTRLRAM_APPL_DATA) StatusPtr)<br><P> : R-Port Name | |

# 7. Generator

## 7.1 Generator Option

None

## 7.2 Generator Error Message

### 7.2.1 CtrlRam

#### 7.2.1.1 Error Messages

None

#### 7.2.1.2 Warning Messages

None

#### 7.2.1.3 Information Messages

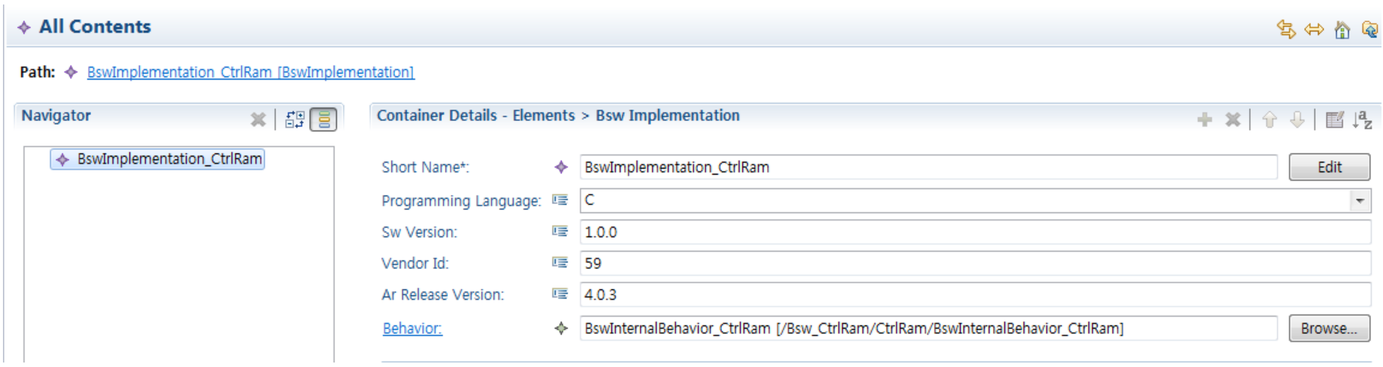
None

# 8. Appendix

## 8.1 Bswmd (Bsw Module Description)

### 8.1.1 Bsw 모듈 version 설정

각 모듈을 컴파일할 때, version 정보가 맞지 않으면 Compile에서 Error를 발생시킨다.
이때는 Bswmd의 다음과 같이 BswImplementation Container에서 version 정보를 수정해야 한다.


All Contents
Navigator
BswImplementation_CtrlRam
Container Details - Elements > Bsw Implementation
Edit
Browse...

Path: BswImplementation_CtrlRam [BswImplementation]

Short Name*: BswImplementation_CtrlRam
Programming Language: C
Sw Version: 1.0.0
Vendor Id: 59
Ar Release Version: 4.0.3
Behavior: BswInternalBehavior_CtrlRam [/Bsw_CtrlRam/CtrlRam/BswInternalBehavior_CtrlRam]

## 8.2 Exclusive Areas

| Module | SchM APIs |
|---|---|
| CtrlRam | SchM_Enter_CtrlRam_SECURITY_CHECK_PROTECTION ()<br>SchM_Exit_CtrlRam_SECURITY_CHECK_PROTECTION ()<br>SchM_Enter_CtrlRam_WRITE_PROTECTION ()<br>SchM_Exit_CtrlRam_WRITE_PROTECTION ()<br>SchM_Enter_CtrlRam_STRUCT_WRITE_PROTECTION ()<br>SchM_Exit_CtrlRam_STRUCT_WRITE_PROTECTION () |

Note:
SECURITY_CHECK_PROTECTION 및 STRUCT_WRITE_PROTECTION 의 Exclusive 시간은 사용하는 Controlled RAM 의 전체 크기 및 Write 하는 Structure 크기에 따라 변동된다.

## 8.3 Memory Mapping

Controlled RAM 의 정상적인 동작을위해서는 각 메모리 Section은 다음과 같은 특정한 순서를 준수하여야 한다. 해당 메모리 section의 순서는 Linker Script 파일에 정의되어야 한다.

Memory mapping

| RAM areas | Section Name |
|---|---|
| Controlled RAM (NO CLEAR) | CTRLRAM_RAM_VAR_NO_INIT_BEGIN |
| | CTRLRAM_RAM_VAR_POWER_ON_CLEARED_8 |
| | CTRLRAM_RAM_VAR_POWER_ON_CLEARED_16 |
| | CTRLRAM_RAM_VAR_POWER_ON_CLEARED_32 |
| | CTRLRAM_RAM_VAR_POWER_ON_CLEARED_UNSPECIFIED |
| | CTRLRAM_RAM_VAR_NO_INIT_END |

또한 CtrlRam 에서 사용하는 메모리 section이 FBL 또는 HSM 등의 다른 모듈이 사용하는 메모리 section과 겹쳐지면, CtrlRam 메모리 영익이 의도치않게 지워지거나 오염 될 수 있다. 따라서 FBL 또는 HSM 등의 User Manual 등을 통해 CtrlRam 메모리 영역과 다른 모듈의 메모리 영역이 중첩 되지 않도록 한다.

## 8.4 Example

### 8.4.1 Initialization

Defined Controlled RAM variable:

```
#define CTRLRAM_START_SEC_VAR_POWER_ON_CLEARED_8
#include "MemMap.h"
uint8 ctrlram_TestByte;
```

Callback routine:

```
#define Asw_IntTst_CtrlRam_START_SEC_CODE
#include "Asw_IntTst_CtrlRam_MemMap.h"
void CtrlRamConfigSet1_CB(void)
{
  ctrlram_TestByte = 0x08;
}
#define Asw_IntTst_CtrlRam_STOP_SEC_CODE
#include "Asw_IntTst_CtrlRam_MemMap.h"
```

### 8.4.2 ByteWrite

```
#define Asw_IntTst_CtrlRam_START_SEC_CODE
#include "Asw_IntTst_CtrlRam_MemMap.h"

FUNC(void,Asw_IntTst_CtrlRam_CODE) IntTst_CtrlRam_10ms(void)
{
  uint8 lub_TestByte;

  lub_TestByte = ctrlram_TestByte+1;
  Rte_Call_CtrlRamService_ByteWrite(&ctrlram_TestByte, lub_TestByte);

}
#define Asw_IntTst_CtrlRam_STOP_SEC_CODE
#include "Asw_IntTst_CtrlRam_MemMap.h"
```