AUTOSAR OS Improvement Code User Manual

Document Change History				
Date (YYYY-MM-DD)	Ver.	Editor	Chap	내용(개정 전 -> 개정 후)
2020-03-10	1.0.0.0	MJ.Woo	All	• Initial Creation
2020-07-03	2.0.0.0	JH.Cho	4	• Update Change Log
2020-08-14	2.0.1.1	JH.Cho	4	• Update Change Log
2020-12-22	2.1.0.0	JH.Cho	4	• Update Change Log
2022-03-10	2.1.1.0	MJ.Woo	All 4	• Change company name • Update Change Log
2022-05-12	2.1.2.0	YH.Han	4 8.1.2	• Update Change Log • Add WDT reset in case of ECC RAM
2022-08-10	2.1.2.1	JC.Kim	1 4	• Update Copyright comment • Update Change Log
2023-02-23	2.1.3.0	JC.Kim	4	• Update Change Log
2023-04-12	2.1.3.1	JC.Kim	4	• Update Change Log
2023-09-08	2.2.0.0	HG.Kim	4	• Update Change Log

Table of Contents

1. Overview

OS Improvement Code 는 AUTOSAR OS 와 MCU 의 Integration 에서 요구되는 추가적인 기능을 담당한다. 즉, AUTOSAR OS 의 Specification 문서에서 정의되지 않았지만 MCU 위에서 AUTOSAR OS 를 구동시키기 위해 필요한 기능을 제공한다.
OS Improvement Code 사용 시 보다 자세한 기능적인 설명이 필요한 경우 Reference 문서를 참조해야 한다.

설정관련 Category 의 해석은 다음과 같다.

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

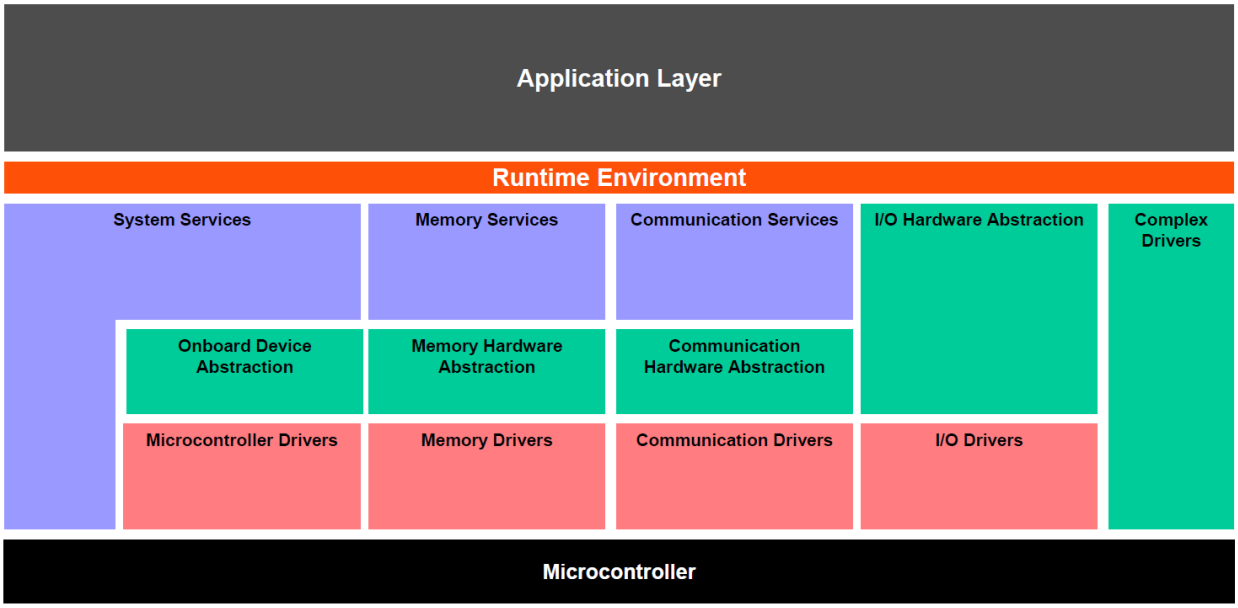
2. Reference

Sl. No.	Title	Version
1.	002-19314_0C_V_TRAVEO II BODY ENTRY ARCHITECTURE TRM.pdf	Rev. *C
2.	Os_UM.pdf	1.1.0.0

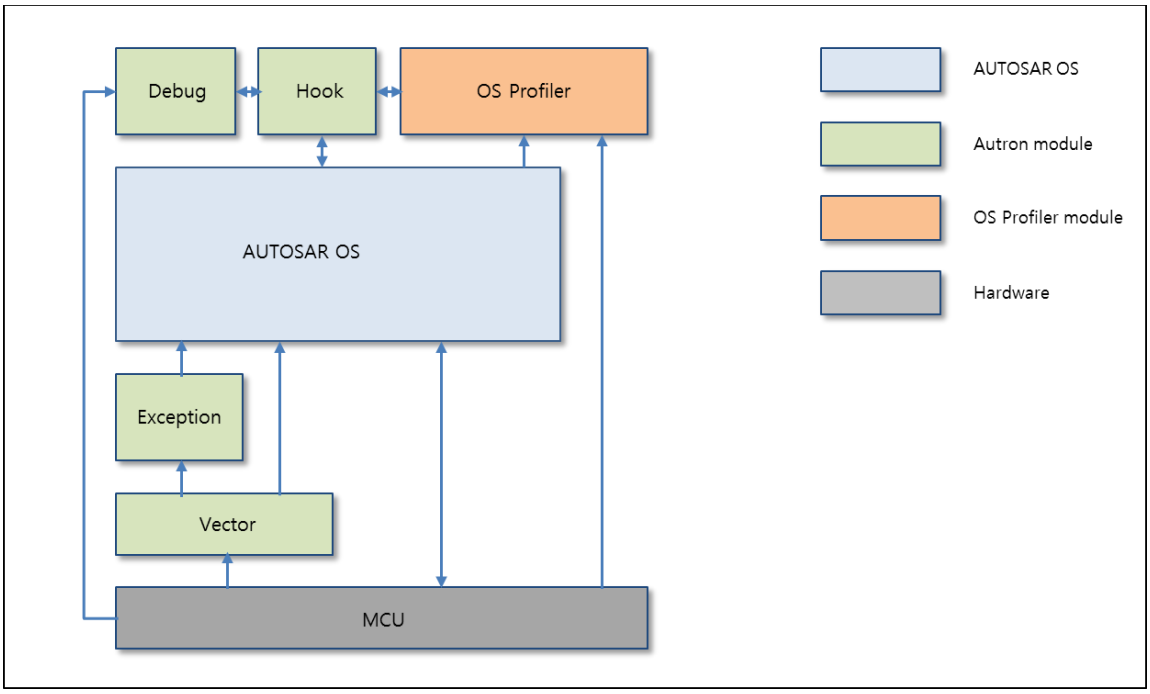
3. AUTOSAR System

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.



AUTOSAR OS 는 System Service 에 속하며, 하기 그림과 같은 구조를 가진다. AUTOSAR OS 를 기반으로 Vector, Exception, Debug, Hook 이 추가되어 전체적인 AUTORSAR OS 모듈을 완성한다. 이들 추가되는 코드들은 OS Improvement Code 에 포함된다.



3.2 OS Improvement Code

3.2.1 Startup

OS Improvement Code 는 RTSW(Runtime Software)의 startup 과정에서 다음 기능을 수행한다.

- **ECC 처리 설정**
  FAULT_STRUCT1 레지스터를 설정해 Non-correctable RAM ECC error 와 Non-correctable PFLASH ECC error 를 탐지할 수 있도록 한다.
  CPUSS_CM4_NMI_CTL0 레지스터를 설정해 FAULT_STRUCT1 에서 탐지한 ECC error 를 NMI 로 발생시킨다. 이후의 처리는 Os_CallBackNMInterrupt 에서 한다.

- **RAM 초기화**
  Power On Reset 을 포함하는 Destructive Reset 의 경우, ECC 초기화를 위해 SRAM 을 초기화 한다. 초기화 하는 RAM 의 범위는 Ram Sector 설정에서 OsImpRamInitProperty 가 'PowerOnReset'으로 지정된 sector 들이다.

3.2.2 Memory ECC error 처리

여기에서는 OS Improvement Code 에서의 Memory ECC error 처리에 대해 설명한다.

[Correction 가능한 ECC error]
SWP 에서는 correction 가능한 ECC error 의 처리에 관여하지 않음.

[Correction 불가능한 ECC error]

Memory Type		지원 방법		기타 사항
		Handling	Limitation	
RAM		RAM 전체 clear 후, AppCallbackOnSystemError 호출. 이후 MCU Reset	영구적인 HW 손상으로 인한 ECC error 인 경우 복구 불가능. 이 경우 ECC error 반복 발생 가능.	Callback 전달인자: E_OS_SYS_RAMECC
ROM	PFLASH	OS shutdown / ShutdownHook 에서 AppCallbackOnSystemError 호출 후 MCU Reset	복구 불가능. ECC error 반복 발생 가능.	Callback 전달인자: E_OS_SYS_PFLASHECC
	DFLASH	없음(MCAL 에서 모두 처리)	-	MCAL 의 FLS 모듈 참고

Note: RAM ECC error 발생시 AppCallbackOnSystemError() 는 모든 RAM 영역이 초기화된 후 호출되므로 callback 안에서는 모든 전역변수가 신뢰할 수 없는 값을 가진 상태이다.

3.2.3 CPU Load

The CPU load corresponds to the time spent for scheduler activity, execution of tasks and interrupts. The IT load corresponds to the time spent for execution of interrupts, but it does not take into account the Category1 interrupts.
The CPU load corresponds to the time spent for scheduler activity, execution of tasks and interrupts. The IT load corresponds to the time spent for execution of interrupts, but it does not take into

account the Category1 interrupts.

- CPU load = TASK execution time + CAT1ISR execution time + CAT2ISR execution time + OS execution time
- IT load = CAT2ISR execution time

It is calculated over period of mapped periodic Task (Default: 10 ms) and expressed in ‰
If you want to measure CPU/IT load in different period, change the configuration according to Appendix 8.3.1.

The load measurement is started after execution of Task that Timing Event for CPU/IT Load is mapped.
You can restart load measurement by calling the API.

- Os_UserGetCPULoad(LpLoad, TRUE)

You can get the current load value at the debugger through following variables.
- Os_GusCPULoad : Current CPU Load
- Os_GusCPULoadPeak : CPU Load Peak value
- Os_GusCPULoadMean : CPU Load Mean value
- Os_GusITLoad : Current Interrupt Load
- Os_GusITLoadPeak : IT Load Peak value
- Os_GusITLoadMean : IT Load Mean value

HW dependency
- To measure the CPU/IT load the 32 bit free running timer “TCPWM0 Group #2, Counter #0” is used.

3.2.4 Stackdepth

There is an API function which allows to get the number of used bytes of the user stack(Please refer OS User Manual).
The stack area is initialized with a specific pattern. When the stack depth is calculated each byte from the user stack is compared with specific pattern to obtain the stack depth of user stack.

4. Product Release Notes

4.1 Overview

이 Chapter 에서는, OS Improvement Code 에 대한 release 관련 내용을 제공하는데 목적이 있으며, OS Improvement Code Software product release version 에 대한, 제한사항 및 특이사항을 기술하고 있다.

4.2 Scope of the release

이 문서에 대한 모든 내용은, 다음의 OS Improvement Code 에 한정한다.

Module	Module version
OS Improvement Code	2.2.0

4.3 Module release notes

4.3.1 Change Log

4.3.1.1 Version 2.2.0.0

➢ 기능 개선

■ 신규 MCU CPULOAD 측정 지원

• 원인	• 6BJ MCU CPULOAD 측정 지원
• 동작 영향	• 없음
• 설정 영향	• 없음
• ASW 조치 필요 사항	• 없음

4.3.1.2 Version 2.1.3.1

➢ 기능 개선

■ 글로벌 사용자를 위한 영문 user manual 제공

• 원인	• 글로벌 사용자를 위한 영문 user manual 제공
• 동작 영향	• 없음
• 설정 영향	• 없음
• ASW 조치 필요 사항	• 없음

4.3.1.3 Version 2.1.3.0

➢ 기능 개선

■ Fault Structure init 시 clear 코드 추가

• 원인	• MPU Enable 이전에 발생한 Fault 로 인한 NMI 발생 방지
• 동작 영향	• 없음
• 설정 영향	• 없음
• ASW 조치 필요 사항	• 없음

4.3.1.4 Version 2.1.2.1

➢ 기능 개선

■ UNECE Cyber Security 법규 대응을 위한 보안 코딩 개선(Report 발행)

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

4.3.1.5 Version 2.1.2.0

➢ 기능 개선

■ RAM ECC 발생 시 reset 방법 변경 (기존: SW Reset -> 변경: WDT Reset)

• 원인	• 기능 안전 테스트 시에 RAM ECC 처리 시 SW reset 수행할 경우, lock-up 발생하는 경우가 있어서 WDT Reset 으로 변경 필요.
• 동작 영향	• 없음
• 설정 영향	• 없음
• ASW 조치 필요 사항	• 없음

4.3.1.6 Version 2.1.1.0

➢ 기능 개선

■ UNECE Cyber Security 법규 대응을 위한 보안 코딩 개선

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

■ Generation 되는 파일 주석의 Input File 목록 정렬

• 원인	• Generation 되는 파일의 내용 변경이 없어도 주석의 input file 순서가 변경되어 파일 변경으로 인식되는 불편사항 존재
• 동작 영향	• 없음
• 설정 영향	• 없음
• ASW 조치 필요 사항	• 없음

4.3.1.7 Version 2.1.0.0

➢ 기능 개선

■ Schema 에 RAM Size 설정 변경 및 신규 MCU 지원

• 원인	• 2BL, 4BF MCU 지원
• 동작 영향	• 없음
• 설정 영향	• 없음
• ASW 조치 필요 사항	• 없음

➢ 기능 개선

■ Schema 에 RAM Size 설정 변경

• 원인	• Schema(PDF 파일)에 RAM Size 최대 값 변경
• 동작 영향	• 없음

• 설정 영향	• 없음
• ASW 조치 필요 사항	• 없음

■ Polyspace (Runtime Error) measurement

Cause	To handle Runtime Error of Polyspace
Operation Impact	N/A
Configuration Impact	N/A
Required measure of ASW	N/A

4.3.1.8 Version 2.0.0.0

➢ 신규 기능

■ 최초 버전

• 원인	• Traveo II 4M 지원
• 동작 영향	• 없음
• 설정 영향	• 없음
• ASW 조치 필요 사항	• 없음

4.3.1.9 Version 1.0.0.0

➢ 신규 기능

■ 최초 버전

• 원인	• OS Improvement Code 기능 요구
• 동작 영향	• 없음
• 설정 영향	• 없음
• ASW 조치 필요 사항	• 없음

4.3.2 Limitation

4.3.2.1 Callback function of pre / post RAM ECC

- Variable is not initialized in Os_PreRamInitCallout(),Os_PostRamInitCallout() function.
- RAM ECC could not be properly processed during Os_PreRamInitCallout() function.

4.3.2.2 FAULT_STRUCT 사용 제한

- OS Improvement Code 에서는 4개의 FAULT_STRUCT 레지스터 중 FAULT_STRUCT1을 사용하므로 다른 사용자는 FAULT_STRUCT1을 사용할 수 없다.

- 다른 FAULT_STRUCT(0, 2, 3)에는 OS Improvement Code에서 처리할 fault source 인 FLASHC_MAIN_NC_ECC, RAMC0_NC_ECC, RAMC1_NC_ECC를 등록해서는 안 된다.

4.3.2.3 Timer 사용 제한

- CPU/IT Load 기능은 “TCPWM0 Group #2, Counter #0”(MCAL Gpt 에서는 TCPWM_0_512)을 사용한다. 따

라서 CPU/IT Load 기능 사용시 해당 Timer는 다른 용도로 사용하지 말아야 한다.

- CPU/IT Load 기능 사용시 TCPWM0 Group #2, Counter #0 이 32bit Free running Timer로 동작 하도록 설정 해 주어야 한다. (예: GptPredefTimerChannelConfiguration 설정)

4.3.3 Deviation

None

5. Configuration Guide

5.1 General

5.1.1 OsImpGeneral

This container contains the configuration parameters to configure general properties of OsImp.
The description of the parameters present in the container is as below

Parameter Name	Value	
OsImpCpuLoad	User Defined	Changeable
OsImpDebugStackdepth	User Defined	Changeable
OsImpOpfUsed	User Defined	Changeable
OsImpErrmUsed	User Defined	Changeable

1) OsImpCpuLoad
   - This parameter is used to select whether the CPU/Interrupt load measurement is used or not.
   - true: Provides CPU/Interrupt load measurement
   - false: Does not provide CPU/Interrupt load measurement
2) OsImpDebugStackdepth
   - This parameter is used to select whether the stack depth measurement is used or not.
   - true: Provides Stack depth measurement
   - false: Does not provide Stack depth measurement
3) OsImpOpfUsed
   - This parameter is used to note whether the OsProfiler is used or not.
   - true: OsProfiler is used
   - false: OsProfiler is not used
4) OsImpErrmUsed
   - This parameter is used to note whether the ErrM module is used or not.
   - true: ErrM is used
   - false: ErrM is not used

5.2 Ram Sector

5.2.1 OsImpRamSector

This container contains the configuration parameters to configure RAM sector of OsImp.
The description of the parameters present in the container is as below

Parameter Name	Value	
OsImpRamDefaultValue	User Defined	Changeable
OsImpRamSectionBaseAddress	User Defined	Changeable
OsImpRamSectionSize	User Defined	Changeable
OsImpRamSectionBaseAddrLinkerSym	User Defined	Changeable
OsImpRamSectionSizeLinkerSym	User Defined	Changeable
OsImpRamInitProperty	User Defined	Changeable

1) OsImpRamDefaultValue
   - This parameter shall represent the Data pre-setting to be initialized.
2) OsImpRamSectionBaseAddress
   - This parameter represents the RAM section base address. The address must be aligned to 32 bytes.
3) OsImpRamSectionSize
   - This parameter represents the RAM section size in bytes. The size must be multiple of 32. If OsImpRamSectionSizeLinkerSym is not empty, then this parameter is not used.
4) OsImpRamSectionBaseAddrLinkerSym
   - This parameter represents the RAM section base address which is defined in linker script. The address must be aligned to 32 bytes.
     If this parameter is empty, then the integer values from OsImpRamSectionBaseAddress will be used.
5) OsImpRamSectionSizeLinkerSym
   - This parameter represents the RAM section size in bytes which is defined in linker script. The size must be multiple of 32.
     If this parameter is empty, then the integer values from OsImpRamSectionSize will be used.
6) OsImpRamInitProperty
   - This parameter selects when RAM needs to be initialized.
   • PowerOnReset: Initialize RAM when power on reset

6. Application Programming Interface (API)

6.1 Type Definitions

6.1.1 Os_ErrorValueType

Name:	Os_ErrorValueType		
Type:	enum		
Range:	_E_OK	0	No error, successful completion
	_E_OS_ACCESS	1	Access to the service/object denied
	_E_OS_CALLEVEL	2	Access to the service from the ISR is not permitted
	_E_OS_ID	3	The object ID is invalid
	_E_OS_LIMIT	4	The limit of services/objects exceeded
	_E_OS_NOFUNC	5	The object is not used, the service is rejected
	_E_OS_RESOURCE	6	The task still occupies the resource
	_E_OS_STATE	7	The state of the object is not correct for the required service
	_E_OS_VALUE	8	A value outside of the admissible limit
	_E_OS_STACKFAULT	9	Stack fault detected via stack monitoring by the OS
	_E_OS_PROTECTION_ARRIVAL	10	Task/Category 2 ISR arrives before its timeframe has expired
	_E_OS_PROTECTION_TIME	11	Task/Category 2 ISR exceeds its execution time budget
	_E_OS_PROTECTION_LOCKED	12	Task/Category 2 ISR exceeds Resource or ISRs lock time
	_E_OS_DISABLEDINT	13	OS service is called inside an interrupt disable/enable pair
	_E_OS_PROTECTION_EXCEPTION	14	Trap occurred
	_E_OS_CORE	15	All functions that are not allowed to operate cross core shall return E_OS_CORE in extended status if called with parameters that require a cross core operation
	_E_OS_INTERFERENCE_DEADLOCK	16	The function GetSpinlock shall return this error if the spinlock referred by the parameter SpinlockID is already occupied by a TASK/ISR2 on the same core.
	_E_OS_NESTING_DEADLOCK	17	A TASK tries to occupy the spinlock while holding a different spinlock in a way that may cause a deadlock.
	_E_OS_SPINLOCK	18	This error means de-scheduling with occupied spinlock
	_E_OS_SERVICEID	19	Service cannot be called
	_E_OS_PARAM_POINTER	20	A pointer argument to an API is null
	_E_OS_PROTECTION_MEMORY	21	Memory access violation occurred

	_E_OS_ILLEGAL_ADDRESS	22	An invalid address is given as a parameter to a service
	_E_OS_SYS_ALARM_INUSE	23	Counter interrupt is nested
	_E_OS_SYS_RAMECC	24	An ECC error has occurred on the RAM
	_E_OS_SYS_DFLASHECC	25	An ECC error has occurred on the Data Flash
	_E_OS_SYS_PFLASHECC	26	An ECC error has occurred on the Program Flash
	_E_OS_MISSINGEND	35	Tasks terminates without a TerminateTask() or ChainTask() call
	_E_OS_SYS_CORE_IS_DOWN	100	This error code means that the core is shutting down state.
	_E_OS_SYS_PANIC	101	This error code means that Inter-core message handling is fault.
	_E_OS_SYS_NMI	102	This error code means that NMI handling is fault.
	_E_OS_SYS_INTERCOREMSG	103	A problem occurred during the inter-core API request process.
Description:	OS Error code redefine for easy to see in debugger		

6.1.2 Os_ErrorApiType

Name:	Os_ErrorApiType		
Type:	enum		
Range:	_OSServiceId_GetApplicationID	0x00	GetApplicationID
	_OSServiceId_GetISRID	0x01	GetISRID
	_OSServiceId_CallTrustedFunction	0x02	CallTrustedFunction
	_OSServiceId_CheckISRMemoryAccess	0x03	CheckISRMemoryAccess
	_OSServiceId_CheckTaskMemoryAccess	0x04	CheckTaskMemoryAccess
	_OSServiceId_CheckObjectAccess	0x05	CheckObjectAccess
	_OSServiceId_CheckObjectOwnership	0x06	CheckObjectOwnership
	_OSServiceId_StartScheduleTableRel	0x07	StartScheduleTableRel
	_OSServiceId_ScheduleTableAbs	0x08	ScheduleTableAbs
	_OSServiceId_StopScheduleTable	0x09	StopScheduleTable
	_OSServiceId_NextScheduleTable	0x0a	NextScheduleTable
	_OSServiceId_StartScheduleTableSynchron	0x0b	StartScheduleTableSynchron
	_OSServiceId_SyncScheduleTable	0x0c	SyncScheduleTable
	_OSServiceId_SetScheduletableAsync	0x0d	SetScheduletableAsync
	_OSServiceId_GetScheduleTableStatus	0x0e	GetScheduleTableStatus
	_OSServiceId_IncrementCounter	0x0f	IncrementCounter
	_OSServiceId_GetCounterValue	0x10	GetCounterValue
	_OSServiceId_GetElapsedValue	0x11	GetElapsedValue
	_OSServiceId_TerminateApplication	0x12	TerminateApplication

	_OSServiceId_AllowAccess	0x13	AllowAccess
	_OSServiceId_GetApplicationState	0x14	GetApplicationState
	_OSServiceId_ActivateTask	0x15	ActivateTask
	_OSServiceId_TerminateTask	0x16	TerminateTask
	_OSServiceId_ChainTask	0x17	ChainTask
	_OSServiceId_Schedule	0x18	Schedule
	_OSServiceId_GetTaskID	0x19	GetTaskID
	_OSServiceId_GetTaskState	0x1a	GetTaskState
	_OSServiceId_EnableAllInterrupts	0x1b	EnableAllInterrupts
	_OSServiceId_DisableAllInterrupts	0x1c	DisableAllInterrupts
	_OSServiceId_ResumeAllInterrupts	0x1d	ResumeAllInterrupts
	_OSServiceId_SuspendAllInterrupts	0x1e	SuspendAllInterrupts
	_OSServiceId_ResumeOSInterrupts	0x1f	ResumeOSInterrupts
	_OSServiceId_SuspendOSInterrupts	0x20	SuspendOSInterrupts
	_OSServiceId_GetResource	0x21	GetResource
	_OSServiceId_ReleaseResource	0x22	ReleaseResource
	_OSServiceId_SetEvent	0x23	SetEvent
	_OSServiceId_ClearEvent	0x24	ClearEvent
	_OSServiceId_GetEvent	0x25	GetEvent
	_OSServiceId_WaitEvent	0x26	WaitEvent
	_OSServiceId_GetAlarmBase	0x27	GetAlarmBase
	_OSServiceId_GetAlarm	0x28	GetAlarm
	_OSServiceId_SetRelAlarm	0x29	SetRelAlarm
	_OSServiceId_SetAbsAlarm	0x2a	SetAbsAlarm
	_OSServiceId_CancelAlarm	0x2b	CancelAlarm
	_OSServiceId_GetActiveApplicationMode	0x2c	GetActiveApplicationMode
	_OSServiceId_StartOS	0x2d	StartOS
	_OSServiceId_ShutdownOS	0x2e	ShutdownOS
	_OSServiceId_GetCoreID	0x2f	GetCoreID
	_OSServiceId_GetSpinlock	0x30	GetSpinlock
	_OSServiceId_ReleaseSpinlock	0x31	ReleaseSpinlock
	_OSServiceId_TryToGetSpinlock	0x32	TryToGetSpinlock
	_OSServiceId_ShutdownAllCores	0x38	ShutdownAllCores
	_OSServiceId_StartCore	0x3c	StartCore
Description	OS Service id redefine for easy to see in debugger		

6.1.3 Os_ParamBlockType1

Name:	Os_ParamBlockType1		
Type:	union		
Range:	AlarmType	OsAlarmId	This parameter gives Id of the configured alarm
	ApplicationType	OsApplicationId	This parameter defines the application id.
	CounterType	OsCounterId	This parameter gives Id of the configured counter
	ResourceType	OsResourceId	This parameter gives Id of the configured resource
	TaskType	OsTaskId	This parameter gives Id of the configured task
	ScheduleTableType	OsScheduleTableId	This parameter gives Id of the configured schedule table
	ScheduleTableType	OsScheduleTableId_From	This parameter gives Id of the configured schedule table
	TrustedFunctionIndexType	OsTrustedFunctionIndexId	This parameter identifies a trusted function
	EventMaskType	OsMask	This parameter identifies the mask of an event
	SpinlockIdType	OsSpinlockId	This parameter gives Id of the configured spinlock
Description:	This union is defined for first parameter of OS API		

6.1.4 Os_ParamBlockType2

Name:	Os_ParamBlockType2		
Type:	union		
Range:	ScheduleTableType	OsScheduleTableId_To	This parameter gives Id of the configured schedule table
	TickType	OsValue	This parameter holds the current value of the synchronization counter
	void *	OsTrustedFunctionParams	This parameter is a pointer to parameters for a trusted function
	RestartType	OsRestartOption	This parameter defines the use of a Restart Task after terminating an OS-Application.
	EventMaskType	OsMaskParam2	This parameter identifies the current status of the event mask of task
	TickType	OsIncrement	This parameter holds a relative start value in alarm
	TickType	OsOffset	This parameter holds offset value in schedule table
	TickType	OsStart	This parameter holds an absolute start value in alarm
Description:	This union is defined for first parameter of OS API		

6.1.5 Os_ParamBlockType3

Name:	Os_ParamBlockType3		
Type:	union		
Range:	TickType	OsCycle	This parameter holds the cycle value in case of cyclic alarm
Description:	This union is defined for first parameter of OS API		

6.1.6 Os_ErrorType

Name:	Os_ErrorType		
Type:	structure		
Range:	Os_ErrorApiType	enApi	OS API name
	Os_ErrorValueType	enErrorNo	Error reason
	Os_ParamBlockType1	unPar1	OS API first parameter
	Os_ParamBlockType2	unPar2	OS API second parameter
	Os_ParamBlockType3	unPar3	OS API third parameter
Description:	This structure is defined for OS error information type		

6.1.7 Os_LoadType

Name:	Os_LoadType		
Type:	structure		
Range:	usCPULoad	uint16 (0 ~ 1000)	Current CPU Load value
	usCPULoadPeak	uint16 (0 ~ 1000)	CPU Load Peak value after value reset
	usCPULoadMean	uint16 (0 ~ 1000)	CPU Load Mean value
	usITLoad	uint16 (0 ~ 1000)	Current Interrupt Load value
	usITLoadPeak	uint16 (0 ~ 1000)	Interrupt Load Peak value after value reset
	usITLoadMean	uint16 (0 ~ 1000)	Interrupt Load Mean value
Description:	This structure holds the CPU and Interrupt Load value		

6.2 Macro Constants

None

6.3 Functions

6.3.1 AppCallbackOnSystemError

Function Name	AppCallbackOnSystemError
Syntax	FUNC(void, OS_CALLOUT_CODE) AppCallbackOnSystemError(StatusType ErrorId)
Service ID	N/A
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (In)	ErrorId - 실제 값과 의미는 6.1.1 Os_ErrorValueType 참조
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	이 Callback 함수는 ECC error 또는 기타 OS 의 error 가 발생했을 때 호출된다. 사용자는 여기에서 ECC 가 발생했을 경우의 추가적인 처리를 할 수 있다.
Preconditions	[주의] RAM ECC error 가 발생할 경우 이 Callback 함수가 호출되기 전 모든 RAM 이 '0'으로 초기화된다. 따라서 이 함수가 호출되었을 때는 모든 전역변수가 지워진 상태이다.
Configuration Dependency	None

6.3.2 Os_UpdateOsErrorInfo

Function Name	Os_UpdateOsErrorInfo
Syntax	FUNC(void, OS_CODE) Os_UpdateOsErrorInfo(StatusType LddError)
Service ID	N/A
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (In)	LddError – OS Error Id
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	OS 의 Hook 들로부터 error 정보를 수집해 저장하고 사용자가 확인할 수 있도록 한다. - E_OS_LIMIT, E_OS_STACKFAULT 발생 횟수 - 전체 OS error 발생 횟수 - 최근 8 회동안 발생한 OS error 종류, 관련 API 및 parameter
Preconditions	None
Configuration Dependency	None

6.3.3 Os_UserGetCPULoad

Function Name	Os_UserGetCPULoad
Syntax	FUNC(void, OS_CODE) Os_UserGetCPULoad(P2VAR(Os_LoadType, AUTOMATIC, OS_VAR) LpLoad, boolean restart)
Service ID	N/A
Sync/Async	Synchronous
Reentrancy	Non Reentrant

Parameters (In)	restart – TRUE: Get load and restart measurement FALSE: Just get load.
Parameters (Inout)	None
Parameters (Out)	LpLoad - Os_LoadType pointer for save CPU and IT load.
Return Value	None
Description	This service is used to get CPU and Interrupt Load.
Preconditions	None
Configuration Dependency	None

example) This example below is not applicable to the project because it is a simple reference.

```
TASK(Task_SWP_FG1_100ms)
{
        Os_LoadType LddLoad = {0u, 0u, 0u, 0u, 0u, 0u};
        Os_UserGetCPULoad(&LddLoad, OS_TRUE);
        .. .. .. ..
}

TASK(Task_SWP_FG1_1s)
{
        Os_LoadType LddLoad = {0u, 0u, 0u, 0u, 0u, 0u};
        Os_UserGetCPULoad(&LddLoad, OS_FALSE);
        .. .. .. ..
}
```

6.3.4 Os_ClearCPULoadPeak

Function Name	Os_ClearCPULoadPeak
Syntax	FUNC(void, OS_CODE) Os_ClearCPULoadPeak(void)
Service ID	N/A
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	Clear CPU Load Peak value in current core.
Preconditions	None
Configuration Dependency	None

example) This example below is not applicable to the project because it is a simple reference.

```
TASK(Task_SWP_FG1_100ms)
{
        Os_ClearCPULoadPeak();
        .. .. .. ..
}
```

6.3.5 Os_ClearITLoadPeak

Function Name	Os_ClearITLoadPeak
Syntax	FUNC(void, OS_CODE) Os_ClearITLoadPeak(void)
Service ID	N/A
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	Clear Interrupt Load Peak value in current core.
Preconditions	None
Configuration Dependency	None

example) This example below is not applicable to the project because it is a simple reference.

```
TASK(Task_SWP_FG1_100ms)
{
        Os_ClearITLoadPeak();
        .. .. .. ..
}
```

6.3.6 Os_RestartMeanLoad

Function Name	Os_RestartMeanLoad
Syntax	FUNC(void, OS_CODE) Os_RestartMeanLoad(void)
Service ID	N/A
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	This service is used to restart the measure of the mean of CPU/IT load.
Preconditions	None
Configuration Dependency	None

example) This example below is not applicable to the project because it is a simple reference.

```
TASK(Task_SWP_FG1_100ms)
{
        Os_RestartMeanLoad();
        .. .. .. ..
}
```

6.3.7 Os_GetMaxStackUsage

Function Name	Os_GetMaxStackUsage
Syntax	FUNC(StatusType, OS_CODE) Os_GetMaxStackUsage(TaskType LddTaskId, uint32* pStackUsage, uint32* pStackSize)
Service ID	N/A
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (In)	LddTaskId – Task ID
Parameters (Inout)	None
Parameters (Out)	pStackUsage - Stack usage of the stack which is used by Task pStackSize - Total stack size of the stack which is used by Task
Return Value	StatusType - E_OK: no error - E_OS_STATE: error occurs during execution
Description	This service is used to get max stack usage of the stack which is used by the target Task.
Preconditions	None
Configuration Dependency	None

example) This example below is not applicable to the project because it is a simple reference.

```
TASK(Task_SWP_FG1_100ms)
{
        StatusType statusReturn;
        uint32 stackUsage;
        uint32 stackSize;
        statusReturn = Os_GetMaxStackUsage(Task_SWP_FG1_100ms, &stackUsage, &stackSize);
        .. .. .. ..
}
```

6.3.8 Os_CallBackNMInterrupt

Function Name	Os_CallBackNMInterrupt
Syntax	FUNC(void, OS_CALLOUT_CODE) Os_CallBackNMInterrupt(void)
Service ID	N/A
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	NMI 가 발생하면 호출되는 callout 함수이다. NMI 발생시 필요한 처리를 할 수 있다.
Preconditions	None
Configuration Dependency	None

6.3.9 Os_PreRamInitCallout

Function Name	Os_PreRamInitCallout
Syntax	FUNC(void, OS_CALLOUT_CODE) Os_PreRamInitCallout(void)
Service ID	N/A
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	Startup 코드에서 RAM ECC 초기화 전에 호출되는 callout 함수이다.
Preconditions	None
Configuration Dependency	None

6.3.10 Os_PostRamInitCallout

Function Name	Os_PostRamInitCallout
Syntax	FUNC(void, OS_CALLOUT_CODE) Os_PostRamInitCallout (void)
Service ID	N/A
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (In)	None
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None
Description	Startup 코드에서 RAM ECC 초기화 후에 호출되는 callout 함수이다.
Preconditions	None
Configuration Dependency	None

6.4 Global Variables

Name:	Type:	*Description:*
GulOsErrorCount	uint32	OS error 발생 횟수
GulOsErrorLastPosition	uint32	GucOsError 의 마지막 error 정보 위치 index
Os_GulOsLimitError	uint32	E_OS_LIMIT 발생 횟수
Os_GulOsStackFaultError	uint32	E_OS_STACKFAULT 발생 횟수
GucOsError	Os_ErrorType []	최근 8 회동안 발생한 OS error 종류, 관련 API 및 parameter 를 저장

7. Generator

7.1 Generator Option

Options	Description
-H/-Help	To display help regarding usage of the tool.
-O/-Output	To generate the output files in the specified directory location.
-V/-Version	To display the copyright information and the tool version.
-L/-Log	To generate "Os_Imp_Cfg.log" file.
-D/-DryRun	To execute in validation mode.
-I/-Info	To disable an Information Message(s).
-W/-Warn	To disable Warning Message(s).
-DDT	Not to generate the time stamp in the generated files.

7.2 Generator Error Message

This section helps to analyze the errors or warnings displayed during the execution of the tool. It ensures conformance of input file(s) with syntax and semantics.

The Generation Tool displays errors or warnings or information when the user has configured incorrect inputs. The format of Error/Warning/Information message is as shown below:

- ERR/WRN/INF<mid><xxx>: < Error/Warning/Information Message>

  Where,
  <mid>: 255 – OsImp Module Id (255) for user configuration checks.
  000 – for command line checks.
  <xxx>: 001 – 999 – Message ID.
- File Name : Name of the file in which the error has occurred
- Path : Absolute path of the container in which the parameter is present

'File Name' and 'Path' are optional.

Below section provides the list of error, warning and information messages.

7.2.1 Error Messages

ERR255001: <OsImpRamDefaultValue> shall not be empty

Ram Sector 에서 'Default Value' 를 설정하지 않으면 발생한다. 해당 항목은 반드시 설정해야 한다.

ERR255002: At least one of <OsImpRamSectionBaseAddress> and <OsImpRamSectionBaseAddrLinkerSym> should be set.

Ram Sector 에서 'Ram Section Base Address' 와 'Ram Section Base Addr Linker Sym' 을 모두 설정하지 않았을 때 발생한다. 둘 중 하나는 반드시 설정하여야 한다.

ERR255003: At least one of <OsImpRamSectionSize> and <OsImpRamSectionSizeLinkerSym> should be set.

Ram Sector 에서 'Ram Section Size' 와 'Ram Section Size Linker Sym' 을 모두 설정하지 않았을 때 발생한다. 둘 중 하나는 반드시 설정하여야 한다.

ERR255004: <OsImpRamInitProperty> shall not be empty

Ram Sector 에서 'Ram Init Property' 를 설정하지 않으면 발생한다. 해당 항목은 반드시 설정해야 한다.

ERR255005: <Parameter Name> shall be aligned <32>

<Parameter Name>이 32 의 배수로 설정되지 않으면 발생한다. 해당 항목은 기본적으로 32byte 로 align 되어야 한다.

Parameter Name
OsImpRamSectionBaseAddress
OsImpRamSectionSize

7.2.2 Warning Messages

None

7.2.3 Information Messages

None

8. Appendix

8.1 설계 시 유의사항

8.1.1 Callout function of pre / post RAM ECC

RAM ECC 초기화 수행 이전 및 완료 이후에 대해서 각각 Os_PreRamInitCallout() / Os_PostRamInitCallout() 함수를 통해서 사용자 처리를 추가 할 수 있다. Os_PreRamInitCallout() / Os_PostRamInitCallout() 함수에서 변수 초기화는 완료되지 않은 상태이기 때문에 변수 사용에 주의 해야 한다.
Traveo II MCU 에서 RAM ECC 탐지와 처리는 Os_PreRamInitCallout() 함수에서는 동작하지 않는다.
Os_PreRamInitCallout() / Os_PostRamInitCallout() 함수에 대한 정의는 아래와 같이 reference code 인 App_OsHook.c 파일에 정의 되어 있으므로 사용자가 적절히 변형하여 사용 가능 하다.

```
FUNC(void, OS_CALLOUT_CODE) Os_PreRamInitCallout(void)
{
}

FUNC(void, OS_CALLOUT_CODE) Os_PostRamInitCallout(void)
{
}
```

8.1.2 ECC 처리를 위한 NMI callback 구현

Os 에서는 NMI 가 발생할 경우 Os_CallBackNMInterrupt() 를 호출한다. ECC 처리를 위해서는 이 callback 함수에 필요한 코드를 구현해야 한다.
RAM ECC 가 발생할 경우는 아래 함수를 순차적으로 호출해 주어야 한다.
Os_ResetMCU() 에서 최종적으로 WDT Reset 이 발생한다.

```
Os_InitRamSector32(OS_RAM_SECTOR_POR_COUNT, Os_GaaRamSectorInitPowerOnReset);
AppCallbackOnSystemError(E_OS_SYS_RAMECC);
Os_ResetMCU();
```

PFLASH ECC 가 발생할 경우는 아래 함수를 호출해 주어야 한다.

```
ShutdownOS(E_OS_SYS_PFLASHECC);
```

실제 ECC 처리에 필요한 코드가 구현된 Os_CallBackNMInterrupt() 샘플이 Reference_Code/App_NMI.c 파일에 포함되어 배포되므로 참고하도록 한다.

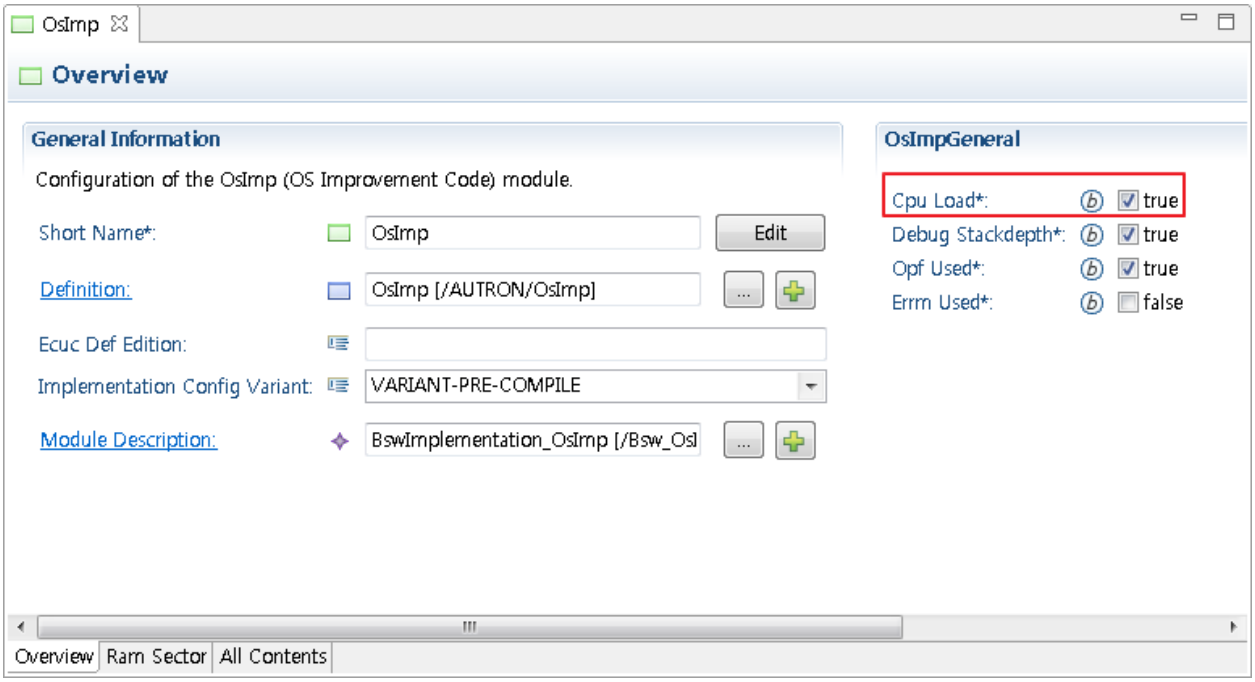
8.2 Exclusive Areas

None

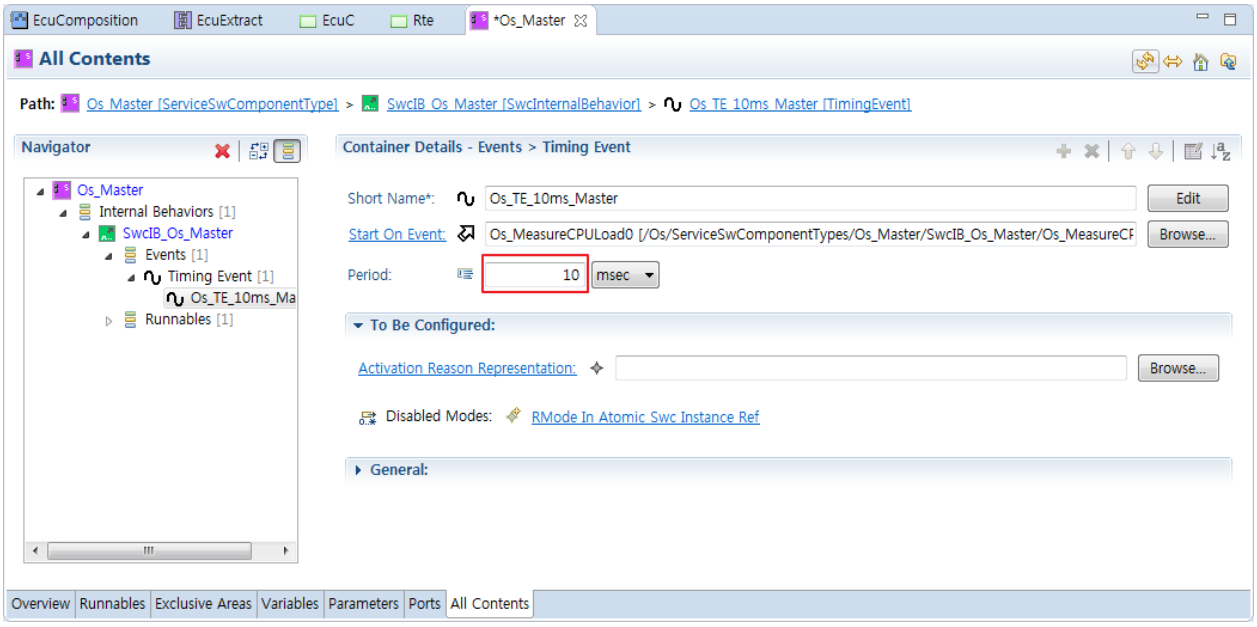
8.3 Debugging Features

8.3.1 CPU & IT Load configuration

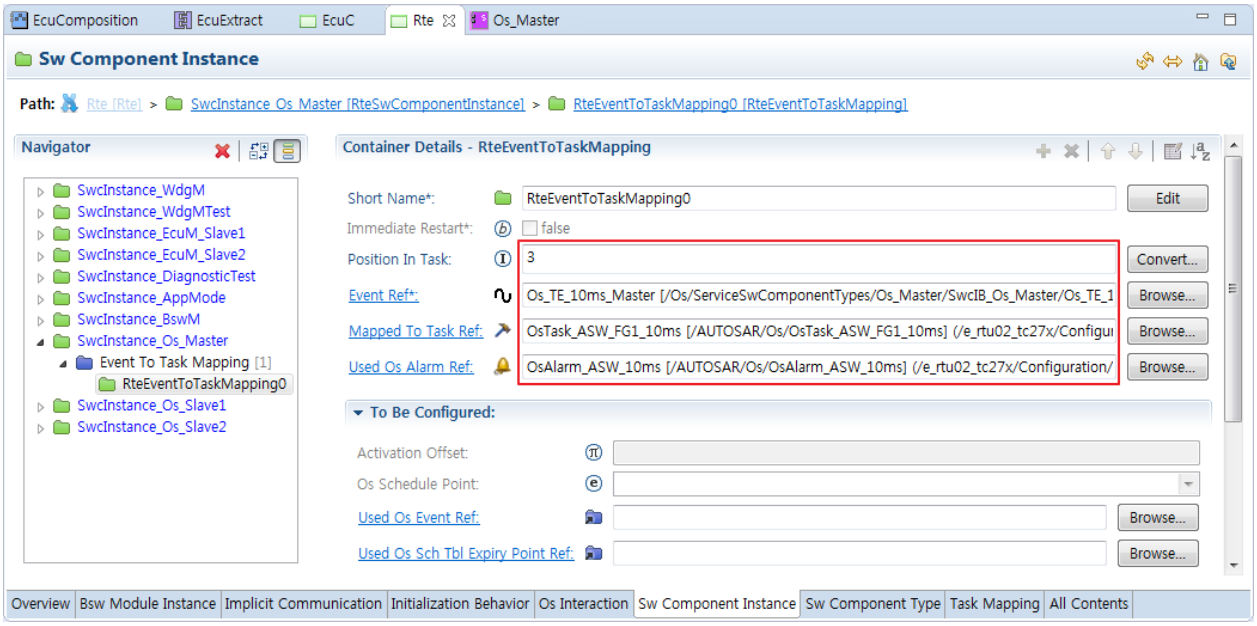
1. Enable CPU/IT Load functionality in the OsImp/OsImpGeneral configuration window.



2. Set measurement period in the Os/ServiceSwComponentTypes/Os_Master/Internal Behaviors/SwcIB_Os_Master/Events/Timing Event/Os_TE_10ms_Master of Swcd_Bsw_Os.arxml



3. Set Alarm and Task mapping of Timing Event in the Rte/SwcInstance_Os_Master/RteEventToTaskMapping0 configuration window.
Note1: Alarm and Task should belong to trusted OS-Application.
Note2: 'Position In Task' should not overlap with other SWC RteEvent.



4. In case of multicore environment, configurations for slave cores are needed.
   If the ‘OsNumberOfCores’ is 2, RteEvent of Os_Slave1 should be configured.
   And if ‘OsNumberOfCores’ is 3, RteEvent of Os_Slave1 and Os_Slave2 should be configured as shown above 2, 3.