


SCOPE OF APPLICATION All Project/Engineering		SHT/SHTS 1 / 21
Responsibility: 클래식오토사팀	AUTOSAR OS Improvement Code User Manual	DOC. NO
AUTOSAR OS Improvement Code User Manual		

Document Change History				
Date (YYYY-MM-DD)	Ver.	Editor	Chap	내용(개정 전 -> 개정 후)
2023-01-11	1.0.0.0	HG.Kim	All	• Initial Creation
2023-04-07	1.1.0.0	HG.Kim	4.3.1	• Ecc error detection/Handling 내용 추가

Edition Date: 2023-01-11	File Name Os_Imp_UM.pdf	Creation HG Kim 2023-01-11	Check JH Cho 2023-01-11	Approval JH Jung 2023-01-11
Document Management System				

## Table of Contents

<b>1. OVERVIEW .....</b>	<b>4</b>
<b>2. REFERENCE .....</b>	<b>4</b>
<b>3. AUTOSAR SYSTEM .....</b>	<b>5</b>
<b>3.1 Overview of Software Layers .....</b>	<b>5</b>
<b>3.2 OS Improvement Code .....</b>	<b>6</b>
3.2.1 CPU Load.....	6
3.2.2 Stackdepth (Not supported).....	6
3.2.3 Memory Ecc error 처리 .....	7
<b>4. PRODUCT RELEASE NOTES .....</b>	<b>7</b>
<b>4.1 Overview .....</b>	<b>7</b>
<b>4.2 Scope of the release .....</b>	<b>7</b>
<b>4.3 Module release notes .....</b>	<b>7</b>
4.3.1 Change Log.....	7
4.3.1.1 Version 1.1.0.0.....	7
4.3.1.2 Version 1.0.0.0.....	8
4.3.2 Limitation.....	8
4.3.2.1 Timer 사용 제한 .....	8
4.3.3 Deviation .....	8
<b>5. CONFIGURATION GUIDE .....</b>	<b>9</b>
<b>5.1 General .....</b>	<b>9</b>
5.1.1 OslmpGeneral .....	9
<b>6. APPLICATION PROGRAMMING INTERFACE (API).....</b>	<b>10</b>
<b>6.1 Type Definitions .....</b>	<b>10</b>
6.1.1 Os_ErrorValueType .....	10
6.1.2 Os_ErrorApiType .....	11
6.1.3 Os_ParamBlockType1 .....	12
6.1.4 Os_ParamBlockType2 .....	13
6.1.5 Os_ParamBlockType3 .....	13
6.1.6 Os_ErrorType .....	14
6.1.7 Os_LoadType .....	14
<b>6.2 Macro Constants .....</b>	<b>14</b>
<b>6.3 Functions .....</b>	<b>14</b>
6.3.1 AppCallbackOnSystemError .....	14
6.3.2 Os_UpdateOsErrorInfo .....	15
6.3.3 Os_UserGetCPULoad .....	15

6.3.4	Os_ClearCPULoadPeak.....	16
6.3.5	Os_ClearITLoadPeak.....	17
6.3.6	Os_RestartMeanLoad .....	17
6.3.7	Os_GetMaxStackUsage (NOT Supported) .....	17
<b>6.4</b>	<b>Global Variables.....</b>	<b>18</b>
<b>7.</b>	<b>GENERATOR.....</b>	<b>19</b>
<b>7.1</b>	<b>Generator Option.....</b>	<b>19</b>
<b>7.2</b>	<b>Generator Error Message.....</b>	<b>19</b>
7.2.1	Error Messages .....	19
7.2.2	Warning Messages.....	19
7.2.3	Information Messages .....	19
<b>8.</b>	<b>APPENDIX .....</b>	<b>20</b>
<b>8.1</b>	<b>설계 시 유의사항 .....</b>	<b>20</b>
<b>8.2</b>	<b>Exclusive Areas .....</b>	<b>20</b>
<b>8.3</b>	<b>Debugging Features .....</b>	<b>20</b>
8.3.1	CPU & IT Load configuration .....	20

## 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 사용시 보다 자세한 기능적인 설명이 필요한 경우 Reference 문서를 참조 해야 한다.

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

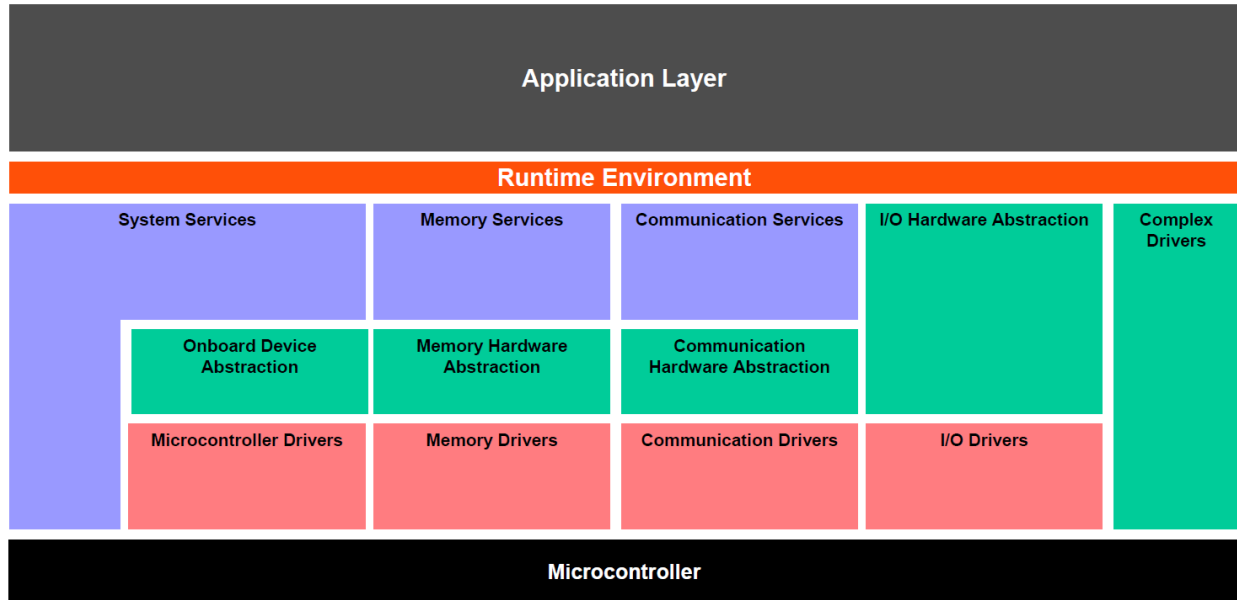
## 2. Reference

Sl. No.	Title	Version
1.	AUTOSAR_SRS_OS.pdf	4.4.0
2.	AUTOSAR_SWS_OS.pdf	4.4.0
3.	Os_Imp_UM.pdf	1.0.0.0

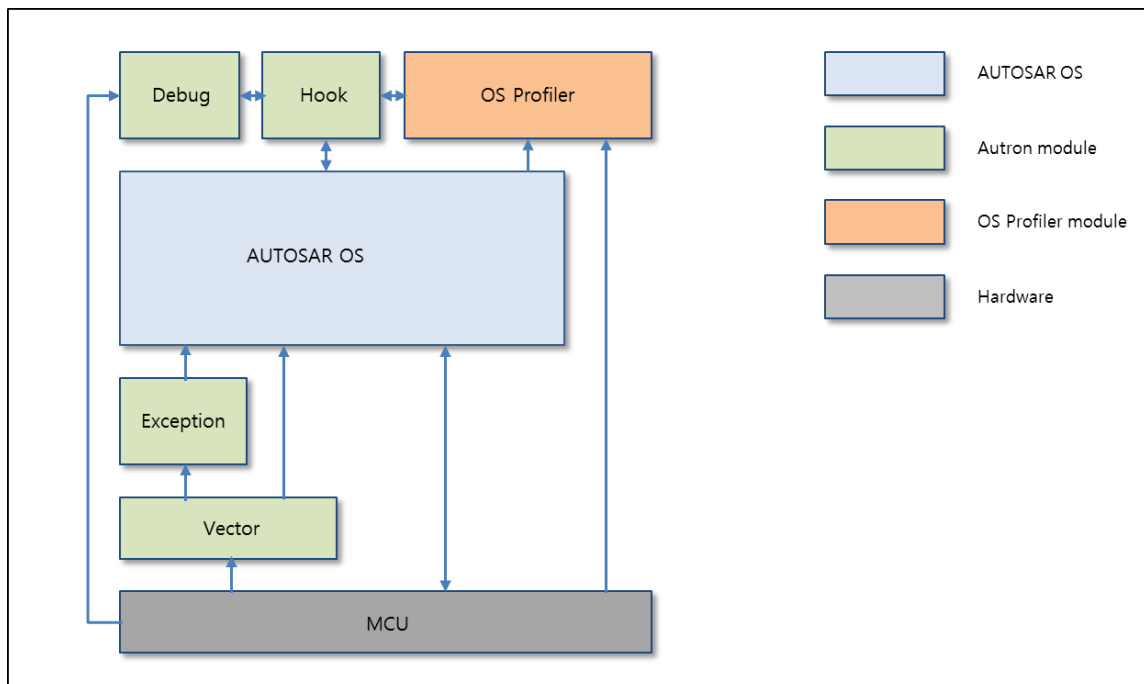
### 3. AUTOSAR System

#### 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 의 Autron 모듈이 추가되어 전체적인 AUTORSAR OS 모듈을 완성한다. 이들 Autron 모듈은 OS Improvement Code 에 포함된다.



## 3.2 OS Improvement Code

### 3.2.1 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: 100 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 "STM0" is used.

### 3.2.2 Stackdepth (Not supported)

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.

### 3.2.3 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 않는다.

[Correction 불가능한 ECC error]

Memory Type		지원 방법		기타 사항
		Handling	Limitation	
RAM		Os_NMIHandler 가 Os_CallbackNMInterrupt 호출	-	사용자는 해당 함수를 이용하여 ECC error 에 대한 Reaction 을 설정할 수 있음 (SW Reset, HW Reset)
ROM	PFLASH	Os_NMIHandler 가 Os_CallbackNMInterrupt 호출	-	사용자는 해당 함수를 이용하여 ECC error 에 대한 Reaction 을 설정할 수 있음 (SW Reset, HW Reset)
	DFLASH	없음 (MCAL 에서 모두 처리)	-	-

**Note:** Reference Code(App\_NMI.c)는 Os\_CallbackNMInterrupt 호출 시, Destructive Reset 발생시키도록 구현되어 있다.

## 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	1.1.0

### 4.3 Module release notes

#### 4.3.1 Change Log

##### 4.3.1.1 Version 1.1.0.0

- 신규 기능
  - Ecc Error detection/Handling 지원

• 원인	• Ecc Error detection/Handling 기능 요구
• 동작 영향	• 없음
• 설정 영향	• 없음
• ASW 조치 필요 사항	• 없음

#### 4.3.1.2 Version 1.0.0.0

##### ➤ 신규 기능

##### ■ CPU Load 지원

• 원인	• CPU Load 기능 요구
• 동작 영향	• 없음
• 설정 영향	• 없음
• ASW 조치 필요 사항	• 없음

#### 4.3.2 Limitation

##### 4.3.2.1 Timer 사용 제한

- CPU/IT Load 기능은 “STM0” 을 사용한다. 따라서 CPU/IT Load 기능 사용시 해당 Timer 는 다른 용도로 사용하지 말아야 한다.

#### 4.3.3 Deviation

None



## 5. Configuration Guide

### 5.1 General

#### 5.1.1 OslmpGeneral

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

Parameter Name	Value	
OslmpCpuLoad	User Defined	Changeable
OslmpDebugStackdepth	User Defined	NOT Supported

##### 1) OslmpCpuLoad

- 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) OslmpDebugStackdepth

- 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

## 6. Application Programming Interface (API)

### 6.1 Type Definitions

#### 6.1.1 Os\_ErrorValueType

<b>Name:</b>	Os_ErrorValueType		
<b>Type:</b>	enum		
<b>Range:</b>	_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.
<b>Description:</b>	OS Error code redefine for easy to see in debugger		

## 6.1.2 Os\_ErrorApiType

<b>Name:</b>	Os_ErrorApiType		
<b>Type:</b>	enum		
<b>Range:</b>	_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
<b>Description</b>	OS Service id redefine for easy to see in debugger		

<b>Name:</b>	Os_ParamBlockType1		
<b>Type:</b>	union		
<b>Range:</b>	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
<b>Description:</b>	This union is defined for first parameter of OS API		

#### 6.1.4 Os\_ParamBlockType2

<b>Name:</b>	Os_ParamBlockType2		
<b>Type:</b>	union		
<b>Range:</b>	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
<b>Description:</b>	This union is defined for first parameter of OS API		

#### 6.1.5 Os\_ParamBlockType3

<b>Name:</b>	Os_ParamBlockType3
--------------	--------------------

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

### 6.1.6 Os\_ErrorType

<b>Name:</b>	Os_ErrorType		
<b>Type:</b>	structure		
<b>Range:</b>	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
<b>Description:</b>	This structure is defined for OS error information type		

### 6.1.7 Os\_LoadType

<b>Name:</b>	Os_LoadType		
<b>Type:</b>	structure		
<b>Range:</b>	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
<b>Description:</b>	This structure holds the CPU and Interrupt Load value		

## 6.2 Macro Constants

None

## 6.3 Functions

### 6.3.1 AppCallbackOnSystemError

<b>Function Name</b>	AppCallbackOnSystemError
<b>Syntax</b>	FUNC(void, OS_CALLOUT_CODE) AppCallbackOnSystemError(StatusType ErrorId)
<b>Service ID</b>	N/A
<b>Sync/Async</b>	Synchronous
<b>Reentrancy</b>	Non Reentrant
<b>Parameters (In)</b>	ErrorId - 실제 값과 의미는 6.1.1 Os_ErrorValueType 참조
<b>Parameters (Inout)</b>	None
<b>Parameters (Out)</b>	None
<b>Return Value</b>	None
<b>Description</b>	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.
<b>Preconditions</b>	[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.
<b>Configuration Dependency</b>	None

### 6.3.2 Os\_UpdateOsErrorInfo

<b>Function Name</b>	Os_UpdateOsErrorInfo
<b>Syntax</b>	FUNC(void, OS_CODE) Os_UpdateOsErrorInfo(StatusType LddError)
<b>Service ID</b>	N/A
<b>Sync/Async</b>	Synchronous
<b>Reentrancy</b>	Non Reentrant
<b>Parameters (In)</b>	LddError – OS Error Id
<b>Parameters (Inout)</b>	None
<b>Parameters (Out)</b>	None
<b>Return Value</b>	None
<b>Description</b>	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
<b>Preconditions</b>	None
<b>Configuration Dependency</b>	None

### 6.3.3 Os\_UserGetCPULoad

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

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

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

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

<b>Function Name</b>	Os_RestartMeanLoad
<b>Syntax</b>	FUNC(void, OS_CODE) Os_RestartMeanLoad(void)
<b>Service ID</b>	N/A
<b>Sync/Async</b>	Synchronous
<b>Reentrancy</b>	Non Reentrant
<b>Parameters (In)</b>	None
<b>Parameters (Inout)</b>	None
<b>Parameters (Out)</b>	None
<b>Return Value</b>	None
<b>Description</b>	This service is used to restart the measure of the mean of CPU/IT load.
<b>Preconditions</b>	None
<b>Configuration Dependency</b>	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 (NOT Supported)

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

```

## 6.4 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

None

#### 7.2.2 Warning Messages

None

#### 7.2.3 Information Messages

None

## 8. Appendix

### 8.1 설계 시 유의사항

None

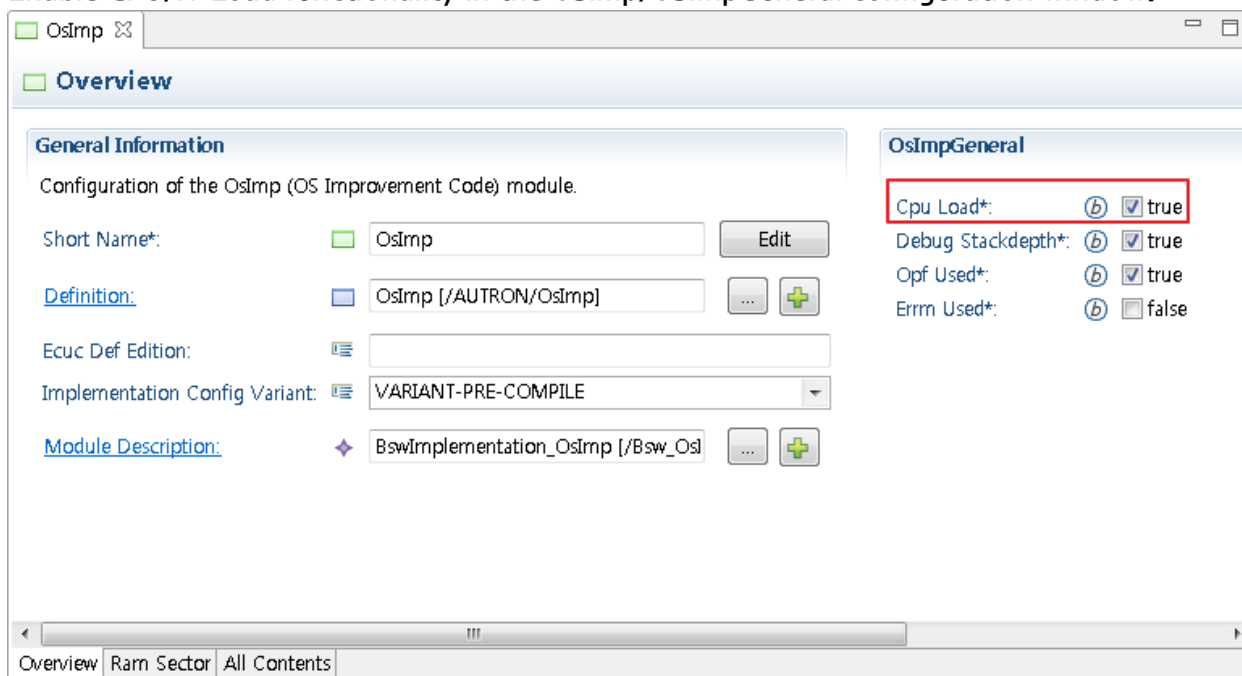
### 8.2 Exclusive Areas

None

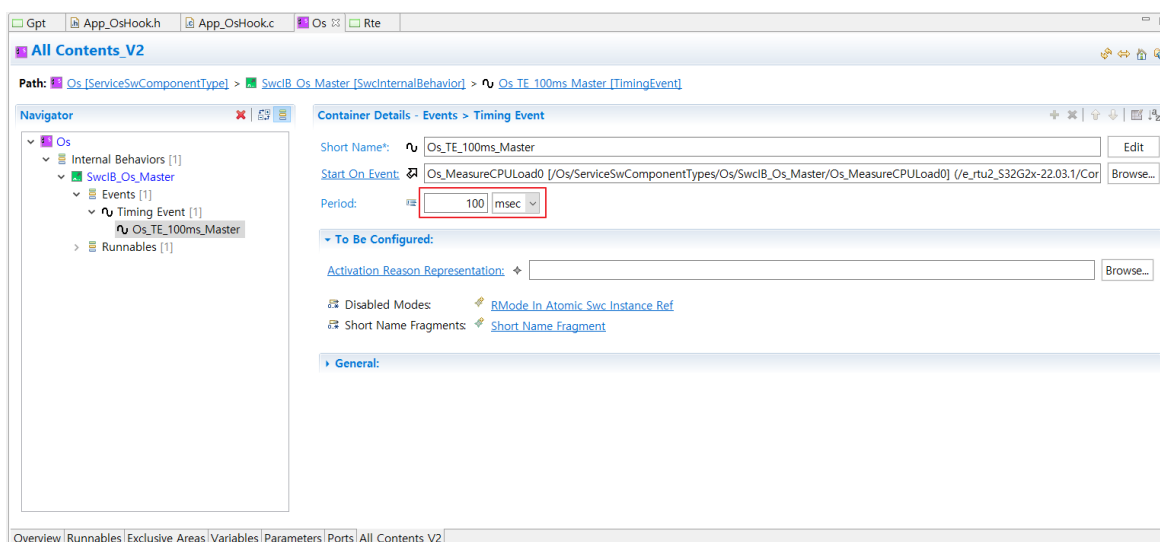
### 8.3 Debugging Features

#### 8.3.1 CPU & IT Load configuration

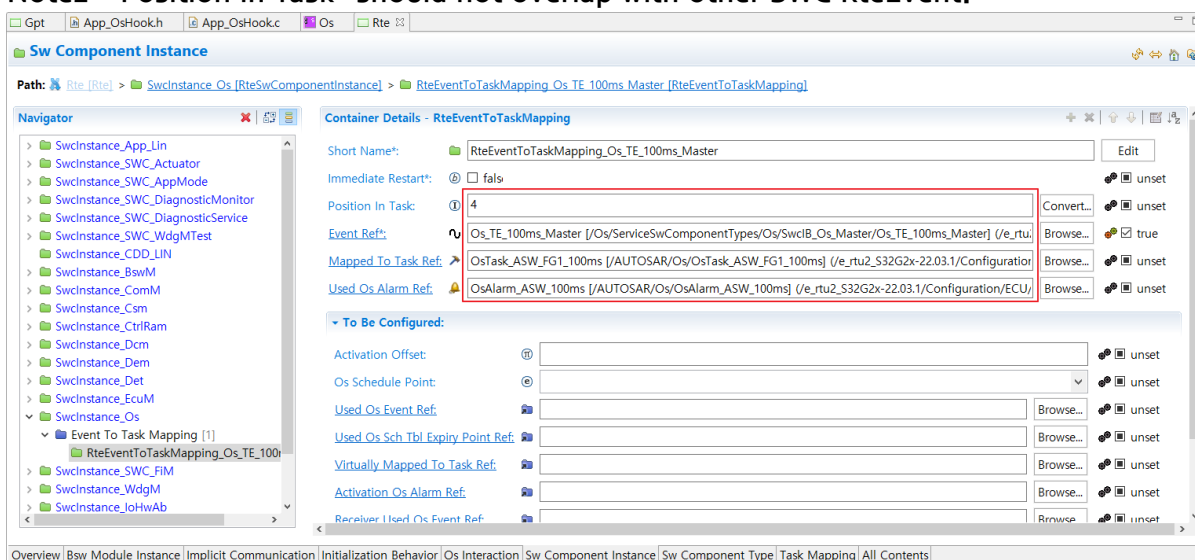
1. Enable CPU/IT Load functionality in the OsImp/OsImpGeneral configuration window.



2. Set measurement period in the Os/ServiceSwComponentTypes/Os\_Master/Internal Behaviors/SwclB\_Os\_Master/Events/Timing Event/Os\_TE\_100ms\_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.