HYUNDAI AUTOEVER

AUTOSAR OS Improvement Code User Manual

DOC. NO

SCOPE OF APPLICATION All Project/Engineering
Responsibility : Classic AUTOSAR 1 Team

File Name Os_Imp_UM-en.pdf
Creation HG Kim 2023/09/08
Check JH Cho 2023/09/08
Approval JH Jung 2023/09/08
Edition Date: 2023/09/08
Document Management System

HYUNDAI AutoEver

Document Change History				
Date (YYYY-MM-DD)	Ver.	Editor	Chap	Description (before -> after revision)
2020-03-10	1.0.0.0	MJ.Woo	All	• Initial Creation
2020-07-03	2.0.0.0	JH.Cho	4	•Updated Change Log
2020-08-14	2.0.1.1	JH.Cho	4	• Updated Change Log
2020-12-22	2.1.0.0	JH.Cho	4	• Updated Change Log
2022-03-10	2.1.1.0	MJ.Woo	All 4	• Changed company name • Updated Change Log
2022-05-12	2.1.2.0	YH.Han	4 8.1.2	• Updated Change Log • Added WDT reset in case of ECC RAM
2022-08-10	2.1.2.1	JC.Kim	1 4	• Updated Copyright comment • Updated 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 provides additional functionalities required for the integration between AUTOSAR OS and MCUs. It provides functionalities that are not defined in the specification documents of the AUTOSAR OS but necessary to run the AUTOSAR OS on MCUs.
See relevant reference documents for more details on the functionalities of OS Improvement Code.

Each configuration category is defined as follows.

- Changeable (C): Items that can be configured by users
- Fixed (F): Items that cannot be changed by users
- NotSupported (N): Unavailable items

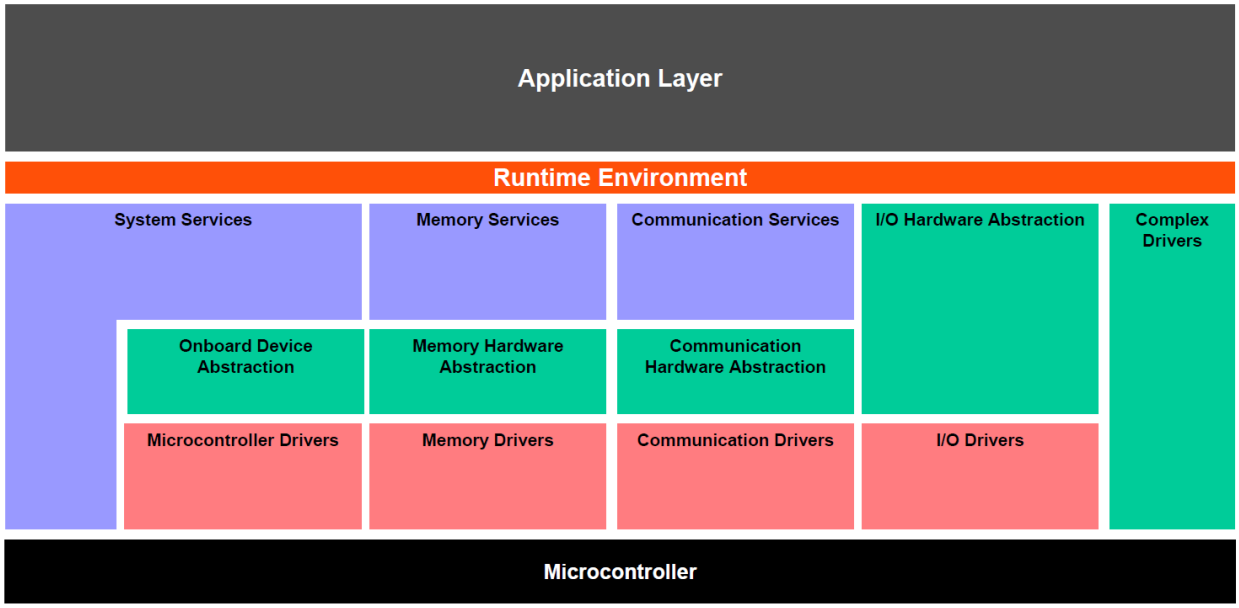
2. Reference

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

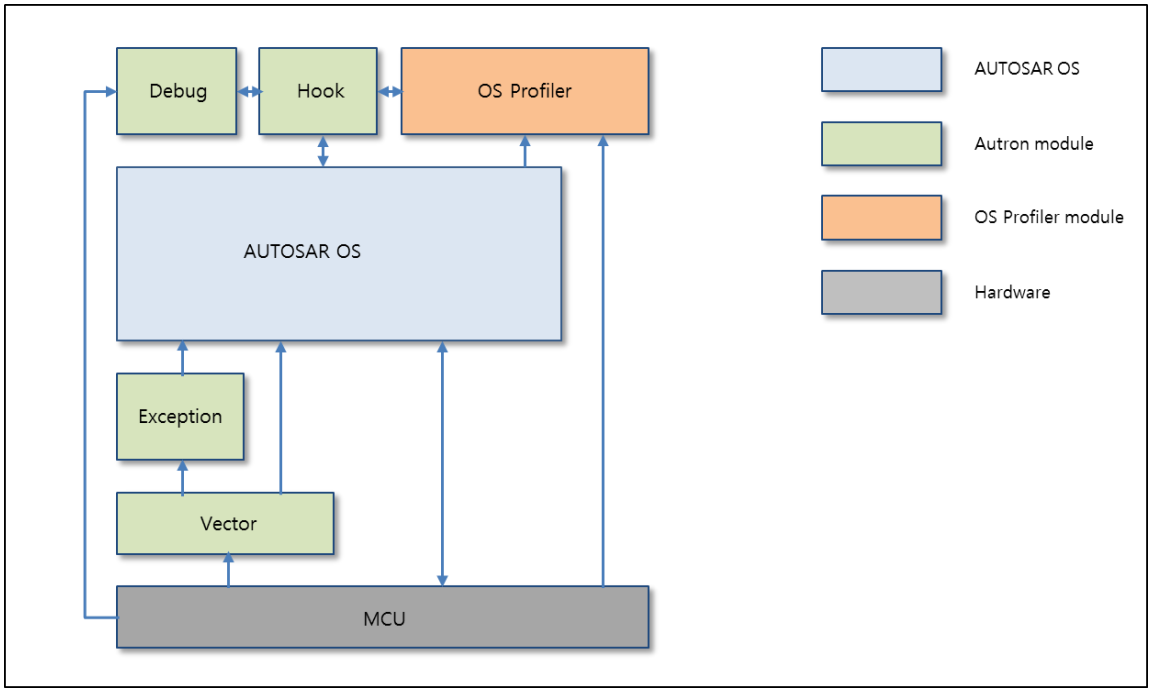
3. AUTOSAR System

3.1 Overview of Software Layers

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



The AUTOSAR OS belongs to System Services, and its structure is as the diagram below. Based on AUTOSAR OS, Vector, Exception, Debug, and Hook are added to complete the AUTOSAR OS module. Those codes added are OS Improvement Codes.



3.2 OS Improvement Code

3.2.1 Startup

OS Improvement Code executes the following functionalities during the startup of RTSW (Runtime Software).

- Configuration of ECC processing
  Enables the detection of non-correctable RAM ECC errors and non-correctable PFLASH ECC errors by setting the FAULT_STRUCT1 register.
  surfaces ECC errors detected by FAULT_STRUCT1 as NMIs by setting the CPUSS_CM4_NMI_CTL0 register. Follow-up processes are handled by Os_CallBackNMInterrupt.

- RAM initialization
  In case of Destructive Reset including Power On Reset, it initializes the SRAM for ECC initialization. The RAM sectors whose OsImpRamInitProperty in the RAM Sector setting is set to 'PowerOnReset' are initialized.

3.2.2 Handling of Memory ECC errors

This section explains the handling of Memory ECC errors by OS Improvement Code.

[Correctable ECC errors]
SWP does not handle correctable ECC errors.

[Non-correctable ECC errors]

Memory Type		Error Support		Note
		Handling	Limitation	
RAM		After clearing all RAM, call AppCallbackOnSystemError. Then reset MCU.	Recovery is not possible if the ECC error occurred due to permanent HW damage. ECC errors may occur repeatedly in such case.	Callback argument: E_OS_SYS_RAMECC
ROM	PFLASH	After calling AppCallbackOnSystemError in OS shutdown/ShutdownHook, performs MCU Reset.	Recovery is not possible. ECC errors may occur repeatedly in such case.	Callback argument: E_OS_SYS_PFLASHECC
	DFLASH	None (all handled by MCAL)	-	See the FLS module of MCAL

Note: When a RAM ECC error occurs, AppCallbackOnSystemError() is called after all RAM area is initialized(cleared), so values of all global variables are unreliable in the callback.

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

This chapter provides the release information on OS Improvement Code, describing the features and restrictions of different versions of OS Improvement Code software product.

4.2 Scope of the release

All descriptions in this document are limited to the following OS Improvement Code.

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

➢ **Features**

- CPULOAD measurement support for New MCU(cyt6bj).

Cause	CPULOAD measurement support for 6BJ MCU
Operation effect	None
Setting effect	None
ASW action	None

4.3.1.2 Version 2.1.3.1

➢ **Improvements**

- English user manual is supported for global users.

Cause	English user manual is supported for global users.
Operation effect	None
Setting effect	None
ASW action	None

4.3.1.3 Version 2.1.3.0

➢ **Improvements**

- Added code to clear the Fault Structure when it is initialized.

Cause	Prevent NMI from occurring by detected fault before MPU enable.
Operation effect	None

Setting effect	None
ASW action	None

4.3.1.4 Version 2.1.2.1

➢ **Improvements**

- Improved the codes to comply with the UNECE Cyber Security regulations (report published)

Cause	Required to comply with the UNECE Cyber Security regulations.
Operation effect	None
Setting effect	None
ASW action	None

4.3.1.5 Version 2.1.2.0

➢ **Improvements**

- Modified reset method in case of RAM ECC (as-is: SW Reset -> to-be: WDT Reset)

Cause	It is necessary to switch to WDT Reset since SW reset results in a lock-up in some cases during the processing of RAM ECC in functional safety tests.
Operation effect	None
Setting effect	None
ASW action	None

4.3.1.6 Version 2.1.1.0

➢ **Improvements**

- Security coding improved to comply with the UNECE Cyber Security regulations

Cause	Required to comply with the UNECE Cyber Security regulations.
Operation effect	None
Setting effect	None
ASW action	None

- The input file list in the comments of the generated files will now be sorted in order.

Cause	Even when there was no modified content in the generated files, if the order of input files in the comments was changed, it was mistakenly recognized as change in file content and caused inconvenience.
Operation effect	None
Setting effect	None
ASW action	None

4.3.1.7 Version 2.1.0.0

➢ Features

- Changed RAM size configurations in the schema and supported new MCUs.

Cause	Supporting 2BL, 4BF MCUs
Operation effect	None
Setting effect	None
ASW action	None

➢ Improvements

- Changed RAM size configurations in the schema.

Cause	Changed the max value of RAM size in the schema (PDF file).
Operation effect	None
Setting effect	None
ASW action	None

- Polyspace (Runtime Error) measurement

Cause	To handle Runtime Error of Polyspace
Operation effect	N/A
Setting effect	N/A
ASW action	N/A

4.3.1.8 Version 2.0.0.0

➢ Features

- Initial version

Cause	Traveo II 4M support
Operation effect	None
Setting effect	None
ASW action	None

4.3.1.9 Version 1.0.0.0

➢ Features

- Initial version

Cause	OS Improvement Code
Operation effect	None
Setting effect	None
ASW action	None

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 Restrictions on using FAULT_STRUCT

- Out of four FAULT_STRUCT registers, OS Improvement Code uses FAULT_STRUCT1, which other users therefore can't use.
- Fault sources processed by OS Improvement Code, namely FLASHC_MAIN_NC_ECC, RAMC0_NC_ECC, and RAMC1_NC_ECC, must not be registered with any other FAULT_STRUCT (0, 2, 3).

4.3.2.3 Restrictions on using the timer

- The CPU/IT Load functionality uses "TCPWM0 Group #2, Counter #0" (TCPWM_0_512 in MCAL Gpt). Therefore, if the CPU/IT Load feature is used, this timer must not be used for other purposes.
- To utilize the CPU/IT Load feature, configurations must be done so that TCPWM0 Group #2, Counter #0 acts as 32bit Free running timers (example: GptPredefTimerChannelConfiguration setting).

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 – see 6.1.1 Os_ErrorValueType for actual values and their meaning
Parameters (Inout)	None
Parameters (Out)	None
Return Value	None

Description	This callback function is called when an ECC error or other OS errors occur. Additional handling is allowed for users here in case of ECC occurrence.
Preconditions	[Caution] When a RAM ECC error occurs, all RAM is initialized to '0' before this callback function is called. Therefore, all global variables are cleared when this function is called.
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	Collect error information from OS Hooks and save it so that users can check the information. - E_OS_LIMIT, E_OS_STACKFAULT occurrence count - Total number of OS errors - The types of the last 8 OS errors, related APIs and parameters
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	A callout function that is called and performs necessary processing when an NMI occurs.
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	A callout function that is called by the startup code before RAM ECC initialization.
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	A callout function that is called by the startup code after RAM ECC initialization.
Preconditions	None
Configuration Dependency	None

6.4 Global Variables

Name:	Type:	Description:
GulOsErrorCount	uint32	OS error occurrence count
GulOsErrorLastPosition	uint32	Position/index of the last error information of GucOsError
Os_GulOsLimitError	uint32	E_OS_LIMIT occurrence count
Os_GulOsStackFaultError	uint32	E_OS_STACKFAULT occurrence count
GucOsError	Os_ErrorType	Save the types of the last 8 OS errors, related APIs, and parameters

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

ER,R255001: 〈OsImpRamDefaultValue〉 shall not be empty

This error occurs if ‘Default Value’ is not set in Ram Sector. Setting this item is mandatory.

ERR255002: At least one of 〈OsImpRamSectionBaseAddress〉 and 〈OsImpRamSectionBaseAddrLinkerSym〉 should be set.

This error occurs if neither ‘Ram Section Base Address’ nor ‘Ram Section Base Addr Linker Sym’ is set in Ram Sector. One of the two must be set.

ERR255003: At least one of 〈OsImpRamSectionSize〉 and 〈OsImpRamSectionSizeLinkerSym〉 should be set.

This error occurs if neither 'Ram Section Size' nor 'Ram Section Size Linker Sym' is set in Ram Sector. One of the two must be set.

ERR255004: <OsImpRamInitProperty> shall not be empty

This error occurs if 'Ram Init Property' is not set in Ram Sector. Setting this item is mandatory.

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

This error occurs if <Parameter Name> is not set as a multiple of 32. The item should be aligned to 32 bytes.

Parameter Name
OsImpRamSectionBaseAddress
OsImpRamSectionSize

7.2.2 Warning Messages

None

7.2.3 Information Messages

None

8. Appendix

8.1 Precautions on Design

8.1.1 Callout function of pre / post RAM ECC

User processes can be added using Os_PreRamInitCallout() and Os_PostRamInitCallout() functions for before RAM ECC initialization and after its completion respectively. Initialization of variables are not complete when Os_PreRamInitCallout() and Os_PostRamInitCallout() is called; therefore use of variables warrants caution.
RAM ECC detection and processing does not work in Os_PreRamInitCallout() on Traveo II MCU.
Definitions for Os_PreRamInitCallout() and Os_PostRamInitCallout() are included in the App_OsHook.c file, which is part of the reference code, as shown below. Users may modify it as necessary.

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

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

8.1.2 Implementing NMI callback for ECC processing

The Os calls Os_CallBackNMInterrupt() when an NMI occurs. To handle ECC, code necessary for the callback function must be implemented.
When RAM ECC occurs, the functions specified below must be called in sequence.
WDT Reset finally occurs in Os_ResetMCU().

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

When PFLASH ECC occurs, the function specified below must be called.

```
ShutdownOS(E_OS_SYS_PFLASHECC);
```

A sample of Os_CallBackNMInterrupt(), which has necessary code implemented for actual ECC handling, is included in the Reference_Code/App_NMI.c file in the distribution.

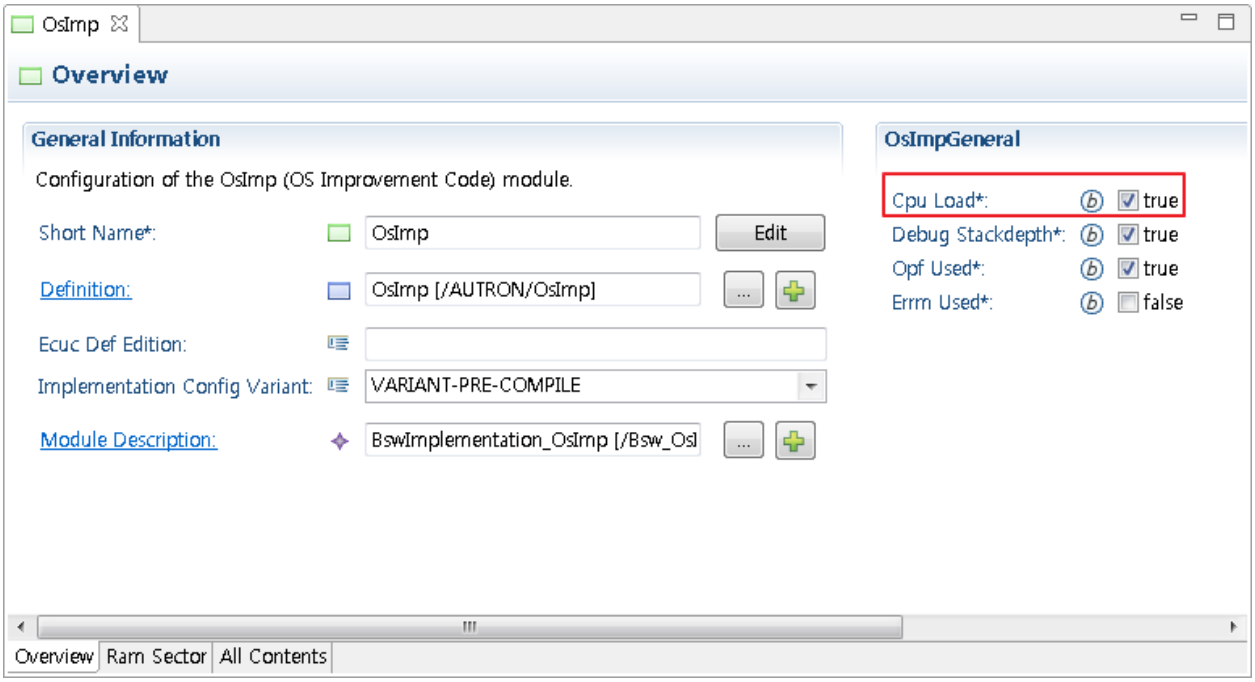
8.2 Exclusive Areas

None

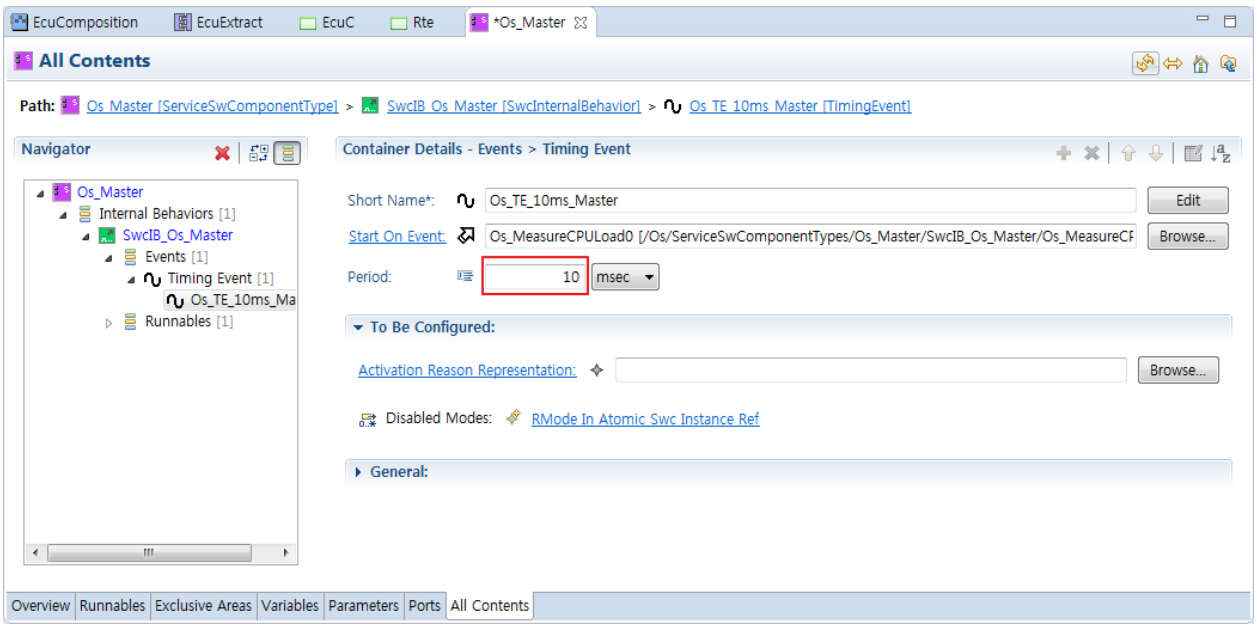
8.3 Debugging Features

8.3.1 CPU & IT Load configuration

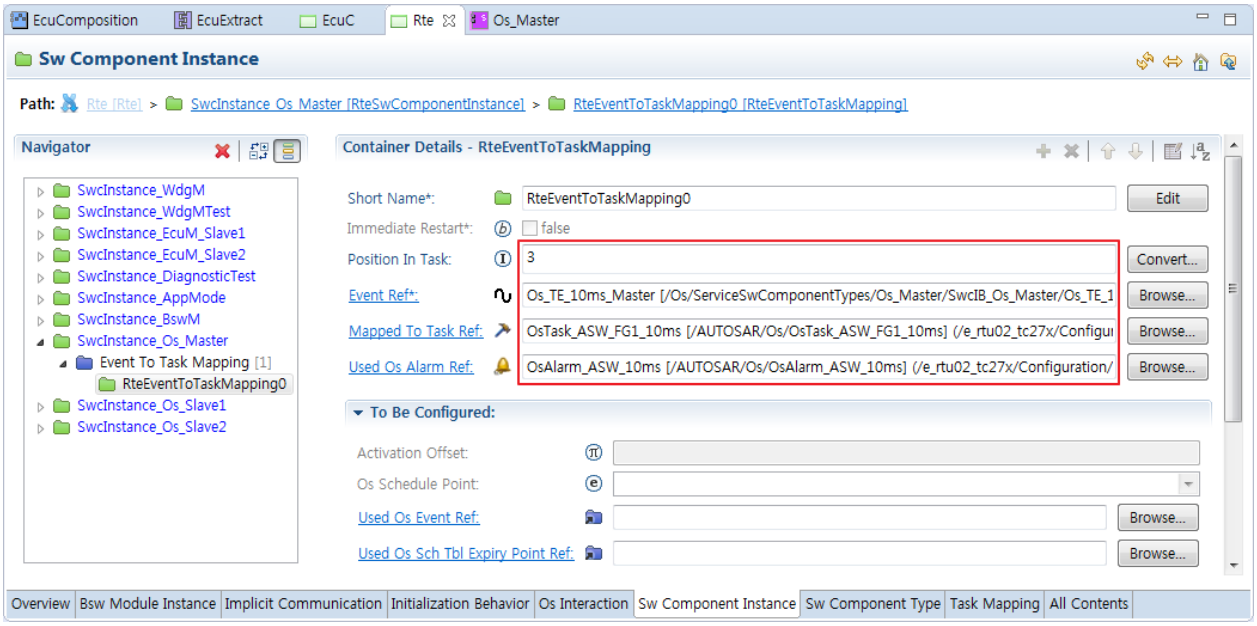
1. Enable CPU/IT Load functionality in the OsImp/OsImpGeneral configuration window.



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.