# HCC OAL for OSE Epsilon User Guide

Version 1.10

For use with OAL for OSE Epsilon versions 2.01 and above

**Date:** 28-Jun-2017 12:23

# Table of Contents

# 1 System Overview

## 1.1 Introduction

This guide is for those who want to use HCC Embedded's OS Abstraction Layer (OAL) for their developments in embedded systems that use the OSE Epsilon operating system from Enea Ab.

The HCC OAL is an abstraction of a Real Time Operating System (RTOS). It defines how HCC software requires an RTOS to behave and its Application Programming Interface (API) defines the functions it requires. Most HCC systems and modules use one or more components of the OAL.

HCC has ported its OAL to OSE Epsilon, in the process creating "hooks" which call OSE Epsilon functions from the HCC abstractions. Once you unzip the files from the **oal_os_ose_epsilon** package into the **oal/os** folder in the source tree, these files will automatically call the correct functions.

The OAL API defines functions for handling the following elements:

- Tasks.
- Events – these are used as a signaling mechanism, both between tasks, and from asynchronous sources such as Interrupt Service Routines (ISRs) to tasks.
- Mutexes – these guarantee that, while one task is using a particular resource, no other task can pre-empt it and use the same resource.
- Interrupt Service Routines (ISRs) – in OSE Epsilon ISRs are platform-specific.

## 1.2 Feature Check

The main features of the module are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Integrated with the HCC OS Abstraction Layer (OAL).
- Allows all HCC middleware to run with the OSE Epsilon RTOS.

## 1.3 Packages and Documents

### Packages

The table below lists the packages that you need in order to use the OAL:

| Package | Description |
|---|---|
| oal_base | The OAL base package. |
| oal_os_ose_epsilon | The OAL for OSE Epsilon package. Unzip the files from this package into the **oal/os** folder in the source tree. |

### Documents

For an overview of HCC RTOS software, see Product Information on the main HCC website.

Readers should note the points in the HCC Documentation Guidelines on the HCC documentation website.

**HCC Firmware Quick Start Guide**

This document describes how to install packages provided by HCC in the target development environment. Also follow the *Quick Start Guide* when HCC provides package updates.

**HCC Source Tree Guide**

This document describes the HCC source tree. It gives an overview of the system to make clear the logic behind its organization.

**HCC OS Abstraction Layer (Base) User Guide**

This document describes the base OAL package, defining the standard functions that must be provided by an RTOS. Use this as your reference to global configuration options and the API.

**HCC OAL for OSE Epsilon User Guide**

This is this document.

## 1.4 Change History

This section describes past changes to this manual.

- To view or download earlier manuals, see Archive: OAL for OSE Epsilon User Guide.
- For the history of changes made to the package code itself, see History: oal_os_ose_epsilon.

The current version of this manual is 1.10. The full list of versions is as follows:

| Manual version | Date | Software version | Reason for change |
|---|---|---|---|
| 1.10 | 2017-06-28 | 2.01 | New *Change History* format. |
| 1.00 | 2015-04-01 | 2.01 | First online version. |

# 2 Source File List

This section describes all the source code files included in the system. These files follow the HCC Embedded standard source tree system, described in the *HCC Source Tree Guide*. All references to file pathnames refer to locations within this standard source tree, not within the package you initially receive.

> **Note:** Do not modify any files except the configuration options file and the PSP files.

## 2.1 Configuration Files

The file **src/config/config_oal_os.h** contains configuration options specific to the system. Configure these as required. (Global configuration parameters are controlled by the base package's configuration file.)

The directory **src/config/ose_epsilon** contains the following files:

| File | Description |
|---|---|
| **gen_proc_list.exe** | Application. |
| **osarm.con** | CON file. |
| **proc_list.c** | Used for ISRs; see Implementation Notes. |
| **proc_list.h** | Used for ISRs. |

## 2.2 Platform Support Package (PSP) Files

These files in the directory **src/psp/target/isr** provide the functions and other elements the core code needs to use, depending on the hardware. Modify these files as required for your hardware.

> **Note:** These are PSP implementations for the specific microcontroller and board; you may need to modify these to work with a different microcontroller and/or development board. See PSP Porting for details.

| File | Description |
|---|---|
| **psp_isr.c** | ISR functions source code. |
| **psp_isr.h** | ISR header file. |
| **psp_isr_fn.s** | File for task context saving. |

## 2.3 Source Files

These files are in the directory **src/oal/os**. **These files should only be modified by HCC.**

| File | Description |
|---|---|
| **oalp_defs.h** | System defines header file. |
| **oalp_event.c** | Event functions source code. |
| **oalp_event.h** | Event functions header file. |
| **oalp_isr.c** | ISR functions source code. |
| **oalp_isr.h** | ISR functions header file. |
| **oalp_mutex.c** | Mutex functions source code. |
| **oalp_mutex.h** | Mutex functions header file. |
| **oalp_task.c** | Task functions source code. |
| **oalp_task.h** | Task functions header file. |

## 2.4 Version File

The file **src/version/ver_oal_os.h** contains the version number of this module. This version number is checked by all modules that use this module to ensure system consistency over upgrades.

# 3 Configuration Options

**Note:** Systemwide configuration options which allow you to disable certain functions or sets of functions are set in the base package's configuration file. See the *HCC OS Abstraction Layer (Base) User Guide* for details.

Set the OSE Epsilon configuration options in the file **src/config/config_oal_os.h**. This section lists the available configuration options and their default values.

**OAL_TICK_RATE**

The tick rate in ms. The default is 1.

**OAL_HIGHEST_PRIORITY, OAL_HIGH_PRIORITY, OAL_NORMAL_PRIORITY, OAL_LOW_PRIORITY, OAL_LOWEST_PRIORITY**

By default these are respectively 5, 10, 15, 20, and 25.

**OAL_EVENT_FLAG_SHIFT**

The amount to shift the event flag by (see the following option). The default is 8.

**OAL_EVENT_FLAG**

The event flag to use for user tasks invoking internal functions. The value of this flag should be over 0x80 because lower bits might be used by internal tasks.

The default is ( 1 << OAL_EVENT_FLAG_SHIFT ).

**OAL_ISR_COUNT**

The maximum number of interrupt sources supported in HCC modules. The default is 2.

# 4 Implementation Notes

The RTOS elements are implemented as follows.

> **Note:** Before compiling the code, run the file **config/ose_epsilon/gen_proc_list.exe** with **osarm.con** as a parameter. This generates the files **proc_list.c** and **proc_list.h** that need to be compiled with the project and placed in **config/ose_epsilon**.

**Events**

There are no rules governing events.

**Mutexes**

There are no rules governing mutexes.

**Tasks**

There are no rules governing tasks.

**ISRs**

The platform ISR is used.

The configuration option OAL_ISR_COUNT defines the number of interrupts supported in HCC modules. All ISR handlers require ISR INIT_INT_NESTED at the beginning and QUIT_INT_NESTED at the end.

The implementation depends on the target microcontroller. The aim is to predefine OAL_ISR_COUNT number of ISR routines that will perform the save context, call the real ISR based on the description passed in *oal_isr_dsc*, and restore the context. This way ISR handlers can be dynamically assigned to the wrapper functions.

The function (fn) passed in *oal_isr_dsc* cannot be called from the ISR; an OSE_ prefix needs to be written before the isr function. To find the real pointer that must be used, do the following:

1. Include the file **config/ose_epsilon/proc_list.h**.
2. Search for oal_isr_dsc->fn in the *osee_int_proc_list* array that has OSEE_INT_PROC_COUNT elements. The whole search needs to be conditionally compiled if OSEE_INT_PROC_COUNT>0. *osee_int_proc_list_t* has two members: fn is the pointer to *oal_isr_dsc->fn* and *osee_fn* is the real address of the ISR.

**Ticks**

Specify the tick rate OAL_TICK_RATE in ms. The default is 1.

# 5 PSP Porting

These functions are provided by the PSP to perform various tasks. They are designed for a specific microcontroller and development board. You may need to port them to work with your hardware solution; they are designed to make porting easy.

The package includes samples in the **psp_isr.c** file.

| Function | Description |
|---|---|
| **psp_isr_install()** | Initializes the ISR. |
| **psp_isr_delete()** | Deletes the ISR, releasing the associated resources. |
| **psp_isr_enable()** | Enables the ISR. |
| **psp_isr_disable()** | Disables the ISR. |
| **psp_int_enable()** | Enables global interrupts. |
| **psp_int_disable()** | Disables global interrupts. |

These functions are described in the following sections.

## 5.1 psp_isr_install

This function is provided by the PSP to initialize the ISR.

### Format

```
int psp_isr_install (
    const oal_isr_dsc_t *   isr_dsc,
    oal_isr_id_t *          isr_id )
```

### Arguments

| Argument | Description | Type |
|---|---|---|
| isr_dsc | The ISR descriptor. | oal_isr_dsc_t * |
| isr_id | The ISR ID. | oal_isr_id_t * |

### Return Values

| Return value | Description |
|---|---|
| OAL_SUCCESS | Successful execution. |
| OAL_ERROR | Operation failed. |

## 5.2 psp_isr_delete

This function is provided by the PSP to delete the ISR, releasing the associated resources.

**Format**

```
int psp_isr_delete ( oal_isr_id_t isr_id )
```

**Arguments**

| Argument | Description | Type |
|---|---|---|
| isr_id | The ISR ID. | oal_isr_id_t |

**Return Values**

| Return value | Description |
|---|---|
| OAL_SUCCESS | Successful execution. |
| OAL_ERROR | Operation failed. |

## 5.3 psp_isr_enable

This function is provided by the PSP to enable the ISR.

**Format**

```
int psp_isr_enable ( oal_isr_id_t isr_id )
```

**Arguments**

| Argument | Description | Type |
|---|---|---|
| isr_id | The ISR ID. | oal_isr_id_t |

**Return Values**

| Return value | Description |
|---|---|
| OAL_SUCCESS | Successful execution. |
| OAL_ERROR | Operation failed. |

## 5.4 psp_isr_disable

This function is provided by the PSP to disable the ISR.

**Format**

```
int psp_isr_disable ( oal_isr_id_t isr_id )
```

**Arguments**

| Argument | Description | Type |
|---|---|---|
| isr_id | The ISR ID. | oal_isr_id_t |

**Return Values**

| Return value | Description |
|---|---|
| OAL_SUCCESS | Successful execution. |
| OAL_ERROR | Operation failed. |

## 5.5 psp_int_enable

This function is provided by the PSP to enable global interrupts.

### Format

```
int psp_int_enable ( void )
```

### Arguments

None.

### Return Values

| Return value | Description |
|---|---|
| OAL_SUCCESS | Successful execution. |
| OAL_ERROR | Operation failed. |

## 5.6 psp_int_disable

This function is provided by the PSP to disable global interrupts.

### Format

```
int psp_int_disable ( void )
```

### Arguments

None.

### Return Values

| Return value | Description |
|---|---|
| OAL_SUCCESS | Successful execution. |
| OAL_ERROR | Operation failed. |