

# HCC OAL for OSE Epsilon User's Guide

Version 1.00

For use with OAL for OSE Epsilon versions 2.01 and above

**Date:** 01-Apr-2015 11:51

All rights reserved. This document and the associated software are the sole property of HCC Embedded. Reproduction or duplication by any means of any portion of this document without the prior written consent of HCC Embedded is expressly forbidden.

HCC Embedded reserves the right to make changes to this document and to the related software at any time and without notice. The information in this document has been carefully checked for its accuracy; however, HCC Embedded makes no warranty relating to the correctness of this document.

# Table of Contents

---

System Overview	3
Introduction	3
Feature Check	4
Packages and Documents	4
Packages	4
Documents	4
Change History	5
Source File List	6
Configuration Files	6
PSP Files	6
Source Files	7
Version File	7
Configuration Options	8
Implementation Notes	9
PSP Porting	10
psp_isr_install	11
psp_isr_delete	12
psp_isr_enable	13
psp_isr_disable	14
psp_int_enable	15
psp_int_disable	16

# 1 System Overview

## 1.1 Introduction

---

This guide is for those who want to use the HCC Embedded OS Abstraction Layer (OAL) for their developments in embedded systems which 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 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 preempt 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:

- It conforms to the HCC Advanced Embedded Framework.
- It is integrated with the OAL base package.
- It provides a standard interface for HCC tasks.
- It provides a standard interface for HCC mutexes.
- It provides a standard interface for HCC events.

## 1.3 Packages and Documents

---

### Packages

The table below lists the packages which 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 <b>oal/os</b> folder in the source tree.

### Documents

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's 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's Guide

This is this document.

## 1.4 Change History

---

This section includes recent changes to this product. For a list of all the changes, refer to the file **src/history/oal/oal\_os\_ose\_epsilon.txt** in the distribution package.

Version	Changes
2.01	Added <b>config_oal.h</b> file which allows disabling/enabling of parts of the OAL.
2.00	Added <i>oal_task_t</i> type added to file <b>oalp_task.h</b> . Added a pointer to <i>oal_task_t</i> as the first parameter of <b>oal_task_create()</b> . The function <b>oal_task_delete()</b> now expects a pointer to <i>oal_task_t</i> instead of <i>oal_task_id_t</i> . Added the <b>oal_task_yield()</b> function.

## 2 Source File List

This section lists and describes all the source code files included in the system. These files follow HCC Embedded's 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
<code>gen_proc_list.exe</code>	Application.
<code>osarm.con</code>	CON file.
<code>proc_list.c</code>	Used for ISRs; see <a href="#">Implementation Notes</a> .
<code>proc_list.h</code>	Used for ISRs.

### 2.2 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
<code>psp_isr.c</code>	ISR functions source code.
<code>psp_isr.h</code>	ISR header file.
<code>psp_isr_fn.s</code>	File for task context saving.

## 2.3 Source Files

---

These files should only be modified by HCC.

File	Description
<code>src/oal/os/oalp_defs.h</code>	System defines header file.
<code>src/oal/os/oalp_event.c</code>	Event functions source code.
<code>src/oal/os/oalp_event.h</code>	Event functions header file.
<code>src/oal/os/oalp_isr.c</code>	ISR functions source code.
<code>src/oal/os/oalp_isr.h</code>	ISR functions header file.
<code>src/oal/os/oalp_mutex.c</code>	Mutex functions source code.
<code>src/oal/os/oalp_mutex.h</code>	Mutex functions header file.
<code>src/oal/os/oalp_task.c</code>	Task functions source code.
<code>src/oal/os/oalp_task.h</code>	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's 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 (  $1 \ll OAL\_EVENT\_FLAG\_SHIFT$  ).

### **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
<b>psp_isr_install()</b>	Initializes the ISR.
<b>psp_isr_delete()</b>	Deletes the ISR, releasing the associated resources.
<b>psp_isr_enable()</b>	Enables the ISR.
<b>psp_isr_disable()</b>	Disables the ISR.
<b>psp_int_enable()</b>	Enables global interrupts.
<b>psp_int_disable()</b>	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.