HCC OAL for OSE User's Guide

Version 1.00

For use with OAL for OSE versions 2.01 and above

Date: 10-Mar-2015 10:36

Table of Contents

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 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, in the process creating "hooks" which call OSE functions from the HCC abstractions. Once you unzip the files from the **oal_os_ose** 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 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	The OAL for OSE package. Unzip the files from this package into the **oal/os** 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 User's Guide

This is this document.

1.4 Change History

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

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

2.1 Configuration File

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.)

2.2 Source Files

These files should only be modified by HCC.

File	Description
src/oal/os/oalp_defs.h	System defines header file.
src/oal/os/oalp_event.c	Event functions source code.
src/oal/os/oalp_event.h	Event functions header file.
src/oal/os/oalp_isr.c	ISR functions source code.
src/oal/os/oalp_isr.h	ISR functions header file.
src/oal/os/oalp_mutex.c	Mutex functions source code.
src/oal/os/oalp_mutex.h	Mutex functions header file.
src/oal/os/oalp_task.c	Task functions source code.
src/oal/os/oalp_task.h	Task functions header file.

2.3 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.

2.4 PSP Files

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

File	Description
psp_isr.c	Source code for the psp_isr_get_info() ISR function.
psp_isr.h	ISR function header file.

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

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_STACK_SIZE

The stack size used for ISR routines. The default is 1024.

4 Implementation Notes

The RTOS elements are implemented as follows.

Events

There are no rules governing events.

Mutexes

There are no rules governing mutexes.

Tasks

There are no rules governing tasks.

ISRs

A combination of OAL ISR and platform ISR is used.

The configuration option OAL_ISR_STACK_SIZE defines the stack size of the ISR processes.

The two PSP template files are used as follows:

- **psp_isr.c** is required to find out the vector number and the priority of the required interrupt. In OSE bios calls must be used to get this information.
- **psp_isr.h** must contain the following definitions:

```
#define ISR_ID(major,minor)      ((major<<16)|minor)
#define ISR_MAJOR(id)            (id>>16)
#define ISR_MINOR(id)            (id&0xffff)
```

Major and minor numbers are used to generate a unique ID for an interrupt. Based on this number, the function **psp_isr_get_info()** (in **psp_isr.c**) can request the vector number and priority by making a bios call. ISR_ID is used by HCC modules to specify the unique ID. ISR_MAJOR and ISR_MINOR decode the major and minor number from the ID.

Ticks

There are no rules governing ticks.

5 psp_isr_get_info

The PSP provides this function to get ISR information based on the ISR_ID.

Note:

- This function is designed for a specific microcontroller and development board. You may need to port it to work with your hardware solution; it is designed to make porting easy.
- The package includes a sample in the **psp_isr.c** file.

```
int psp_isr_get_info (
    uint32_t       id,
    OSVECTOR *     vector,
    OSPRIORITY *   priority)
```

Arguments

Parameter	Description	Type
id	The ID generated by ISR_ID.	uint32_t
vector	On return, the vector number.	OSVECTOR *
priority	On return, the priority.	OSPRIORITY *

Return Values

Return value	Description
OAL_SUCCESS	Successful execution.
OAL_ERROR	Operation failed.