

# HCC OAL for ST OS20 User Guide

Version 1.20

For use with OAL for ST OS20 versions 1.00 and above

**Date:** 28-Jun-2017 14:56

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               | 5  |
| Packages                             | 5  |
| Documents                            | 5  |
| Change History                       | 6  |
| Source File List                     | 7  |
| Configuration File                   | 7  |
| Source Files                         | 7  |
| Version File                         | 7  |
| Platform Support Package (PSP) Files | 8  |
| Configuration Options                | 9  |
| Implementation Notes                 | 10 |
| PSP Porting                          | 11 |
| psp_isr_install                      | 12 |
| psp_isr_delete                       | 13 |
| psp_isr_enable                       | 14 |
| psp_isr_disable                      | 15 |
| psp_int_enable                       | 16 |
| psp_int_disable                      | 17 |

# 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 which use the OS20 operating system from STMicroelectronics.

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 OS20, in the process creating "hooks" which call OS20 functions from the HCC abstractions. Once you unzip the files from the **oal\_os\_stos20** package into the **oal/os** folder in the source tree, these files 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 OS20 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 ST OS20 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_stos20 | The OAL for ST OS20 package. Unzip the files from this package into the <b>oal/os</b> 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 ST OS20 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 ST OS20 User Guide](#).
- For the history of changes made to the package code itself, see [History: oal\\_os\\_stos20](#).

The current version of this manual is 1.20. The full list of versions is as follows:

| Manual version | Date       | Software version | Reason for change                 |
|----------------|------------|------------------|-----------------------------------|
| 1.20           | 2017-06-28 | 1.03             | New <i>Change History</i> format. |
| 1.10           | 2015-04-02 | 1.00             | Added <i>PSP Porting</i> section. |
| 1.00           | 2014-12-04 | 1.00             | 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 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 are in the directory `src/oal/os`. **These files should only be modified by HCC.**

| File                      | Description                  |
|---------------------------|------------------------------|
| <code>oalp_defs.h</code>  | System defines header file.  |
| <code>oalp_event.c</code> | Event functions source code. |
| <code>oalp_event.h</code> | Event functions header file. |
| <code>oalp_isr.c</code>   | ISR functions source code.   |
| <code>oalp_isr.h</code>   | ISR functions header file.   |
| <code>oalp_mutex.c</code> | Mutex functions source code. |
| <code>oalp_mutex.h</code> | Mutex functions header file. |
| <code>oalp_task.c</code>  | Task functions source code.  |
| <code>oalp_task.h</code>  | 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 Platform Support Package (PSP) Files

These files in the directory `src/psp/target/isr` provide functions and 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 functions 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 Guide](#) for details.

Set the ST OS20 configuration options in the file `src/config/config_oal_os.h`. This section lists the available configuration options and their default values.

### **OAL\_MUTEX\_COUNT**

The maximum number of mutexes. The default is 16.

### **OAL\_EVENT\_COUNT**

The maximum number of events. The default is 16.

### **OAL\_HIGHEST\_PRIORITY, OAL\_HIGH\_PRIORITY, OAL\_NORMAL\_PRIORITY, OAL\_LOW\_PRIORITY, OAL\_LOWEST\_PRIORITY**

By default these are respectively 14, 12, 8, 4, and 1.

### **OAL\_EVENT\_FLAG**

The event flag to use for user tasks invoking internal functions. For example, this is used when a task calls an internal function that needs to wait for an event.

The value of this flag should be over 0x80 because lower bits might be used by internal tasks. The default is 0x100.

### **OAL\_USE\_PLATFORM\_ISR**

Keep this at the default of 1 if you want to use platform ISR routines.

## 4 Implementation Notes

The RTOS elements are implemented as follows.

### Events

The configuration option `OAL_EVENT_COUNT` defines the maximum number of events available for HCC modules.

### Mutexes

The configuration option `OAL_MUTEX_COUNT` defines the maximum number of mutexes available for HCC modules.

### Tasks

There are no rules governing tasks.

### ISRs

There are two ways to use the configuration option `OAL_USE_PLATFORM_ISR`:

- Set it to 1 if OS-provided functions can be used for ISR handling (`interrupt_*`).
- Set it to zero if interrupts need to be used with another method. In this case PSP ISR functions need to be ported.

The current template implementation demonstrates when interrupts need to be executed from a task context. In this case `psp_isr_install()` only registers a semaphore (by using an ST library function) which will be signaled when the interrupt occurs. A task waits for the semaphore and calls the registered ISR handler.

### Ticks

There are no rules governing ticks.

## 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.     |