HCC OAL for Systems with No RTOS User's Guide

Version 1.10

For use with OAL for Systems with No RTOS versions 2.09 and above

Date: 09-Apr-2015 17:29

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

Version 1.10

For use with OAL for Systems with No RTOS versions 2.09 and above

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.

OAL Documents Home

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 no Real Time Operating System (RTOS).

The HCC OAL is an abstraction of an 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 systems which use no RTOS, in the process creating "hooks" which call functions from the HCC abstractions. Once you unzip the files from the **oal_os_nos** 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 pre-empt it and use the same resource.
- Interrupt Service Routines (ISRs) – ISRs are platform-specific.

1.2 Feature Check

The main features of the module are the following:

- It is fully MISRA-compliant.
- 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_nos	The OAL for Systems with No RTOS 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 Systems with No RTOS 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_nos.txt** in the distribution package.

Version	Changes
2.09	Fixed problem that meant under some circumstances an event flag could be incorrectly delivered when an event was deleted and initialized again.
2.08	Removed return value for functions where it was not required to eliminate compiler warnings. File **psp_types.h** now used instead of **stdint.h** and **stddef.h**.
2.07	Eliminated warning which occurred when interrupts were not used (OAL_ISR_SUPPORTED is zero).

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 API Header File

The file **src/api/api_oal_os.h** is the only file that should be included by an application using this module. For details of the function it defines, see oal_task_poll().

2.2 Configuration File

The file **src/config/config_oal_os.h** contains some configurable parameters specific to the system. Configure these as required. For detailed explanations of these, see Configuration Options. (Global configuration parameters are controlled by the base package's configuration file.)

2.3 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.

2.4 PSP Files

These files in the directory **src/psp** 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
common/psp_sleep.c	Controls the sleep for milliseconds function.
include/psp_sleep.h	Header for the sleep for milliseconds function.
target/isr/psp_isr.c	ISR functions source code.
target/isr/psp_isr.h	ISR functions header file.

2.5 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 **src/config/config_oal.h** configuration file. See the *HCC OS Abstraction Layer (Base) User's Guide* for details.

Set the 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

These priorities have no meaning in a system without an OS. Do not change any of these values from the default zero.

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 0x100.

OAL_TASK_COUNT

The maximum number of tasks. The default is 8.

OAL_INTERRUPT_ENABLE

Keep this at the default of 1 to enable interrupts. Set it to zero to disable interrupts.

OAL_ISR_COUNT

The maximum number of interrupt sources if interrupts are disabled. The default is 4.

4 oal_task_poll

Use this function to poll all active tasks.

Format

```
void oal_task_poll ( void )
```

Arguments

Arguments
None.

Return Values

Return value
None.

5 Implementation Notes

The RTOS elements are implemented as follows.

Events

The OAL_EVENT_COUNT configuration option sets the maximum number of events. The default is 16.

Mutexes

The OAL_MUTEX_COUNT configuration option sets the maximum number of mutexes. The default is 16.

Tasks

The OAL_TASK_COUNT configuration option sets the maximum number of tasks. The default is 8.

Ticks

There are no rules governing ticks.

ISRs

The OAL_ISR_COUNT configuration option defines the number of interrupts supported in HCC modules. The default is 4.

6 PSP Porting

These functions are provided by the Platform Support Package (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 files.

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.
psp_sleep()	"Sleeps" for a number of milliseconds.

These functions are described in the following sections.

6.1 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.

6.2 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.

6.3 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.

6.4 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.

6.5 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.

6.6 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.

6.7 psp_sleep

This function is provided by the PSP to sleep for a number of milliseconds.

This uses the USB host delay function.

Format

```
void psp_sleep ( uint32_t ms )
```

Arguments

Argument	Description	Type
ms	The number of milliseconds to sleep for.	uint32_t

Return Values

None.