HCC OAL for Freescale MQX™ User's Guide

Version 1.10

For use with OAL for Freescale MQX™ versions 2.02 and above

Date: 02-Apr-2015 13:46

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 Freescale MQX™ operating system.

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 MQX™, in the process creating "hooks" which call MQX™ functions from the HCC abstractions. Once you unzip the files from the **oal_os_mqx** 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) – in MQX™ 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_mqx	The OAL for Freescale MQX™ 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 Freescale MQX™ 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_mqx.txt** in the distribution package.

Version	Changes
2.02	Added the **config_oal.h** file that allows disabling/enabling of parts of the OAL.
2.01	Fixed a problem in **oal_event_delete()** that meant the function did not delete events with an ID greater than OAL_TASK_COUNT.
2.00	Added *oal_task_t* type added to 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 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
psp_isr.c	ISR functions source code.
psp_isr.h	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's Guide* for details.

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

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 5, 10, 15, 20, and 25.

OAL_FIRST_FAST_EVENT

The first index to use for fast events. The default is zero.

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_TASK_COUNT

The maximum number of tasks. This number should be equal to the number of elements in the *MQX_template_list*. The default is 8.

4 Implementation Notes

The RTOS elements are implemented as follows.

Events

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

HCC events require the ability to handle multiple event bits within one event. A lightweight event does not provide the possibility of finding out which event bits were set when it was obtained; this can be very important for a task to decide what to do. Therefore fast events are used.

The configuration option OAL_FIRST_FAST_EVENT sets the index of the first fast event used. You cannot use fast event indexes in the range from OAL_FIRST_FAST_EVENT to OAL_FIRST_FAST_EVENT+OAL_EVENT_COUNT-1

_event_close_fast is not allowed from ISRs. To solve the problem of how/where to release resources allocated by **_event_open_fast** in ISR-s, a matrix of [task ID-s, event pointers]x[no. fast events] is created. Every time an event is set from ISR it looks for an event pointer that belongs to the current task ID based on the fast event index. If this is not present then it opens the event. When an event is destroyed, **event_close_fast** is executed for all tasks which have this event opened. OAL_TASK_COUNT is required for this purpose.

Messages are used for event generation and a set of messages is allocated for this purpose. The first message number can be defined in the **config_oal_os.h** file and the total number of messages used is given by OAL_EVENT_COUNT.

Mutexes

There are no rules governing mutexes.

Tasks

The configuration option OAL_TASK_COUNT should be equal to the number of elements in *MQX_template_list*, the list of created tasks.

ISRs

The platform ISR is used.

The configuration option OAL_ISR_COUNT is required for the platforms only when a header and a footer is required around the real ISR. In this case this option should define the maximum number of interrrupts in all HCC modules.

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
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.