

# Embedded USB Host Hub Class Driver User's Guide

Version 1.10

For use with USBH Hub Class Driver Versions 2.07 and  
above

**Date:** 27-Aug-2014 15:33

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

---

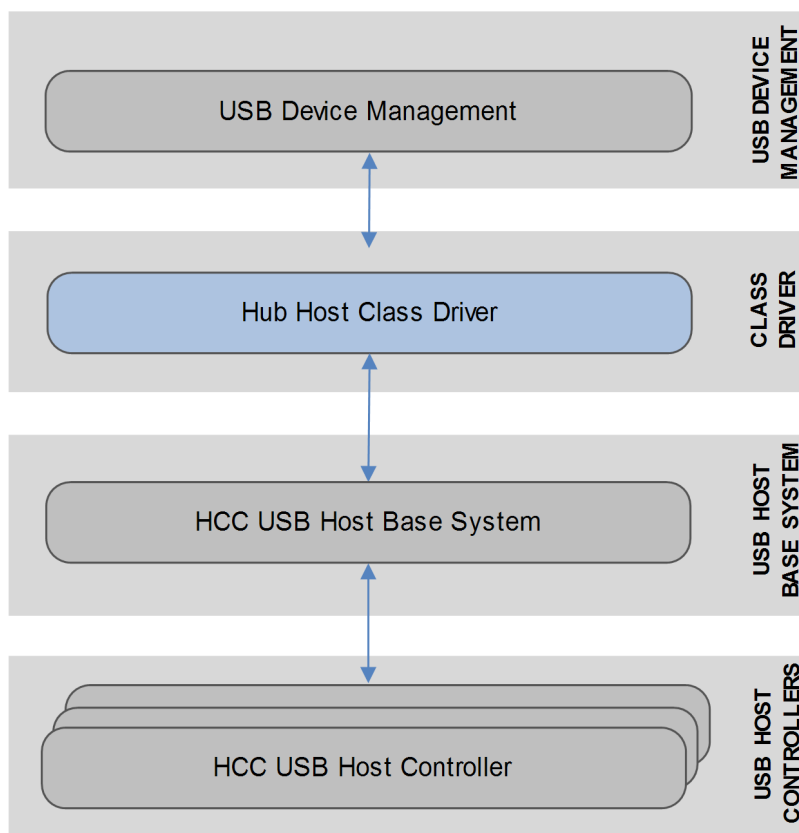
System Overview	3
Introduction	3
Feature Check	4
Packages and Documents	4
Packages	4
Documents	4
Source File List	5
API Header File	5
Source Code	5
Version File	5
API	6
Module Management Functions	6
usbh_hub_init	6
usbh_hub_start	7
usbh_hub_stop	8
usbh_hub_delete	9
Hub Management Functions	10
usbh_hub_get_port_hdl	10
usbh_hub_present	11
usbh_hub_register_ntf	12
Error Codes	13
Types and Definitions	14
t_usbh_ntf_fn	14
Notification Codes	14
Integration	15
OS Abstraction Layer (OAL)	15
PSP Porting	15

# 1 System Overview

## 1.1 Introduction

This guide is for those who want to implement an Embedded USB hub host class driver to control USB hubs. A USB hub is a device that expands one USB port into multiple ports, providing more ports for connecting devices to a host system. The hub class driver is a passive component of the system; once configured and running, it automatically handles connected hubs without any requirement for application control.

The hub package provides a hub host class driver for a USB stack, as shown below.



**Note:** Root hubs contained in the host controller are handled automatically by the port manager and are completely independent of this module.

The lower layer interface is designed to use HCC Embedded's USB Host Interface Layer. This layer is standard over different host controller implementations; this means that the code is unchanged, whichever HCC USB host controller it is interfaced to. For detailed information about this layer, consult the *HCC USB Host Base System User's Guide* that is shipped with the base system.

The package provides a set of API functions for controlling the device. These are described here.

## 1.2 Feature Check

---

The main features of the class driver are the following:

- It conforms to the HCC Advanced Embedded Framework.
- It is compatible with all HCC USB host controllers.
- It supports all devices that conform to the USB Hub specification.
- It supports multiple devices connected simultaneously.
- It uses a system of callbacks for user-specified events.

## 1.3 Packages and Documents

---

### Packages

The table below lists the packages that you need in order to use this module:

Package	Description
<code>hcc_base_doc</code>	This contains the two guides that will help you get started.
<code>usbh_base</code>	The USB host base package. This is the framework used by USB class drivers to communicate over USB using a specific USB host controller package.
<code>usbh_cd_hub</code>	The USB device hub host class driver package described by this document.

### 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 USB Host Base System User's Guide

This document defines the USB host base system upon which the complete USB stack is built.

#### HCC Embedded USB Host Hub Class Driver User's Guide

This is this document.

## 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 of these files.

### 2.1 API Header File

---

The file `src/api/api_usbh_hub.h` is the only file that should be included by an application using this module. For details of the API functions, see [API](#).

### 2.2 Source Code

---

The file `src/usb-host/class-drivers/hub/usbh_hub.c` is the main code for the module. **This file should only be modified by HCC.**

### 2.3 Version File

---

The file `src/version/ver_usbh_hub.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 API

This section documents the Application Programming Interface. It includes all the functions that are available to an application program.

### 3.1 Module Management Functions

---

#### usbh\_hub\_init

Use this function to initialize the class driver and allocate the required resources.

**Note:** You must call this before any other function.

#### Format

```
int usbh_hub_init ( void )
```

#### Arguments

Parameter
None.

#### Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See <a href="#">Error Codes</a> .

## usbh\_hub\_start

Use this function to start the class driver.

**Note:** You must call **usbh\_hub\_init()** before this function.

### Format

```
int usbh_hub_start ( void )
```

### Arguments

Parameter
None.

### Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See <a href="#">Error Codes</a> .

## usbh\_hub\_stop

Use this function to stop the class driver.

### Format

```
int usbh_hub_stop ( void )
```

### Arguments

Parameter
None.

### Return Values

Return value	
USBH_SUCCESS	Successful execution.
Else	See <a href="#">Error Codes</a> .



## usbh\_hub\_delete

Use this function to delete the class driver and release the associated resources.

### Format

```
int usbh_delete ( void )
```

### Arguments

Parameter
None.

### Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See <a href="#">Error Codes</a> .

## 3.2 Hub Management Functions

### usbh\_hub\_get\_port\_hdl

Use this function to get the hub's port handle.

#### Format

```
t_usbh_port_hdl usbh_hub_get_port_hdl ( t_usbh_unit_id uid )
```

#### Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id

#### Return Values

Return value	Description
The port handle.	Successful execution.
USBH_PORT_HDL_INVALID	Invalid port handle.
Else	See <a href="#">Error Codes</a> .

## usbh\_hub\_present

Use this function to check whether a hub is connected.

### Format

```
int usbh_hub_present ( t_usbh_unit_id uid )
```

### Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id

### Return Values

Return value	Description
Zero	No hub is connected.
1	A hub is connected.

## usbh\_hub\_register\_ntf

Use this function to register a notification function for a specified event notification type.

When a device is connected or disconnected, the notification function is called.

**Note:** It is the user's responsibility to provide any notification functions required by the application. Providing such functions is optional.

### Format

```
int usbh_hub_register_ntf (
    t_usbh_unit_id  uid,
    t_usbh_ntf      ntf,
    t_usbh_ntf_fn   ntf_fn )
```

### Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
ntf	The notification ID.	t_usbh_ntf
ntf_fn	The notification function to use when an event occurs.	<a href="#">t_usbh_ntf_fn</a>

### Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
Else	See <a href="#">Error Codes</a> .

### 3.3 Error Codes

If a function executes successfully, it returns with a `USBH_SUCCESS` code, a value of zero. The following table shows the meaning of the error codes:

Return Code	Value	Description
<code>USBH_SUCCESS</code>	0	Successful execution.
<code>USBH_SHORT_PACKET</code>	1	IN transfer completed with short packet.
<code>USBH_PENDING</code>	2	Transfer still pending.
<code>USBH_ERR_BUSY</code>	3	Another transfer in progress.
<code>USBH_ERR_DIR</code>	4	Transfer direction error.
<code>USBH_ERR_TIMEOUT</code>	5	Transfer timed out.
<code>USBH_ERR_TRANSFER</code>	6	Transfer failed to complete.
<code>USBH_ERR_TRANSFER_FULL</code>	7	Cannot process more transfers.
<code>USBH_ERR_SUSPENDED</code>	8	Host controller is suspended.
<code>USBH_ERR_HC_HALTED</code>	9	Host controller is halted.
<code>USBH_ERR_REMOVED</code>	10	Transfer finished due to device removal.
<code>USBH_ERR_PERIODIC_LIST</code>	11	Periodic list error.
<code>USBH_ERR_RESET_REQUEST</code>	12	Reset request during enumeration.
<code>USBH_ERR_RESOURCE</code>	13	OS resource error.
<code>USBH_ERR_INVALID</code>	14	Invalid identifier/type (HC, EP HDL, and so on).
<code>USBH_ERR_NOT_AVAILABLE</code>	15	Item not available.
<code>USBH_ERR_INVALID_SIZE</code>	16	Invalid size.
<code>USBH_ERR_NOT_ALLOWED</code>	17	Operation not allowed.
<code>USBH_ERROR</code>	18	General error.

## 3.4 Types and Definitions

### t\_usbh\_ntf\_fn

The `t_usbh_ntf_fn` definition specifies the format of the notification function. It is defined in the USB host base system in the file `api_usb_host.h`.

#### Format

```
int ( * t_usbh_ntf_fn )(
    t_usbh_unit_id  uid,
    t_usbh_ntf      ntf )
```

#### Arguments

Parameter	Description	Type
uid	The unit ID.	t_usbh_unit_id
ntf	The <a href="#">notification code</a> .	t_usbh_ntf

### Notification Codes

The standard notification codes shown below are defined in the USB host base system in the file `api_usb_host.h`. This module has no specific notification codes of its own.

Notification	Value	Description
USBH_NTF_CONNECT	1	Connection notification code.
USBH_NTF_DISCONNECT	2	Disconnection notification code.

## 4 Integration

This section specifies the elements of this package that need porting, depending on the target environment.

### 4.1 OS Abstraction Layer (OAL)

All HCC modules use the OS Abstraction Layer (OAL) that allows the module to run seamlessly with a wide variety of RTOSes, or without an RTOS.

This module requires the following OAL elements:

OAL Resource	Number Required
Tasks	1
Mutexes	0
Events	1

### 4.2 PSP Porting

The Platform Support Package (PSP) is designed to hold all platform-specific functionality, either because it relies on specific features of a target system, or because this provides the most efficient or flexible solution for the developer.

The module makes use of the following standard PSP macros:

Macro	Package	Component	Description
PSP_RD_LE16	psp_base	psp_endianness	Reads a 16 bit value stored as little-endian from a memory location.
PSP_WR_LE16	psp_base	psp_endianness	Writes a 16 bit value to be stored as little-endian to a memory location.