



# Embedded USB Host Hub Class Driver User Guide

Version 1.50

For use with USBH Hub Class Driver Versions 2.08 and above

Exported on 03/08/2019

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

<b>1 System Overview.....</b>	<b>3</b>
1.1 Introduction .....	4
1.2 Feature Check .....	5
1.3 Packages and Documents .....	6
Packages.....	6
Documents .....	6
1.4 Change History .....	7
<b>2 Source File List .....</b>	<b>8</b>
2.1 API Header File .....	8
2.2 Source Code .....	8
2.3 Version File .....	8
<b>3 Application Programming Interface .....</b>	<b>9</b>
3.1 Module Management Functions.....	9
usbh_hub_init .....	10
usbh_hub_start.....	11
usbh_hub_stop .....	12
usbh_hub_delete .....	13
3.2 Hub Management Functions .....	14
usbh_hub_get_port_hdl.....	15
usbh_hub_present.....	16
usbh_hub_register_ntf .....	17
3.3 Error Codes .....	18
3.4 Types and Definitions .....	19
t_usbh_ntf_fn.....	19
Notification Codes .....	19
<b>4 Integration.....</b>	<b>20</b>
4.1 PSP Porting .....	20

# 1 System Overview

This chapter contains the fundamental information for this module.

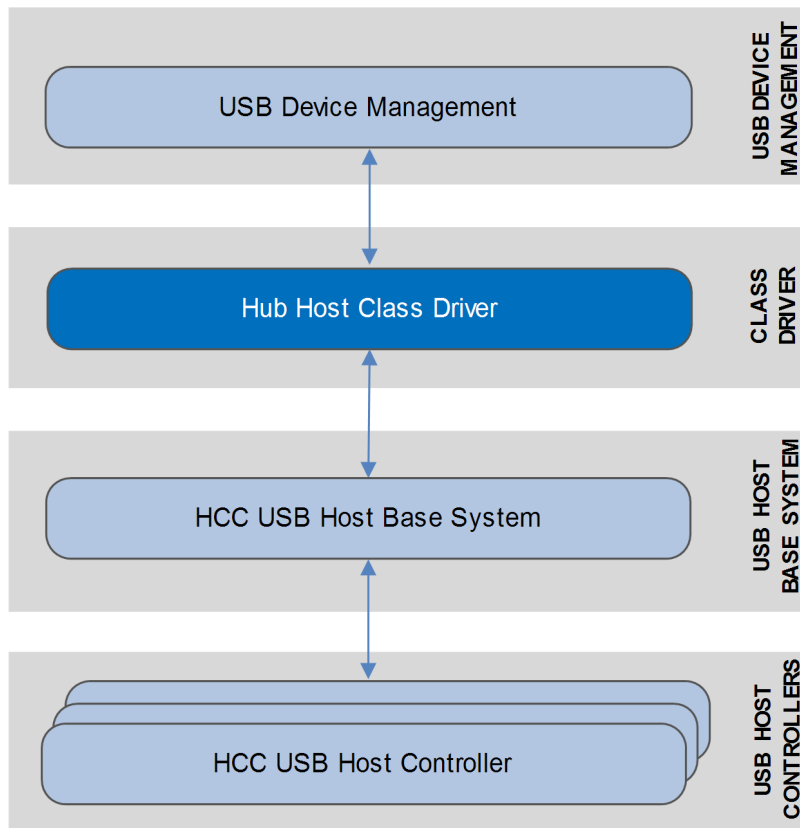
The component sections are as follows:

- [Introduction](#) – describes the main elements of the module.
- [Feature Check](#) – summarizes the main features of the module as bullet points.
- [Packages and Documents](#) – the *Packages* section lists the packages that you need in order to use this module. The *Documents* section lists the relevant user guides.
- [Change History](#) – lists the earlier versions of this manual, giving the software version that each manual describes.

## 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 class driver supports multiple nested hubs.

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 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:

- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Compatible with all HCC USB host controllers.
- Supports all devices that conform to the USB Hub specification.
- Supports multiple devices connected simultaneously.
- Supports multiple nested hubs.
- 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
<b>hcc_base_doc</b>	This contains the two guides that will help you get started.
<b>usbh_base</b>	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.
<b>usbh_cd_hub</b>	The USB device hub host class driver package described by this document.

### Documents

For an overview of HCC's embedded USB stacks, 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 USB Host Base System User 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 Guide**

This is this document.

## 1.4 Change History

This section describes past changes to this manual.

- To view or download earlier manuals, see [USB Host PDFs](#).
- For the history of changes made to the package code itself, see [History: usbh\\_cd\\_hub](#).

The current version of this manual is 1.50. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.50	2019-03-08	2.08	New template.
1.40	2017-06-19	2.08	New <i>Change History</i> format.
1.30	2016-04-20	2.08	Added function group descriptions to API.
1.20	2015-03-27	2.08	Added <i>Change History</i>
1.10	2014-08-27	2.07	First release.

## 2 Source File List

The following sections describe 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 [Application Programming Interface](#).

### 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 Application Programming Interface

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

### 3.1 Module Management Functions

The functions are the following:

Function	Description
<code>usbh_hub_init()</code>	Initializes the module and allocates the required resources.
<code>usbh_hub_start()</code>	Starts the module.
<code>usbh_hub_stop()</code>	Stops the module.
<code>usbh_hub_delete()</code>	Deletes the module and releases the resources it used.

## 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	Description
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

The functions are the following:

Function	Description
<code>usbh_hub_get_port_hdl()</code>	Gets the hub's port handle.
<code>usbh_hub_present()</code>	Checks whether a hub is connected.
<code>usbh_hub_register_ntf()</code>	Registers a notification function for a specified event type.

## 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
0	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 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.	t_usbh_ntf_fn

### 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 0. The following table shows the meaning of the error codes:

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

## 3.4 Types and Definitions

This section describes the *t\_usbh\_ntf\_fn* and the notification codes that are defined in the USBH Base package's API Header file.

### 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 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. For full details of its macros, see the *HCC Base Platform Support Package User Guide*.

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.