USB VUSB Host Controller User Guide

Version 1.40

For use with USBH VUSB Host Controller versions 1.08 and above

Date: 19-Jun-2017 16:22

Table of Contents

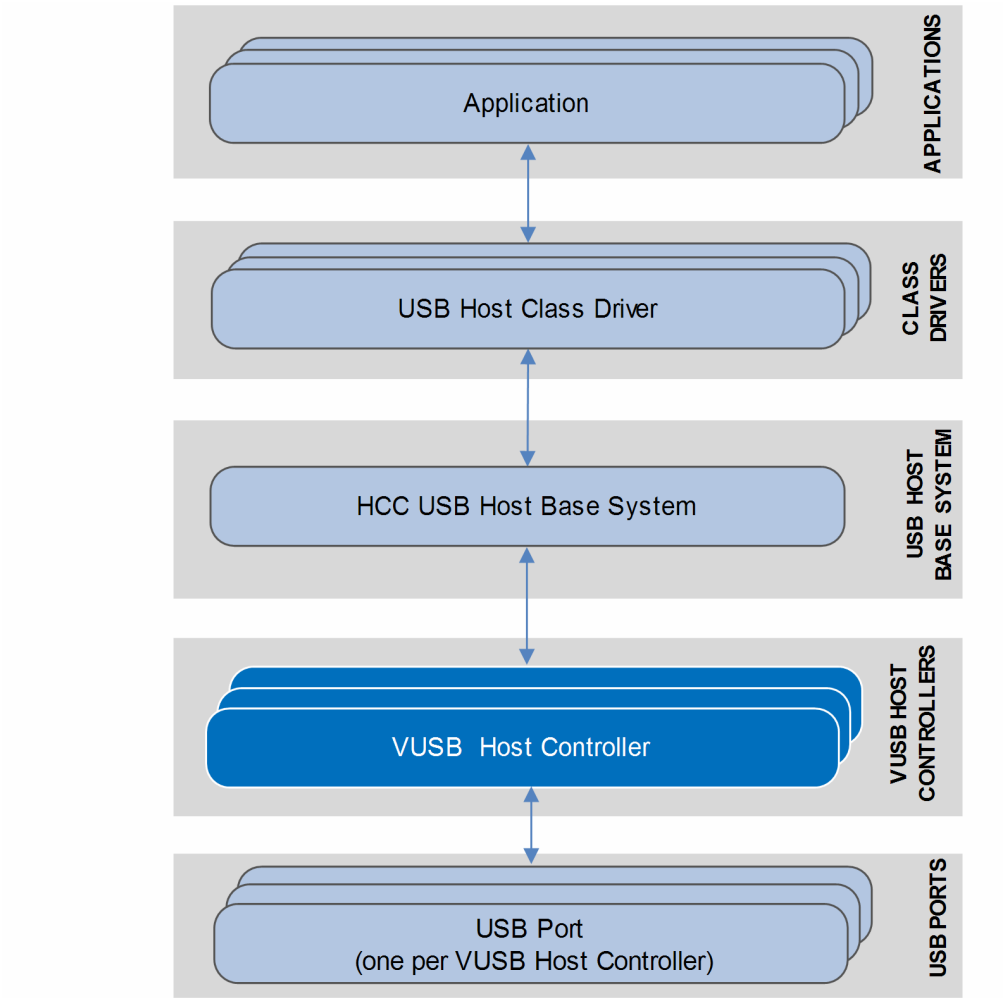
1 System Overview

1.1 Introduction

This guide is for those who want to implement HCC Embedded's VUSB host controller with the HCC USB host stack.

The VUSB module provides a high speed USB 2.0 host controller which provides both full and low speed USB functions. The controller can handle all USB transfer types and, in conjunction with the USB host stack, can be used with any USB class driver. The controller is used by MCUs including the PIC24/32, Freescale™ Kinetis FS, and Freescale™ JM series.

The position of the host controller within the USB stack is shown below:



1.2 Feature Check

The main features of the host controller are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Integrated with HCC USB Host stack and all its class drivers.
- Supports multiple simultaneous host controllers, each with multiple devices attached.
- Supports all USB transfer types: control, bulk, interrupt, and isochronous.

1.3 Packages and Documents

Packages

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

Package	Description
hcc_base_doc	This contains the two guides that will help you get started.
usbh_base	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.
usbh_drv_vusb	The USB VUSB host controller 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 USB VUSB Host Controller 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: VUSB Host Controller User Guide.
- For the history of changes made to the package code itself, see History: usbh_drv_vusb.

The current version of this manual is 1.30. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.40	2017-06-19	1.08	New *Change History* format.
1.30	2017-04-28	1.08	Added to *Configuration Options*.
1.20	2015-04-22	1.06	Added **vusbuh_delay()**.
1.10	2015-03-31	1.05	Added *Change History*, improved *Integration* section.
1.00	2015-03-05	1.05	First release.

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 except the configuration file and PSP files.

2.1 API Header File

The file **src/api/api_usbh_vusb.h** is the only file that should be included by an application using this module. It declares the API functions. For details, see Application Programming Interface.

2.2 Configuration File

The file **src/config/config_usbh_vusb.h** contains all the configurable parameters. Configure these as required. For details of these options, see Configuration Options.

2.3 Source Code

The source code files are in the directory **src/usb-host/usb-driver/vusb**. **These files should only be modified by HCC.**

File	Description
usbh_vusb.c	Source file for VUSB code.
usbh_vusb.h	Header file for VUSB public functions.
usbh_vusb_hc.c	Source file for VUSB HC descriptor.
usbh_vusb_hc.h	VUSB HC descriptor header file.
usbh_vusb_hub.c	Source file for VUSB hub.
usbh_vusb_hub.h	Header file for VUSB hub public functions.
usbh_vusb_reg.h	VUSB register file.

2.4 Version File

The file **src/version/ver_usbh_vusb.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.

2.5 Platform Support Package (PSP) Files

These files provide functions and other elements the core code may need to use, depending on the hardware.

Files are provided for Kinetis and PIC24/32 in the directories **src/psp/target/psp_kinetis_k60** and **src/psp/target/psp_pic32**, respectively.

> **Note:** These are PSP implementations for the specific microcontroller and development board; you may need to modify these to work with a different microcontroller and/or board. See PSP Porting for details.

The directory **psp_kinetis_k60** contains these files:

File	Description
include/hcc_mk60dn512_reg.h	Registers.
usb-host-vusb/usbh_vusb_hw.c	Functions source code.
usb-host-vusb/usbh_vusb_hw.h	Header file for functions.
usb-host-vusb/usbh_vusb_hw_reg.h	Registers header file.

The directory **psp_pic32** contains these files:

File	Description
include/pic32mx4xx_reg.h	PIC32MX4 registers.
include/pic32mx5xx_reg.h	PIC32MX5 registers.
include/pic32mx7xx_reg.h	PIC32MX7 registers.
include/pic32mx_reg.h	Other macros.
usb-host-vusb/usbh_vusb_hw.c	Functions source code.
usb-host-vusb/usbh_vusb_hw.h	Header file for functions.
usb-host-vusb/usbh_vusb_hw_reg.h	Registers header file.

3 Configuration Options

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

MAX_EP

The maximum number of bulk, isochronous, and interrupt endpoints. The default is 12.

MAX_TRANSFERS

The maximum number of simultaneous transfers. The default is 6.

MAX_BULK_PACKET_SIZE

The maximum size of a bulk packet. This is 64; **do not change this**.

MAX_ISO_PACKET_SIZE

The maximum size of an isochronous packet. This is 1023; **do not change this**.

ONE_NAK_PER_FRAME

This controls Bulk endpoints' load configuration. There are two options:

- 0 - Bulk endpoints are polled at the maximum possible rate while NAK is returned by the device.
- 1 (the default) - in case of a NAK handshake, the next token is issued in the next frame.

VUSB_HOST_ISR

The ISR. The default is ISR_ID (USB_ISR, USB_HOST_ISR).

VUSB_HOST_INT_PRIO

The interrupt priority. The default is 1.

VUSB_TFR_IN_ISR

Set one of the following values:

- 1 (the default) – transfer events are handled by interrupts. This is the default.
- 0 – transfer events are handled by polling.

VUSBUH_TRANSFER_TASK_SIZE

The stack size of the transfer task. The default is 1024.

TOTAL_EP

The total number of configured endpoints, plus 1 for endpoint 0. The default is MAX_EP + 1.

VUSB_USE_ALIGN_BUF

Set this to 1 to use 4-byte aligned buffers for transfers. The default is 0.

Set this to 1 for Kinetis MCUs and to zero for PIC32 MCUs.

4 Starting the VUSB Controller

This section shows how to start the host controller and describes the task created. It includes a code example.

4.1 Application Programming Interface

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

usbh_vusbuh_hc

This external interface function provides the host controller descriptor required by the **usbh_hc_init()** function.

Format

```
extern void * const  usbh_vusbuh_hc
```

vusbuh_delay

Use this function to delay the given time.

Format

```
void vusbuh_delay ( uint32_t ms )
```

Arguments

Parameter	Description	Type
ms	The time to delay for in milliseconds.	uint32_t

Return Values

None.

4.2 Host Controller Task

The host controller task handles all completed transfers. Callback requested for the transfer is executed from this task.

The task has the following attributes:

Attribute	Description
Entry point	*vusbuh_transfer_task*
Priority	OAL_HIGHEST_PRIORITY
Stack size	VUSBUH_TRANSFER_TASK_SIZE. The default is 1024.

4.3 Code Example

This example shows how to initialize the host controller. Note the following:

- There is only one external interface function, **usbh_vusbuh_hc()**. To link this host controller to the system, you call the **usbh_hc_init()** function with this function as a parameter.
- The last parameter in the **usbh_hc_init()** call is the number of the host controller.

```
void start_usb_host_stack ( void )

{
int  rc;
rc = hcc_mem_init();

    if ( rc == 0 )
    {
        rc = usbh_init();  /* Initialize the USB host stack */
    }

    if ( rc == 0 )
    {
        /* Attach the VUSB host controller */
        rc = usbh_hc_init( 0, usbh_vusbuh_hc, 0 );
    }

    if ( rc == 0 )
    {
        rc = usbh_start();  /* Start the USB host stack */
    }

    if ( rc == 0 )
    {
        rc = usbh_hc_start( 0 );  /* Start the VUSB Host controller */
    }

.....

}
```

5 Integration

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

5.1 OS Abstraction Layer

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	1
Events	1
ISRs	1

5.2 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 functions:

Function	Package	Element	Description
psp_memcpy()	psp_base	psp_string	Copies a block of memory. The result is a binary copy of the data.
psp_memset()	psp_base	psp_string	Sets the specified area of memory to the defined value.

The host controller makes use of the following functions that must be provided by the PSP. These are designed for you to port them easily to work with your hardware solution. The package includes samples in the **src/psp/target/usb-host-vusb/usbh_vusb_hw.c** file.

Function	Description
vusb_hw_init()	Initializes the device.
vusb_hw_start()	Starts the device.
vusb_hw_stop()	Stops the device.
vusb_hw_delete()	Deletes the device, releasing the associated resources.

These functions are described in the following sections.

> **Note:** HCC can provide samples for different configurations; contact support@hcc-embedded.com.

vusb_hw_init

This function is provided by the PSP to initialize the device.

Format

```
int vusb_hw_init ( void )
```

Arguments

None.

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
USBH_ERROR	Operation failed.

vusb_hw_start

This function is provided by the PSP to start the device.

Format

```
int vusb_hw_start ( void )
```

Arguments

None.

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
USBH_ERROR	Operation failed.

vusb_hw_stop

This function is provided by the PSP to stop the device.

Format

```
int vusb_hw_stop ( void )
```

Arguments

None.

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
USBH_ERROR	Operation failed.

vusb_hw_delete

This function is provided by the PSP to delete the device, releasing the associated resources.

Format

```
int vusb_hw_delete ( void )
```

Arguments

None.

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
USBH_ERROR	Operation failed.