# USB Device Low Level Driver for VUSB User Guide

Version 1.10

For use with USBD Low Level Driver for VUSB versions 1.20 and above

**Date:** 14-Mar-2017 15:35

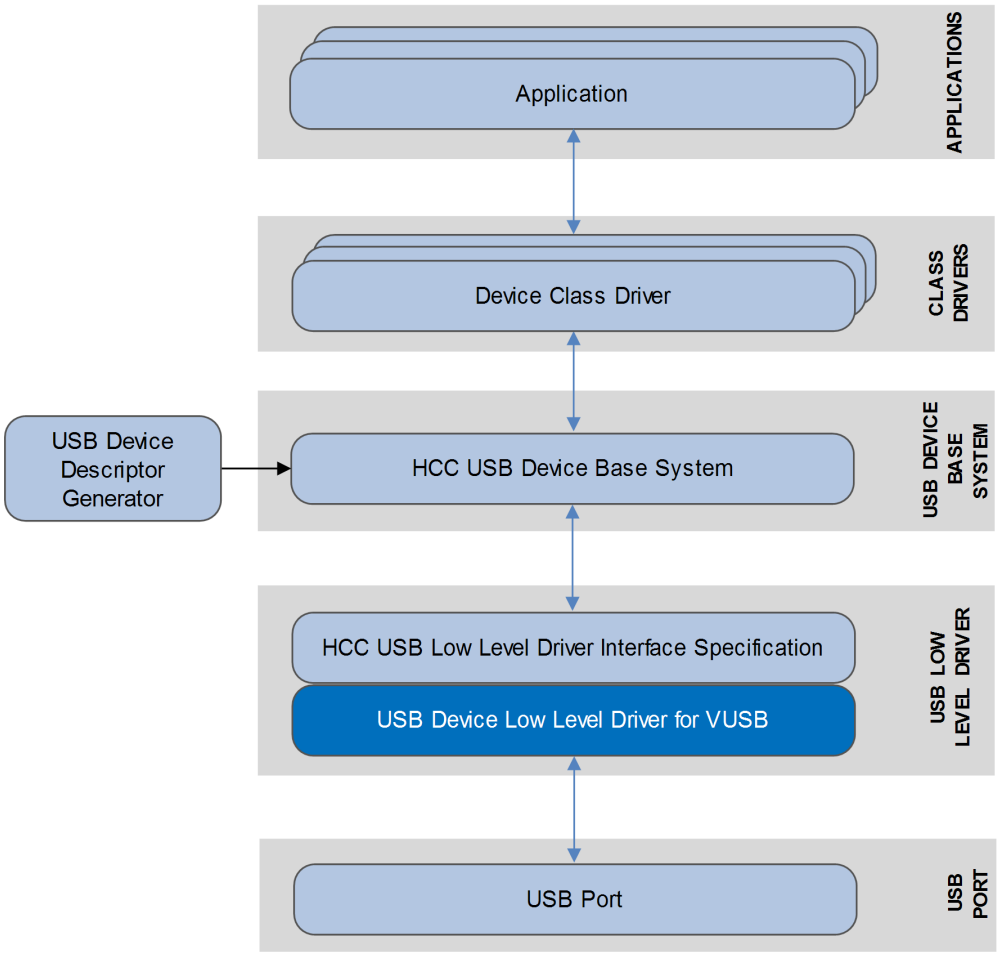
# Table of Contents

# 1 System Overview

## 1.1 Introduction

This guide is for those who want to configure and use the HCC Embedded Low Level Driver for VUSB module with HCC's USB device stack. This module provides a USB device driver for VUSB core microcontrollers (including PIC24/32, Freescale™ Kinetis FS, and the Freescale™ JM series). The driver can handle all USB transfer types and, in conjunction with the USB device stack, can be used with any USB device class driver.

This package provides a low level driver for a USB stack, as shown below.




The low level driver is always started automatically by the USB device stack. The driver is linked to the stack at compile time because each low level driver uses the same function names. This also means that only one driver can run in a system.

## 1.2 Feature Check

The main features of the low level driver are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Conforms to HCC's USB Device Low Level Driver Specification.
- Integrated with the HCC USB device stack and all its class drivers.
- Supports many VUSB core controllers.
- 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. |
| **usbd_base** | The USB device base package. Its source code includes the USB Driver device core. |
| **usbd_drv_vusb** | The VUSB low level 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 Embedded USB Device Base System User Guide**

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

**USB Device Low Level Driver for VUSB User Guide**

This is this document.

## 1.4 Change History

This section includes recent changes to this product. For a complete list of all the changes, refer to the file **hcc/history/usb-device/usbd_drv_vusb.txt** in the distribution package.

| Version | Changes |
|---|---|
| 1.20 | Receiving more consecutive Set Configuration commands from the host, without USB reset, could cause traffic to stop on endpoints with pending TX transfers using a *const* source buffer (located in ROM). The number of Set Configuration requests causing the issue depends on the size of the memory area available for locking via **hcc_mem**. |
| 1.19 | Fixed problem that meant first control transfer after a successful Set Configuration could fail.<br><br>Removed unused code. |
| 1.18 | Fixed the following:<br><br>• Receiving Set Configuration command from the host (without USB reset) could cause immediate TX traffic stop on endpoints with pending TX transfers, lasting until the next USB reset.<br>• Sending from *const* buffers could cause the driver's TX path inoperable after sending a few frames successfully. |
| 1.17 | Updated for compatibility with SOF Timer version 2.x. |

# 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 Configuration File

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

## 2.2 Source Code

These files in the directory **src/usb-device/usb-drivers** are the source code files. **These files should only be modified by HCC**.

| File | Description |
|---|---|
| **usbd_dev.h** | USB driver-specific header file. |
| **usbd_vusb.c** | Source code. |
| **usbd_vusb.h** | USB driver-specific content. |
| **usbd_vusb_reg.h** | VUSB register definitions. |

## 2.3 Version File

The file **src/version/ver_usbd_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.

## 2.4 Platform Support Package (PSP) Files

These files are in the directory **src/psp/target**. They provide functions and elements the core code may need to use, depending on the hardware.

> **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.

| File | Description |
|---|---|
| **include/hcc_k20d72_reg.h** | Register definitions. |
| **usbd-device-vusb/usbd_vusb_hw.c** | Functions source code. |
| **usbd-device-vusb/usbd_vusb_hw.h** | Functions header file. |
| **usbd-device-vusb/usbd_vusb_hw_reg.h** | Compatibility checks. |

# 3 Configuration Options

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

**NUM_OF_HW_EP**

The number of hardware endpoints. Currently only EP0 can be TX and RX. The default is ( MAX_NO_OF_EP + 1 ).

**HCC_VUSB_RENDIAN**

The reverse endianness between VUSB and the system. The default is 0.

**HCC_VUSB_BASE_16**

The VUSB base. Specify 0 for 32 bit or 1 for 16 bit. The default is 0.

**HCC_VUSB_COPY_BUF**

Set this to 1 to copy buffer content from ROM to RAM. The default is 1.

**USBD_VUSB_BDT_SIZE**

The size of one Buffer Descriptor Table (BDT) table in bytes. The default is 8.

> **Note:** The following BDT BIT options define the position of these bits within the Buffer Descriptor Table.

**BDT_CTL_STALL_BIT**

The BDT stall bit. If HCC_VUSB_RENDIAN is 0 the default is 2, otherwise it is 26.

**BDT_CTL_DTS_BIT**

The BDT DTS bit. If HCC_VUSB_RENDIAN is 0 the default is 3, otherwise it is 27.

**BDT_CTL_DATA_BIT**

The BDT Control bit. If HCC_VUSB_RENDIAN is 0 the default is 6, otherwise it is 30.

**BDT_CTL_OWN_BIT**

The BDT Own bit. If HCC_VUSB_RENDIAN is 0 the default is 7, otherwise it is 31.

**BDT_CTL_PID_BIT**

The BDT Product ID bit. The default is 2.

**BDT_BYTE_COUNT_POS_BIT**

The BDT count position bit. The default is 16.

**USBD_VUSB_INT_ID**

The Interrupt ID. The default is ISR_ID( USB_ISR, USB_DEVICE_ISR ).

**USBD_VUSB_INT_PRIO**

The Interrupt priority. The default is 3.

# 4 Integration

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

## 4.1 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 | 0 |
| Mutexes | 0 |
| Events | 1 |
| ISRs | 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. For full details of its functions and macros, see the *HCC Base Platform Support Package User Guide*.

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 module makes use of the following standard PSP macro:

| Macro | Package | Element | Description |
|---|---|---|---|
| PSP_RD_LE16 | psp_base | psp_endianness | Reads a 16 bit value stored as little-endian from a memory location. |

The module makes use of the following PSP functions, provided by the PSP to perform various tasks. These are designed for a specific microcontroller and development board. Their design makes it easy for you to port them to work with your hardware solution. The package includes samples in the PSP file **src/psp/target/usbd-device-vusb/usbd_vusb_hw.c**.

| Function | Description |
|---|---|
| **psp_usbd_vusb_init()** | Initializes the hardware to operate in Device mode. |
| **psp_usbd_vusb_start()** | Starts the device. |
| **psp_usbd_vusb_stop()** | Stops the device. |
| **psp_usbd_vusb_delete()** | Deletes the device, releasing the associated resources. |
| **psp_usbd_vusb_pup_on_off()** | Enables or disables the pull-up resistor. |

These are described in the sections which follow.

## psp_usbd_vusb_init

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

> **Note:** Call this function first.

### Format

```
int psp_usbd_vusb_init ( void )
```

### Arguments

None.

### Return Values

| Return value | Description |
|---|---|
| USBD_SUCCESS | Successful execution. |
| USBD_ERROR | Operation failed. |

## psp_usbd_vusb_start

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

> **Note:** Call **psp_usbd_vusb_init()** before this.

### Format

```
int psp_usbd_vusb_start ( void )
```

### Arguments

None.

### Return Values

| Return value | Description |
|---|---|
| USBD_SUCCESS | Successful execution. |
| USBD_ERROR | Operation failed. |

## psp_usbd_vusb_stop

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

### Format

```
int psp_usbd_vusb_stop( void )
```

### Arguments

None.

### Return Values

| Return value | Description |
|---|---|
| USBD_SUCCESS | Successful execution. |
| USBD_ERROR | Operation failed. |

## psp_usbd_vusb_delete

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

### Format

```
int psp_usbd_vusb_delete( void )
```

### Arguments

None.

### Return Values

| Return value | Description |
|---|---|
| USBD_SUCCESS | Successful execution. |
| USBD_ERROR | Operation failed. |

## psp_usbd_vusb_pup_on_off

This function is provided by the PSP to enable or disable USB pull-up.

### Format

```
int psp_usbd_vusb_pup_on_off ( int on )
```

### Arguments

| Parameter | Description | Type |
|---|---|---|
| on | The pull-up state flag. Set this to 1 to enable pull-up, 0 to disable it. | int |

### Return Values

| Return value | Description |
|---|---|
| USBD_SUCCESS | Successful execution. |
| USBD_ERROR | Operation failed. |