USB Device Low Level Driver for STM32 User Guide

Version 1.10

For use with USBD Low Level Driver for STM32 versions 1.12 and above

Date: 16-Jun-2017 17:42

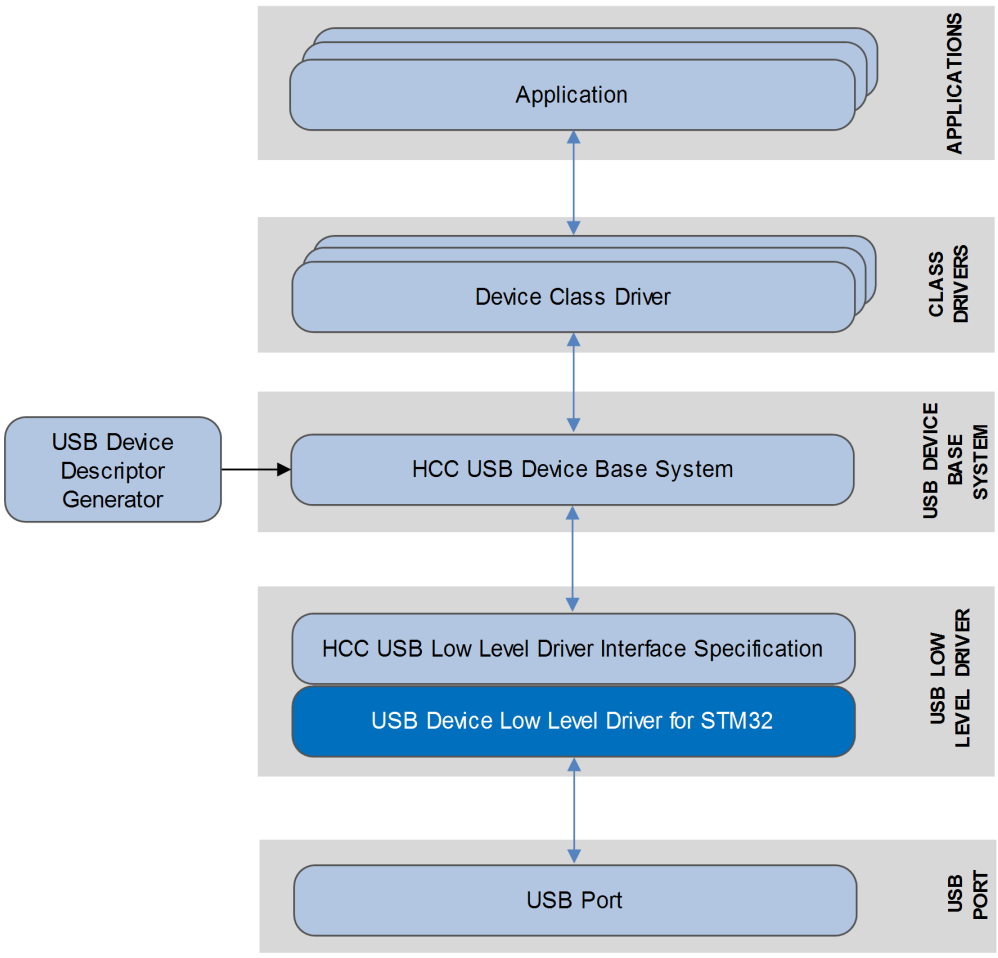
Table of Contents

1 System Overview

1.1 Introduction

This guide is for those who want to configure and use the HCC Embedded Low Level Driver for STM32 module with HCC's USB device stack. This module provides a USB device driver for STM32 controllers from STMicroelectronics. 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 all STMicroelectronics STM32 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_stm32	The STM32 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.

HCC USB Device Low Level Driver for STM32 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 the Archive: USB Device Low Level Driver for STM32 User Guide.
- For the history of changes made to the package code itself, see History: usbd_drv_stm32.

The current version of this manual is 1.10. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.10	2017-06-16	1.12	New *Change History* format.
1.00	2015-04-22	1.09	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 files and PSP files.

2.1 API Header File

The file **src/api/api_usbd_stm32.h** should be included by any application using the system. This is the only file that should be included by an application using this module. For details of the functions, see Callback Functions.

2.2 Configuration Files

The following files in the directory **src/config** contain all the configurable parameters. Configure these as required. For details of these options, see Configuration Options.

File	Description
config_usbd_stm32.c	Contains all the configurable parameters.
config_usbd_stm32.h	Configures FIFO and double-buffering.

2.3 Source Code

The following 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	Macro and type definitions.
usbd_stm32.c	Source file for STM32 code.

2.4 Version File

The file **src/version/ver_usbd_stm32.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

There are three sets of files, located in directories named **src/psp_stm32f10x/target**, **src/psp_stm32f072/target** and **src/psp_stm32l152/target** respectively These 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.

The files are as follows:

File	Description
include/hcc_stm32f10x_reg.h	Register definitions for STM32F10x devices.
include/stm32f0xx_regs.h	Register definitions for STM32F072 devices.
include/stm32l152xx_regs.h	Register definitions for STM32L152 devices.
usbd_stm32/psp_usbd_stm32.c	Function source code.
usbd_stm32/psp_usbd_stm32.h	Function definitions.

3 Configuration Options

This section lists the available options in the two configuration files and their default values.

3.1 src/config/config_usbd_stm32.h

Set the system configuration options in the file **src/config/config_usbd_stm32.h**.

USBD_STM32F0XX, USBD_STM32F1XX, USBD_STM32L1XX

Set the appropriate device family indicator(s) to 1.

STM32_NUM_OF_HW_EP

The number of hardware endpoints for the STM32F10x. The default is 8.

STM32_DOUBLE_BUFFER

Set this to 1 if double-buffered endpoints are used. The default is 1.

STM32_USB_INT_HP_ID

The interrupt ID of high priority USB device interrupts.

If USBD_STM32F1XX and USBD_STM32L1XX are set, the default is ISR_ID(USB_ISR_HP, DEVICE_ISR). Otherwise it is ISR_ID(USB_ISR, DEVICE_ISR).

STM32_USB_INT_LP_ID

The interrupt ID of low priority USB device interrupts.

If USBD_STM32F1XX and USBD_STM32L1XX are set, the default is ISR_ID(USB_ISR_LP, DEVICE_ISR).

STM32_USB_INT_HP_PRI

The interrupt priority of high priority USB device interrupts. The default is 2.

STM32_USB_INT_LP_PRI

The interrupt priority of low priority USB device interrupts. The default is 2.

STM32_ENABLE_ESOF_CALLBACK

Set this to 1 to enable detection of missed SOFs. The default is 0.

3.2 src/config/config_usbd_stm32.c

Set up the FIFO and double buffering configuration in the file **src/config/config_usbd_stm32.c**.

The endpoint FIFO size configuration must reflect the configuration in the device descriptor.

ISO endpoints are always double-buffered.

4 Callback Functions

These callbacks are defined in the file **src/api/api_usbd_stm32.h**.

> **Note:** It is the user's responsibility to provide any callback functions the application requires. Providing such functions is optional.

The functions are the following:

Function	Description
usbd_stm32_enable_esof()	Enables/disables detection of missing SOFs.
usbd_stm32_esof()	Use this when one or more missing SOFs are detected.
usbd_stm32_is_esof_enabled()	Reports whether missing SOF detection is enabled.

4.1 usbd_stm32_enable_esof

Use this optional callback function to enable/disable detection of missing SOFs.

Format

```
void usbd_stm32_enable_esof ( uint8_t b_enable_esof )
```

Arguments

Parameter	Description	Type
b_enable_esof	One of the following: • TRUE: enable missing SOF detection. • FALSE: disable missing SOF detection.	uint8_t

4.2 usbd_stm32_esof

Use this optional callback function when one or more missing SOFs are detected.

To enable this callback, set STM32_ENABLE_ESOF_CALLBACK to 1 in the file **config_usbd_stm32.h**.

Format

```
void usbd_stm32_esof ( uint8_t lsof )
```

Arguments

Parameter	Description	Type
lsof	The number of missed SOFs. The range is from 0 to 3.	uint8_t

4.3 usbd_stm32_is_esof_enabled

Use this optional callback function to find whether missing SOF detection is enabled.

Format

```
uint8_t usbd_stm32_is_esof_enabled ( void )
```

Arguments

Parameter
None.

Return Value

Parameter	Description
TRUE	Missing SOF detection is enabled.
FALSE	Missing SOF detection is disabled

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	0
Mutexes	0
Events	1
ISRs	2

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 PSP functions, provided by the PSP to perform various tasks. They 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 for the STM32F10x and STM32L152 devices in the **psp_usbd_stm32.c** files.

Function	Description
psp_usbd_hw_init()	Initializes the device.
psp_usbd_hw_start()	Starts the device.
psp_usbd_hw_stop()	Stops the device.
psp_usbd_hw_delete()	Deletes the device, releasing associated resources.

These functions are described in the following sections.

psp_usbd_hw_init

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

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

Format

```
int psp_usbd_hw_init ( void )
```

Arguments

None.

Return Values

Return value	Description
USBD_SUCCESS	Successful execution.
USBD_ERROR	Operation failed.

psp_usbd_hw_start

This function is provided by the PSP to start the device. This activates pull-up on D+.

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

Format

```
int psp_usbd_hw_start ( void )
```

Arguments

None.

Return Values

Return value	Description
USBD_SUCCESS	Successful execution.
USBD_ERROR	Operation failed.

psp_usbd_hw_stop

This function is provided by the PSP to stop the device. This deactivates pull-up on D+.

Format

```
int psp_usbd_hw_stop ( void )
```

Arguments

None.

Return Values

Return value	Description
USBD_SUCCESS	Successful execution.
USBD_ERROR	Operation failed.

psp_usbd_hw_delete

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

Format

```
int psp_usbd_hw_delete( void )
```

Arguments

None.

Return Values

Return value	Description
USBD_SUCCESS	Successful execution.
USBD_ERROR	Operation failed.