USB Device Low Level Driver for Synopsys OTG User Guide

Version 1.10

For use with USBD Low Level Driver for Synopsys® OTG versions 3.09 and above

Date: 05-Apr-2018 09:54

Table of Contents

1 System Overview

This chapter contains the fundamental information for this module.

The component sections are as follows:

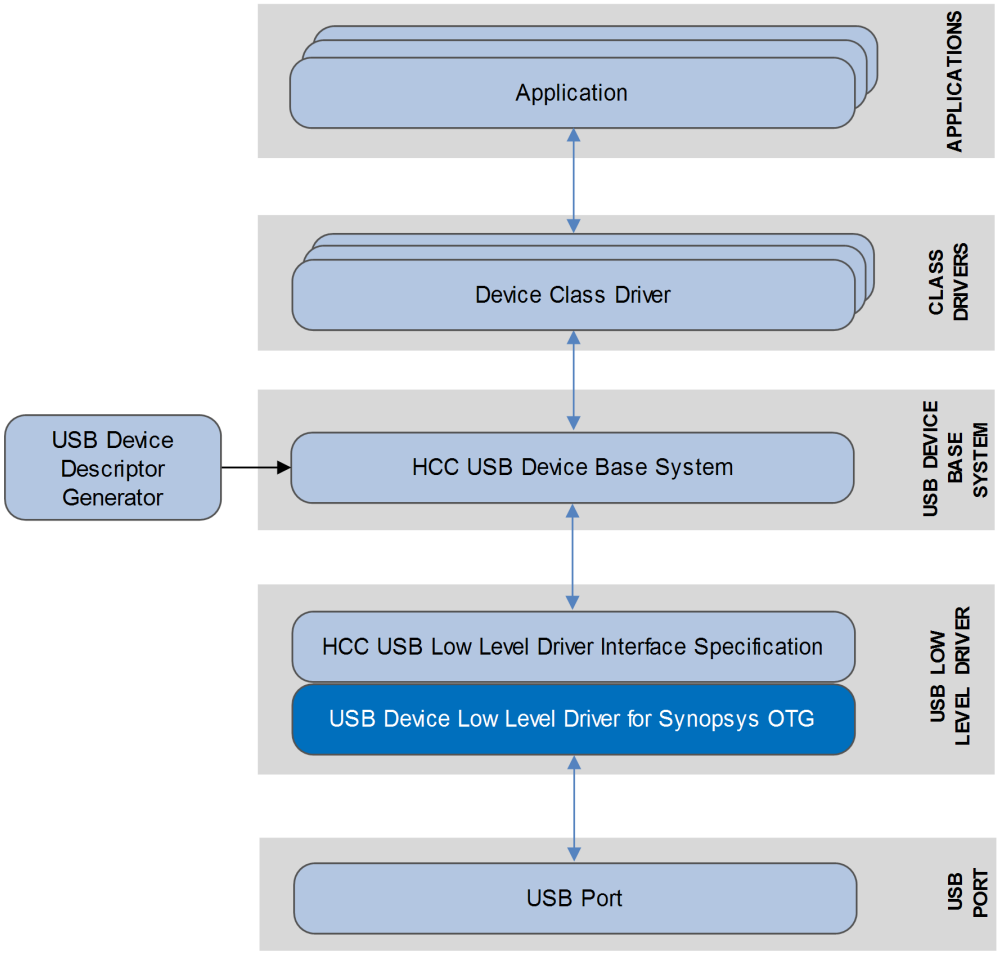
- Introduction – describes the main elements of the module.
- Feature Check – summarizes the main features of the module as bullet points.
- Packages and Documents – the *Packages* section lists the packages that you need in order to use this module. The *Documents* section lists the relevant user guides.
- 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 configure and use the HCC Embedded Low Level Driver for Synopsys® OTG module with HCC's USB device stack. This module provides a USB device driver for Synopsys ®On The Go (OTG) microcontrollers; these include the STM32 connectivity line, STM32F20x, STM32F40x, Infineon XMC microcontrollers, the Silicon Labs EFM32™ family, and some Telit processors.

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 microcontrollers with the Synopsys® On The Go (OTG) core. These include the STM32 connectivity line, STM32F20x, STM32F40x, Infineon XMC microcontrollers, the Silicon Labs EFM32™ family, and some Telit processors.
- 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_synopsys_otg	The Synopsys ® OTG 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 Synopsys OTG User Guide

This is this document.

1.4 Change History

This section describes past changes to this manual.

- To view or download manuals, see USB Device PDFs.
- For the history of changes made to the package code itself, see History: usbd_drv_synopsys_otg.

The current version of this manual is 1.10. The previous versions are as follows:

Manual version	Date	Software version	Reason for change
1.10	2018-04-05	3.09	Removed OTG_STM32L4XX configuration option. Corrected *OS Abstraction Layer* table: added mutex, removed event.
1.00	2015-03-27	3.01	First online version.

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 Configuration Files

These 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_synopsys_otg.h	Configuration options.
config_usbd_synopsys_otg.c	FIFO configuration.

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	Main header file.
usbd_synopsys_otg.c	Source code.
usbd_synopsys_otg_regs.h	Register address definitions.

2.3 Version File

The file **src/version/ver_usbd_synopsys_otg.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_stm32f20x_regs.h	Register address definitions for the STM32F20x.
usbd_synopsys_otg/psp_usbd_synopsys_otg.c	Functions source code.
usbd_synopsys_otg/psp_usbd_synopsys_otg.h	Init/start/stop/delete function definitions.

3 Configuration Options

Set the system configuration options in the following files.

3.1 config_usbd_synopsys_otg.h

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

OTG_USE_HS_PORT

Set this to 1 to use the high speed port. The default is 0.

If this is 0, OTG_ISR_ID is 67 and OTG_BASE is 0x50000000. If this is 1, OTG_ISR_ID is 77 and OTG_BASE is 0x40040000.

OTG_STM32

Set this to a non-zero value only for STM32 implementations. The default is 1.

OTG_NUM_BD_EP

The number of bi-directional endpoints. The total number of endpoints = 2 * OTG_NUM_BD_EP + 1 (Endpoint 0). The default is 3.

If using STM32, set this to 3 for the full speed port and 5 for the high speed port.

OTG_FIFO_SIZE

The FIFO size in 32 bit word units. The default is 320.

OTG_HS_IN_FS_MODE

This only applies if OTG_USE_HS_PORT is set.

The default is 0. Set this to 1 to use the high speed port in full speed mode (no external ULPI used).

OTG_USE_VBUS_IN

Set this to 1 if VBUS detection can be performed (VBUS is connected properly). The default is 0.

OTG_FS_VBUS_CONNECTED

Set this to 1 if VBUS is connected to the OTG_FS_VBUS pin. In this case, VBUS sensing and pulling up of D+ is automatically done by the USB FS core. The default is 0.

OTG_IT_PRIO

The OTG priority. The default is 0.

3.2 config_usbd_synopsys_otg.c

The file **src/config/config_usbd_synopsys_otg.c**.controls FIFO configuration. This is set up as shown below:

```
const uint32_t  otg_fifo_config[OTG_NUM_BD_EP + 2] =
{
  146u  /* Rx FIFO size */
  , 32u /* NP Tx FIFO size */
  , 16u /* IN EP1 FIFO size */
  , 82u /* IN EP2 FIFO size */
  , 8u  /* IN EP3 FIFO size */
  , 16  /* IN EP4 FIFO size */
  , 16  /* IN EP5 FIFO size */
  /*,40*/ /* IN EP6 FIFO size */
  /*,40*/ /* IN EP7 FIFO size */
  /*,40*/ /* IN EP8 FIFO size */
  /*,60*/ /* IN EP9 FIFO size */
  /*,60*/ /* IN EP10 FIFO size */
  /*,60*/ /* IN EP11 FIFO size */
  /*,60*/ /* IN EP12 FIFO size */
  /*,60*/ /* IN EP13 FIFO size */
  /*,60*/ /* IN EP14 FIFO size */
  /*,60*/ /* IN EP15 FIFO size */
};
```

If isochronous endpoints are supported, the isochronous IN endpoint polling interval in [SOFs] is set up as follows. This array must match the configuration descriptor. Use 0 for non-isochronous endpoints.

```
const uint32_t  otg_iso_binterval[OTG_NUM_BD_EP] =
{
  0u     /* IN EP1 */
  , 1u   /* IN EP2 */
  , 512u /* IN EP3 */
  /* , 0u */ /* IN EP4 */
  /* , 0u */ /* IN EP5 */
  /* , 0u */ /* IN EP6 */
  /* , 0u */ /* IN EP7 */
  /* , 0u */ /* IN EP8 */
  /* , 0u */ /* IN EP9 */
  /* , 0u */ /* IN EP10*/
  /* , 0u */ /* IN EP11*/
  /* , 0u */ /* IN EP12*/
  /* , 0u */ /* IN EP13*/
  /* , 0u */ /* IN EP14*/
  /* , 0u */ /* IN EP15*/
};
```

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	1
Events	0
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 macros, see the *HCC Base Platform Support Package User Guide*.

The module makes use of the following standard PSP macros:

Macro	Package	Element	Description
PSP_RD_LE32	psp_base	psp_endianness	Reads a 32 bit value stored as little-endian from a memory location.
PSP_WR_LE32	psp_base	psp_endianness	Writes a 32 bit value to be stored as little-endian to a memory location.

The module makes use of the following PSP functions. These functions are provided by the PSP to perform various tasks. Their design makes it easy for you to port them to work with your hardware solution. The package includes samples in the PSP file **usbd_synopsys_otg/psp_usbd_synopsys_otg.c**.

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.

> **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.

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.