USB Device Low Level Driver for ARC User Guide

Version 1.30

For use with USBD Low Level Driver for ARC® versions 1.13 and above

Date: 16-Oct-2017 12:46

Table of Contents

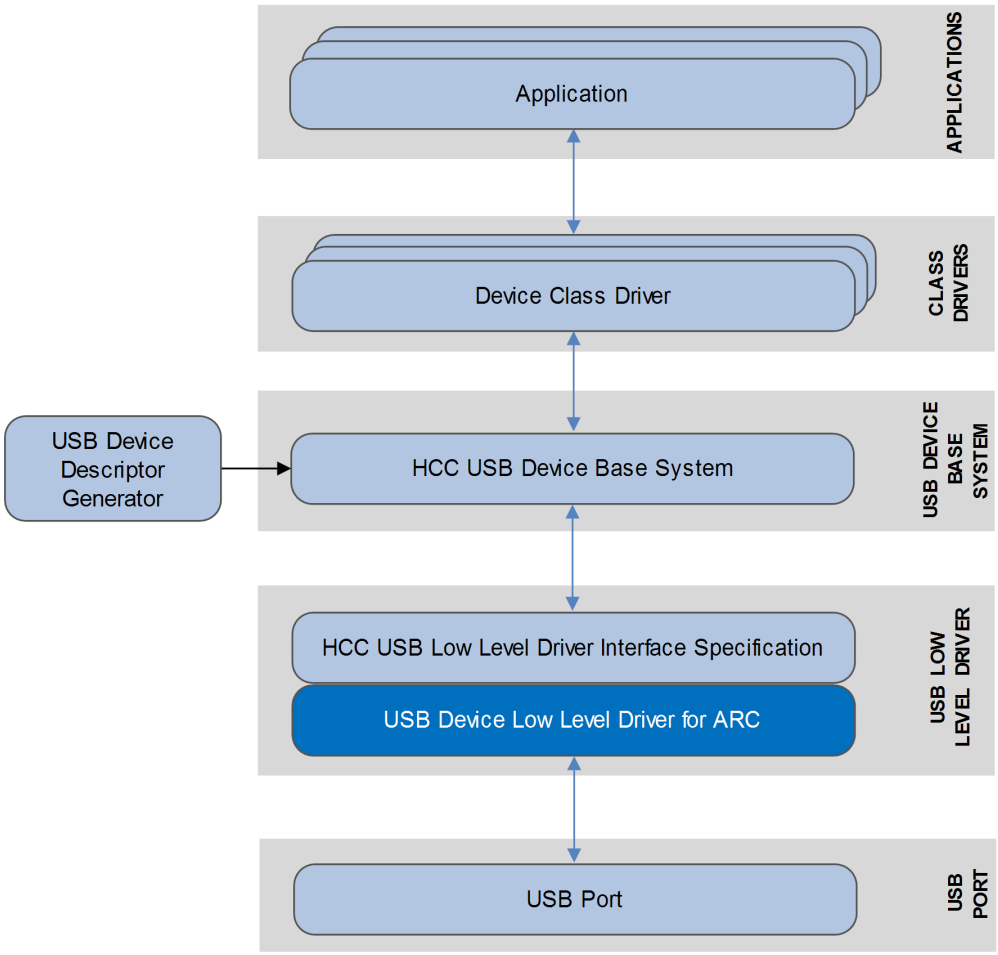
System Overview 3
Introduction 3
Feature Check 4
Packages and Documents 5
Packages 5
Documents 5
Change History 6
Source File List 7
Configuration File 7
Source Code 7
Version File 7
Platform Support Package (PSP) Files 8
Configuration Options 9
Integration 10
OS Abstraction Layer 10
PSP Porting 10
usbd_arc_hw_init 12
usbd_arc_hw_start 13
usbd_arc_hw_stop 14
usbd_arc_hw_delete 15
usbd_arc_hw_pup 16
usbd_arc_hw_dma_is_allowed 17

1 System Overview

1.1 Introduction

This guide is for those who want to configure and use the HCC Embedded Low Level Driver for ARC® module with HCC's USB device stack. This module provides a USB device driver for ARC® controllers from Synopsys®. 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 ARC® controllers.
- Supports all USB transfer types: control, bulk, interrupt, and isochronous.

1.3 Packages and Documents

Packages

This table 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_arc	The ARC® low level driver package described by this document.
util_hcc_mem	The HCC memory management utility.

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 ARC 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 Device PDFs.
- For the history of changes made to the package code itself, see History: usbd_drv_arc.

The current version of this manual is 1.30. The previous versions are as follows:

Manual version	Date	Software version	Reason for change
1.30	2017-10-16	1.13	Added Zync 7000 and TWR-K65F180M PSPs to *Source Files*.
1.20	2017-08-29	1.12	Corrected *Packages* list.
1.10	2017-06-14	1.12	New *Change History* format.
1.00	2015-03-20	1.11	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 file and PSP files.

2.1 Configuration File

The file **src/config/config_usbd_arc.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
arc.c	Source file for ARC code.
arc_reg.h	Header file for read/write register functions.
usbd_dcd.h	DCD header file.
usbd_dev.h	USB driver-specific header file.

2.3 Version File

The file **src/version/ver_usbd_arc.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 directories under **src/psp**. They provide functions and elements the core code may need to use, depending on the hardware.

There are three sets of files, in the following directories:

- psp_template – a standard set of files that you can modify.
- psp_k65f180m – for the TWR-K65F180M development board.
- psp_zync7000 – for Xilinx Zync-7000 devices.

> **Note:**
>
> These are PSP implementations for the specific microcontroller and development board; you can use these directly without porting the PSP. However, you may need to modify them to work with a different microcontroller and/or board; see PSP Porting for details.

The psp_k65f180m and psp_zync7000 sets have a configuration file named **config_usbd_arc.h**.

All of the above file sets include the following files:

File	Description
include/psp_membar.h	Memory barrier header file.
target/membar/psp_membar.c	Memory barrier function.
target/usbd-arc/usbd_arc_hw.h	Functions header file.
target/usbd-arc/usbd_arc_hw_template.c	Functions source code.
target/usbd-arc/usbd_arc_hw_template.h	Hardware definitions.

3 Configuration Options

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

MAX_TRANSFER_SIZE

The maximum number of bytes one endpoint can send with one TX/RX call. The default is (16 * 1024).

USBD_ARC_VBUS_MON

Set this to 1 to enable VBUS monitoring. The default is 0.

If you use On The Go (OTG) mode (combined with USB host), you must disable this option as it requires interrupts to be enabled after **usbd_init()**. With OTG this is not possible as, after **usbd_stop()** is called, the ISR might be required by the host.

USBD_ARC_ISR_ID

The Interrupt ID. The default is ISR_ID(USB, OTG).

USBD_ARC_ISR_PRIORITY

The ISR priority. The default is 0.

USBD_USE_TFR_TASK

The number of USBD ARC transfer tasks. The default is 1.

USBD_ARC_TFR_STACK_SIZE

The ARC transfer task stack size. The default is 1024.

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	1
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 macros:

Macro	Package	Element	Description
PSP_RD_BE16	psp_base	psp_endianness	Reads a 16 bit value stored as big-endian from a memory location.
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 standard PSP functions:

Function	Package	Element	Description
psp_membar()	psp_base	psp_membar	Executes a memory barrier.
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.
psp_rreg32()	psp_base	psp_reg	Reads an ARC register.
psp_wreg32()	psp_base	psp_reg	Writes an ARC register.

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 **src/psp/target/usbd-arc /usbd_arc_hw_template.c**:

Function	Description
usbd_arc_hw_init()	Initializes the device.
usbd_arc_hw_start()	Starts the device.
usbd_arc_hw_stop()	Stops the device.
usbd_arc_hw_delete()	Deletes the device, releasing associated resources.
usbd_arc_hw_pup()	Enables or disables USB pull-up.
usbd_arc_hw_dma_is_allowed()	Checks whether the hardware is capable of executing DMA transfers from an address.

These functions are described in the following sections.

usbd_arc_hw_init

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

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

Format

```
int usbd_arc_hw_init ( void )
```

Arguments

None.

Return Values

Return value	Description
USBD_SUCCESS	Successful execution.
USBD_ERROR	Operation failed.

usbd_arc_hw_start

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

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

Format

```
int usbd_arc_hw_start ( void )
```

Arguments

None.

Return Values

Return value	Description
USBD_SUCCESS	Successful execution.
USBD_ERROR	Operation failed.

usbd_arc_hw_stop

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

Format

```
int usbd_arc_hw_stop( void )
```

Arguments

None.

Return Values

Return value	Description
USBD_SUCCESS	Successful execution.
USBD_ERROR	Operation failed.

usbd_arc_hw_delete

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

Format

```
int usbd_arc_hw_delete( void )
```

Arguments

None.

Return Values

Return value	Description
USBD_SUCCESS	Successful execution.
USBD_ERROR	Operation failed.

usbd_arc_hw_pup

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

Format

```
void usbd_arc_hw_pup ( int on )
```

Arguments

Parameter	Description	Type
on	Flag for the pull-up state. Set this to 1 to enable pull-up, 0 to disable it.	int

Return Values

None.

usbd_arc_hw_dma_is_allowed

This function is provided by the PSP to check whether the hardware is capable of executing DMA transfers from an address.

Format

```
int usbd_arc_hw_dma_is_allowed ( uint32_t addr )
```

Arguments

Parameter	Description	Type
addr	The buffer address.	uint32_t

Return Values

Return value	Description
USBD_SUCCESS	Successful execution.
USBD_ERROR	Operation failed.