USB MUSB CPPI Host Controller User Guide

Version 1.30

For use with USBH MUSB CPPI Host Controller versions 1.04 and above

Date: 19-Jun-2017 15:39

Table of Contents

1 System Overview

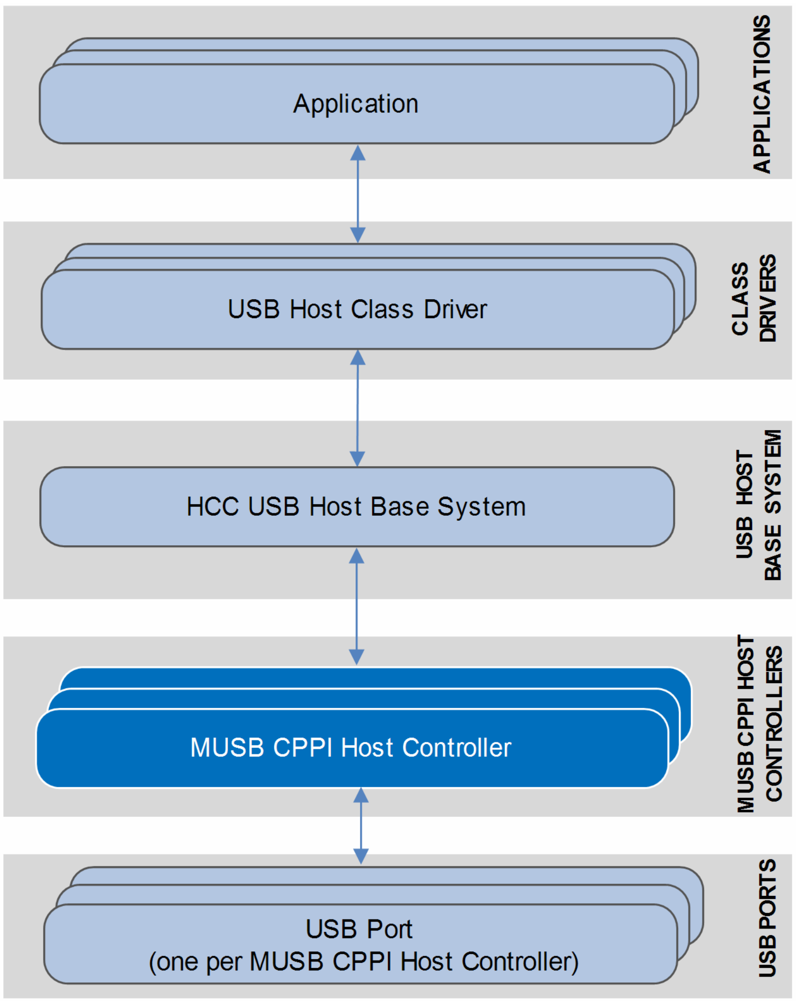
1.1 Introduction

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

The MUSB CPPI module provides a high speed USB 2.0 host controller which provides both full and low speed USB functions. This controller can handle all USB transfer types and, in conjunction with the USB host stack, can be used with any USB class driver.

This module is used by MCUs including Texas Instruments Incorporated devices that have the Mentor USB core and CPPI DMA (DM814x/DM816x).

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 the HCC USB Host stack and all its class drivers.
- Can be used with any MUSB CPPI USB host controller.
- Supports multiple simultaneous MUSB CPPI 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_musb_cppi	The USB MUSB CPPI 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 MUSB CPPI 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: USB MUSB CPPI Host Controller User Guide.
- For the history of changes made to the package code itself, see History: usbh_drv_musb_cppi.

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
1.30	2017-06-19	1.04	New *Change History* format.
1.20	2017-03-31	1.04	Added to *Documents* section.
1.10	2015-03-30	1.03	Added *Change History*, improved *Integration* section.
1.00	2015-03-05	1.03	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_musb_cppi.h** is the only file that should be included by an application using this module. For details, see Starting the Host Controllers.

2.2 Configuration File

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

2.3 Source Code Files

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

File	Description
usbh_musb_cppi.c	Source file for MUSB CPPI code.
usbh_musb_cppi.h	Header file for MUSB CPPI public functions.
usbh_musb_cppi_dsc.c	Descriptor source file.
usbh_musb_cppi_dsc.h	Descriptor header file.
usbh_musb_cppi_hub.c	Source file for MUSB CPPI hub.
usbh_musb_cppi_hub.h	Header file for hub public functions.
usbh_musb_reg.h	MUSB CPPI register file.

2.4 Version File

The file **src/version/ver_usbh_musb_cppi.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 are in the directory **src/psp/target**. They provide functions and elements the core code may need to use, depending on the provide functions the core code needs to call, 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_dm814x_reg.h	Register definitions.
usb-host-musb-cppi/usbh_musb_cppi_hw.c	Functions source code.
usb-host-musb-cppi/usbh_musb_cppi_hw.h	Header file for functions.

3 Configuration Options

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

MUSB_TRANSFER_TASK_STACK_SIZE

The transfer task stack size. The default is 4096.

MUSB_DRV_COUNT

The number of host controllers (if the processor contains multiple host controllers). The default is 1.

MAX_DEVICE

The maximum number of devices supported. The default is 5.

MAX_EP

The maximum number of bulk, isochronous, and interrupt endpoints. The default is 30.

Note:

- The following read/write macros are defined in **psp/include/psp_reg.h** and the parameters are (base, offset) for read and (base, offset, value) for write.
- *base* is always USB_BASE_n.
- *offset* is calculated using EHCI_OFFSET_n+[OPERATIONAL_OFFSET_n] +EHCI_REGISTER_ADDRESS.

MUSB_READ_REG_16, MUSB_WRITE_REG_16

These specify the read/write routines to use when accessing a register. By default these are mapped to **psp_rreg16()** and **psp_wreg16()**.

MUSB_READ_REG_8, MUSB_WRITE_REG_8

Like the previous two parameters, these specify the read/write routines to use when accessing a register. By default these are mapped to **psp_rreg8()** and **psp_wreg8()**.

USB_BASE_0, USB_BASE_1

The base address of the USB module, required if processor-specific registers are available for additional settings. In this case register read/write routines can be used relative to USB_BASE. This can be useful to address registers in **ehci_hw_*** functions. The default for both options is HCC_USB_BASE.

MUSB_OFFSET_0, MUSB_OFFSET_1

This is either the offset of the MUSB controller relative to USB_BASE or, if EHCI_DYNAMIC_OFFSET is set, a pointer to a uint32_t that stores the dynamic offset. The defaults are HCC_MUSB0_OFFSET and HCC_MUSB1_OFFSET.

MUSB_ISR_ID_0, MUSB_ISR_ID_1

The ISR IDs of the MUSB controllers. The defaults are 18 and 19.

MUSB_ISR_PRIO_0, MUSB_ISR_PRIO_1

The ISR priorities of the MUSB controllers. The defaults are both 0.

MUSB_CPPI_DMA_ISR_ID

The ISR ID for DMA traffic. The default is 17.

MUSB_CPPI_DMA_ISR_PRIO

The ISR priority for DMA traffic. The default is 0.

4 Starting the Host Controllers

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

4.1 usbh_musb

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

Format

```
extern void * usbh_musb
```

4.2 Host Controller Tasks

The host controller 0 and 1 tasks handle all completed transfers. Callback requested for the transfer is executed from these tasks.

The tasks have the following attributes:

Attribute	Description
Entry point	*usbh_musb_transfer_task_0* for the first controller. *usbh_musb_transfer_task_1* for the second controller.
Priority	USBH_TRANSFER_TASK_PRIORITY
Stack size	MUSB_TRANSFER_TASK_STACK_SIZE. The default is 4096.

4.3 Sample Code

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

- There is only one external interface function, **usbh_musb_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. In this example there are two controller units so the first call uses 0 and the second call uses 1.

```
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 first MUSB CPPI host controller */
        rc = usbh_hc_init( 0, usbh_musb, 0 );
    }

    if ( rc == 0 )
    {
        /* Attach second MUSB CPPI host controller */
        rc = usbh_hc_init( 0, usbh_musb, 1 );
    }
    if ( rc == 0 )
    {
        rc = usbh_start();   /* Start the USB host stack */
    }

    if ( rc == 0 )
    {
        rc = usbh_hc_start( 0 );   /* Start first MUSB CPPI Host controller */
    }

    if ( rc == 0 )
    {
        rc = usbh_hc_start( 1 );  /* Start second MUSB CPPI 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 per host controller supported. See MUSB_DRV_COUNT.
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-musb-cppi/usbh_musb_cppi_hw.c** file.

Function	Description
usbh_musb_hw_init()	Initializes the device.
usbh_musb_hw_start()	Starts the device.
usbh_musb_hw_stop()	Stops the device.
usbh_musb_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.

usbh_musb_hw_init

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

Format

```
int usbh_musb_hw_init ( t_usbh_unit_id unit )
```

Arguments

Argument	Description	Type
unit	The unit ID.	t_usbh_unit_id

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
USBH_ERROR	Operation failed.

usbh_musb_hw_start

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

Format

```
int usbh_musb_hw_start ( t_usbh_unit_id unit )
```

Arguments

Argument	Description	Type
unit	The unit ID.	t_usbh_unit_id

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
USBH_ERROR	Operation failed.

usbh_musb_hw_stop

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

Format

```
int usbh_musb_hw_stop ( t_usbh_unit_id unit )
```

Arguments

Argument	Description	Type
unit	The unit ID.	t_usbh_unit_id

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
USBH_ERROR	Operation failed.

usbh_musb_hw_delete

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

Format

```
int usbh_musb_hw_delete ( t_usbh_unit_id unit )
```

Arguments

Argument	Description	Type
unit	The unit ID.	t_usbh_unit_id

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
USBH_ERROR	Operation failed.