

USB MUSB DMA Host Controller User Guide

Version 1.00

For use with USBH MUSB DMA Host Controller
versions 1.05 and above

Date: 31-Mar-2017 12:42

All rights reserved. This document and the associated software are the sole property of HCC Embedded. Reproduction or duplication by any means of any portion of this document without the prior written consent of HCC Embedded is expressly forbidden.

HCC Embedded reserves the right to make changes to this document and to the related software at any time and without notice. The information in this document has been carefully checked for its accuracy; however, HCC Embedded makes no warranty relating to the correctness of this document.

Table of Contents

| | |
|--------------------------------------|----|
| System Overview | 3 |
| Introduction | 3 |
| Feature Check | 4 |
| Packages and Documents | 4 |
| Packages | 4 |
| Documents | 4 |
| Change History | 5 |
| Source File List | 6 |
| API Header File | 6 |
| Configuration File | 6 |
| Source Code Files | 6 |
| Version File | 6 |
| Platform Support Package (PSP) Files | 7 |
| Configuration Options | 8 |
| Starting the Host Controllers | 10 |
| usbh_musb | 10 |
| Host Controller Tasks | 10 |
| Code Example | 11 |
| Integration | 12 |
| OS Abstraction Layer | 12 |
| PSP Porting | 13 |
| usbh_musb_hw_init | 14 |
| usbh_musb_hw_start | 15 |
| usbh_musb_hw_stop | 16 |
| usbh_musb_hw_delete | 17 |

1 System Overview

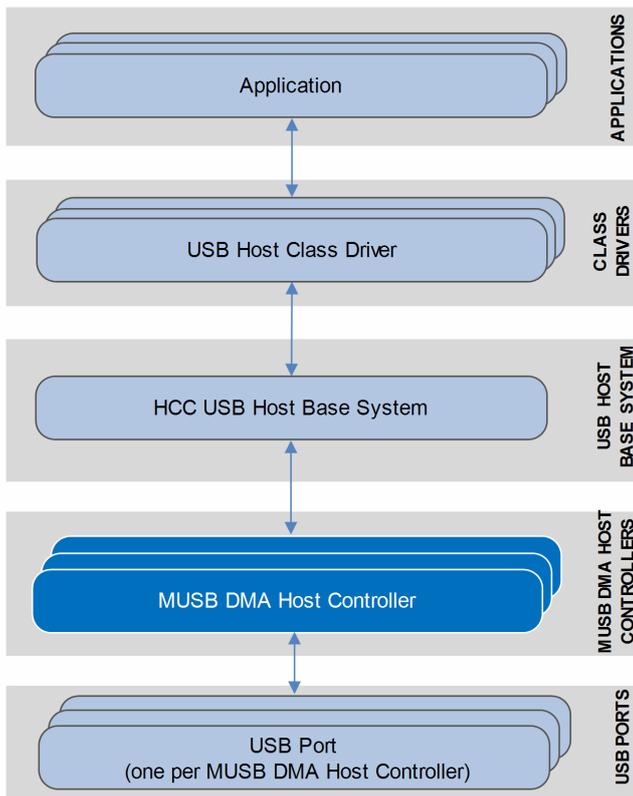
1.1 Introduction

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

The MUSB DMA 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 for Analog Devices Blackfin[®] BF60x, Microchip PIC32MZ, and Smartfusion2 SoC microcontrollers that have the Mentor Graphics[®] MUSB device core.

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.
- Supports multiple simultaneous MUSB DMA controllers, each with multiple devices attached.
- Supports all USB transfer types: control, bulk, interrupt and isochronous.

The system can be used with microcontrollers of the following types that have the Mentor Graphics® MUSB device core:

- Analog Devices Blackfin® BF60x.
- Smartfusion2 SoC.
- Microchip PIC32MZ.

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_dma | The USB MUSB DMA host controller package described by this document. |

Documents

For an overview of HCC's embedded USB stacks, refer to the [Product Information](#) section of 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 DMA Host Controller User Guide

This is this document.

1.4 Change History

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

| Version | Changes |
|---------|---|
| 1.05 | Transfers or enumeration cannot be started as DMA buffers could remain locked. High speed mode can now be enabled/disabled by using the configuration option MUSB_ENABLE_HS. |
| 1.04 | Added virtual address translation. Automatic session restart can now be disabled by using the configuration option MUSB_RESTART_SESSION_AT_DISCONNECT. |
| 1.03 | General code cleanup and fixed the following problems: <ul style="list-style-type: none">• IN transfers ending with short packet could fail.• Simultaneous transfers on different non control endpoints could cause data loss on one of the affected endpoints.• Connecting a device could cause unnoticed VBUS loss. |
| 1.02 | Fixed the maximum NAK rate interval setting. With some Interval settings the USB host high-speed BULK OUT endpoint did not transfer data. |
| 1.01 | Initial 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_dma.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_dma.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_dma`. **These files should only be modified by HCC.**

| File | Description |
|----------------------------------|--|
| <code>usbh_musb_dma.c</code> | Source code for MUSB DMA. |
| <code>usbh_musb_dma.h</code> | Header file for MUSB DMA public functions. |
| <code>usbh_musb_dma_dsc.c</code> | Descriptor source code. |
| <code>usbh_musb_dma_dsc.h</code> | Descriptor header file. |
| <code>usbh_musb_dma_hub.c</code> | Source code for MUSB DMA hub. |
| <code>usbh_musb_dma_hub.h</code> | Header file for hub public functions. |
| <code>usbh_musb_reg.h</code> | MUSB DMA register file. |

2.4 Version File

The file `src/version/ver_usbh_musb_dma.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 two sets of files, in directories **psp_bf60x** and **psp_pic32**. In each case 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 micro-controller and development board; you may need to modify these to work with a different micro-controller and or board. See [PSP Porting](#) for details.

| File | Description |
|--|----------------------------|
| In directory include : | |
| psp_bf60x_regs.h or psp_pic32_regs.h | Register definitions. |
| In directory usb_host_musb_dma : | |
| psp_usbh_musb_dma.c | Functions source code. |
| psp_usbh_musb_dma.h | Header file for functions. |

3 Configuration Options

Set the system configuration options in the file `src/config/config_usbh_musb_dma.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 2 and this is the maximum allowed.

MUSB_ENABLE_HS

Keep the default of 1 to enable High Speed, Full Speed, and Low Speed operation. Set this to 0 to support only Full Speed and Low Speed.

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_32, MUSB_WRITE_REG_32

These specify the read/write routines to use when accessing a register. By default these are mapped to `psp_rreg32()` and `psp_wreg32()`.

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

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 USB_BASE_0 is 0xFFCC1000, for USB_BASE_1 it is xx.

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* value that stores the dynamic offset. The defaults are 0 and xx, respectively.

MUSB_HOST_ISR_0, MUSB_HOST_ISR_1

The ISR IDs of the MUSB controllers. The defaults are PSP_ISR_USB_HOST_ID and xx, respectively.

MUSB_HOST_ISR_PRIO_0, MUSB_HOST_ISR_PRIO_1

The ISR priorities of the MUSB controllers. The defaults are 0 and xx, respectively.

MUSB_DMA_ISR_ID

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

MUSB_DMA_ISR_PRIO

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

MUSB_RESTART_SESSION_AT_DISCONNECT

Set this to 1 to enable the USB core to accept future connections of devices operating at a different speed to a device that has just been disconnected. 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 | <i>usbh_musb_transfer_task_0</i> for the first controller. <i>usbh_musb_transfer_task_1</i> for the second controller. |
| Priority | USBH_TRANSFER_TASK_PRIORITY |
| Stack size | MUSB_TRANSFER_TASK_STACK_SIZE . The default is 4096. |

4.3 Code Example

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 DMA host controller */
rc = usbh_hc_init( 0, usbh_musb, 0 );
}

if ( rc == 0 )
{
/* Attach second MUSB DMA 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 DMA Host controller */
}

if ( rc == 0 )
{
rc = usbh_hc_start( 1 ); /* Start second MUSB DMA 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 |
|---------------------------|----------|------------|--|
| <code>psp_memcpy()</code> | psp_base | psp_string | Copies a block of memory. The result is a binary copy of the data. |
| <code>psp_memset()</code> | 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_dma/psp_usbh_musb_dma.c` file.

| Function | Description |
|------------------------------------|---|
| <code>usbh_musb_hw_init()</code> | Initializes the device. |
| <code>usbh_musb_hw_start()</code> | Starts the device. |
| <code>usbh_musb_hw_stop()</code> | Stops the device. |
| <code>usbh_musb_hw_delete()</code> | 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. |