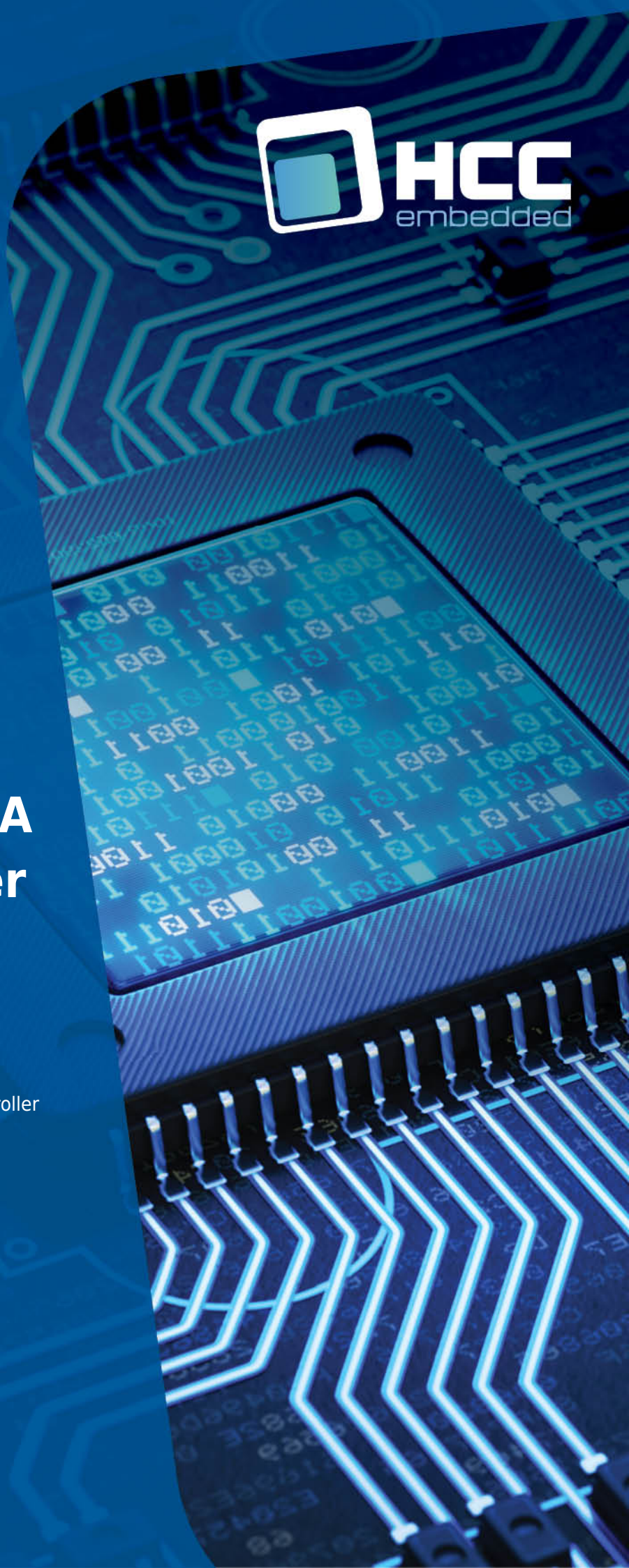


USB MUSB DMA Host Controller User Guide

Version 2.10

For use with USBH MUSB DMA Host Controller versions 1.09 and above

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

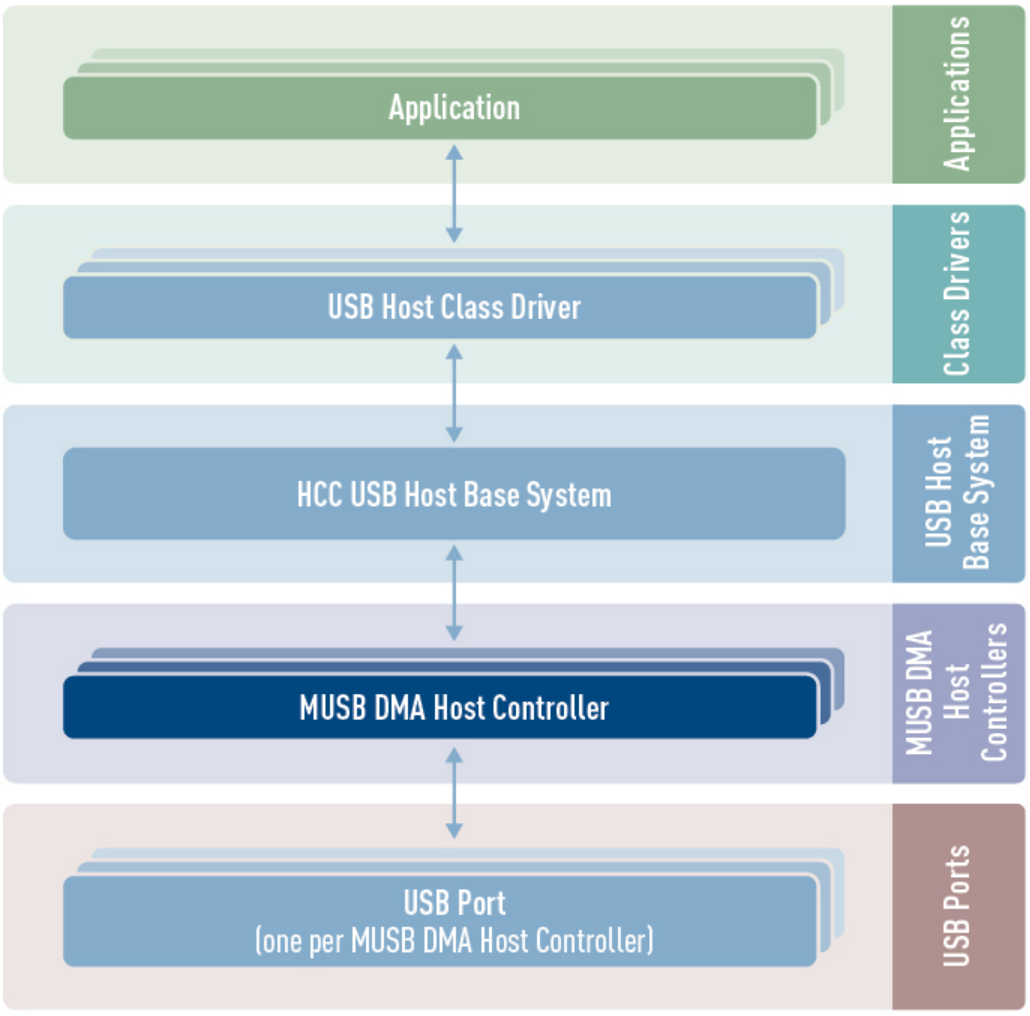
This guide is for those who want to implement HCC Embedded's MUSB DMA USB host controller with the HCC USB host stack.

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 the following microcontrollers, all of which have the Mentor Graphics® MUSB device core interfaced to Mentor DMA:

- Analog Devices Blackfin® BF60x and ADSP-SC5xx
- Microchip Technology PIC32 and PIC32MZ
- Smartfusion2 SoC.

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.

This package can be used with microcontrollers from the following families that have the Mentor Graphics® MUSB device core:

- Analog Devices Blackfin® BF60x and ADSP-SC5xx.
- Smartfusion2 SoC.
- Microchip Technology PIC32 and 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, 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 DMA Host Controller User Guide

This is this document.

1.4. Change History

This section describes past changes to this manual.

- To download this manual as a PDF, see USB Host PDFs.
- For the history of changes made to the package code itself, see History: usbh_drv_musb_dma.

The current version of this manual is 2.10. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
2.10	2020-07-13	1.09	Changed fourth PSP template in *Source Files* from **psp_adsp-sc589_ez-board_freertos** to **psp_adsp-sc57x-sc58x_ez-board_freertos**. Added Memory and ISR functions as subsections of *PSP Porting*.
2.00	2020-02-06	1.08	New template. Added new PSP for **psp_adsp-sc589_ez-board_freertos** to *Source Files*. Added two PSP configuration files.
1.30	2018-10-05	1.07	Made distinction between PIC32 and PIC32MZ devices. Added references to PSP for Microsemi SmartFusion2 development board.
1.20	2017-10-02	1.05	Software versions in this table were wrong.
1.10	2017-06-19	1.05	New *Change History* format.
1.00	2017-03-31	1.05	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.

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.

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.

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
usbh_musb_dma.c	Source code for MUSB DMA.
usbh_musb_dma.h	Header file for MUSB DMA public functions.
usbh_musb_dma_dsc.c	Descriptor source code.
usbh_musb_dma_dsc.h	Descriptor header file.
usbh_musb_dma_hub.c	Source code for MUSB DMA hub.
usbh_musb_dma_hub.h	Header file for hub public functions.
usbh_musb_reg.h	MUSB DMA register file.

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.

Platform Support Package (PSP) Files

There are sets of files in the following directories:

- **psp_bf60x** - Analog Devices Blackfin® BF60x.
- **psp_pic32** and **psp_pic32mz** - Microchip PIC32 and PIC32MZ, respectively.
- **psp_smartfusion2_emcraft-SOM-M2S010** - Microsemi SmartFusion2 (development board: Emcraft SOM-M2S010).
- **psp_adsp-sc57x-sc58x_ez-board_freertos** - Analog Devices ADSP-SC573 EZ-BOARD® with FreeRTOS.

These files provide functions and elements the core code may need to use, depending on the 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.
- In the package these files are offset to avoid overwriting an existing implementation. Copy them to the root **hcc** directory for use.

psp_bf60x

The files are as follows:

File	Description
target/include/psp_bf60x_regs.h	Register definitions.
target/usb_host_musb_dma/psp_usbh_musb_dma.c and **.h**	Functions source code and header file.

This set also has the following version files:

File	Description
ver_psp_proc_reg.h	Version of register definitions file.
ver_psp_usbh_musb_dma.h	Version of header file for MUSB DMA.

psp_pic32 and psp_pic32mz

The files are as follows:

File	Description
target/include/psp_pic32_regs.h and **psp_pic32z_regs.h**	Register definitions.
target/usb_host_musb_dma/psp_usbh_musb_dma.c and **.h**	Functions source code and header file.

This set also has the following version files:

File	Description
ver_psp_proc_reg.h	Version of register definitions file.
ver_psp_usbh_musb_dma.h	Version of header file for MUSB DMA.

psp_smartfusion2_emcraft-SOM-M2S010

The files are as follows:

File	Description
config_usbh_musb_dma.h	A copy of the configuration file with appropriate values.
target/include/hcc_smartfusion2_reg.h	Register definitions.
target/usb_host_musb_dma/psp_usbh_musb_dma.c and **.h**	Functions source code and header file.

This set also has the following version files:

File	Description
ver_psp_proc_reg.h	Version of register definitions file.
ver_psp_usbh_musb_dma.h	Version of header file for MUSB DMA.

psp_adsp-sc57x-sc58x_ez-board_freertos

The files are as follows:

File	Description
config_usbh_musb_dma.h	A copy of the configuration file with appropriate values.
target/include/psp_adsp_sc57x_58x_regs.h	ADSP-SC57x and SC58x register definitions.
target/include/psp_adsp_sc589_regs.h	ADSP-SC589 register definitions.
target/isr/psp_isr.c and **.h**	ISR functions source code and header file.
target/mem/psp_hcc_mem.c and **.h**	Memory/cache functions source code and header file.
target/ usb_host_musb_dma/psp_usbh_musb_dma.c and **.h**	General functions source code and header file.

This set has the following version files:

File	Description
ver_psp.h	Version of PSP.
ver_psp_isr.h	Version of ISR definitions file.
ver_psp_proc_reg.h	Version of register definitions file.
ver_psp_usbh_musb_dma.h	Version of header file for MUSB DMA.

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	*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. 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	One 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_membar()	psp_base	psp_membar	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.

General Functions

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
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 are described in General Functions.

Memory Management Functions

The package includes samples of these functions in the **src/psp/target/mem/psp_hcc_mem.c** file. These are described in Memory Functions.

Function	Description
hcc_mem_buf_alloc()	Allocates a memory area of the required size.
hcc_mem_buf_free()	Frees a previously allocated buffer.
hcc_mem_phy_addr()	Gets the physical address of an uncached address.
hcc_mem_uncached_addr()	Get the uncached address of a physical address.
hcc_mem_flush_cache()	Flushes the cache.
hcc_mem_invalidate_cache()	Invalidates the cache.

ISR Functions

The package includes samples of these functions in the **src/psp/target/mem/psp_isr.c** file. These are described in ISR Functions.

Function	Description
psp_isr_install()	Initializes the ISR.
psp_isr_delete()	Deletes the ISR, releasing the associated resources.
psp_isr_enable()	Enables the ISR.
psp_isr_disable()	Disables the ISR.
psp_int_enable()	Enables global interrupts.
psp_int_disable()	Disables global interrupts.

Note: HCC can provide samples for different configurations; contact support@hcc-embedded.com.

Configuration Files

There are copies of the main configuration file in two of the PSP sets, as follows.

psp_smartfusion2_emcraft-SOM-M2S010

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.

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(b,o)** and **psp_wreg32(b,o,v)**.

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(b,o)** and **psp_wreg16(b,o,v)**.

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(b,o)** and **psp_wreg8(b,o,v)**.

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 0x40043000; 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 USB_IRQn 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 USB_DMA_IRQn.

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.

psp_adsp-sc589_ez-board_freertos

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 (5u * 1024).

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(b, o)** and **psp_wreg32(b, o, v)**.

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(b, o)** and **psp_wreg16(b, o, v)**.

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(b, o)** and **psp_wreg8(b, o, v)**.

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 0x310C1000, 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 configMAX_API_CALL_INTERRUPT_PRIORITY + 1 and xx, respectively.

MUSB_DMA_ISR_ID

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

MUSB_DMA_ISR_PRIO

The ISR priority for DMA traffic. The default is configMAX_API_CALL_INTERRUPT_PRIORITY + 2u.

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.

ISR functions

These functions are provided by the PSP to perform ISR tasks. You may need to port them to work with your hardware solution; they are designed to make porting easy.

The package includes samples in the **psp_isr.c** file.

Function	Description
psp_isr_install()	Initializes the ISR.
psp_isr_delete()	Deletes the ISR, releasing the associated resources.
psp_isr_enable()	Enables the ISR.
psp_isr_disable()	Disables the ISR.
psp_int_enable()	Enables global interrupts.
psp_int_disable()	Disables global interrupts.

These functions are described in the following sections.

psp_isr_install

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

Format

```
int psp_isr_install (
    const oal_isr_dsc_t *   isr_dsc,
    oal_isr_id_t *          isr_id )
```

Arguments

Argument	Description	Type
isr_dsc	The ISR descriptor.	oal_isr_dsc_t *
isr_id	The ISR ID.	oal_isr_id_t *

Return Values

Return value	Description
OAL_SUCCESS	Successful execution.
OAL_ERROR	Operation failed.

psp_isr_delete

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

Format

```
int psp_isr_delete ( oal_isr_id_t isr_id )
```

Arguments

Argument	Description	Type
isr_id	The ISR ID.	oal_isr_id_t

Return Values

Return value	Description
OAL_SUCCESS	Successful execution.
OAL_ERROR	Operation failed.

psp_isr_enable

This function is provided by the PSP to enable the ISR.

Format

```
int psp_isr_enable ( oal_isr_id_t isr_id )
```

Arguments

Argument	Description	Type
isr_id	The ISR ID.	oal_isr_id_t

Return Values

Return value	Description
OAL_SUCCESS	Successful execution.
OAL_ERROR	Operation failed.

psp_isr_disable

This function is provided by the PSP to disable the ISR.

Format

```
int psp_isr_disable ( oal_isr_id_t isr_id )
```

Arguments

Argument	Description	Type
isr_id	The ISR ID.	oal_isr_id_t

Return Values

Return value	Description
OAL_SUCCESS	Successful execution.
OAL_ERROR	Operation failed.

psp_int_enable

This function is provided by the PSP to enable global interrupts.

Format

```
int psp_int_enable ( void )
```

Arguments

None.

Return Values

Return value	Description
OAL_SUCCESS	Successful execution.
OAL_ERROR	Operation failed.

psp_int_disable

This function is provided by the PSP to disable global interrupts.

Format

```
int psp_int_disable ( void )
```

Arguments

None.

Return Values

Return value	Description
OAL_SUCCESS	Successful execution.
OAL_ERROR	Operation failed.

General Functions

These are the basic functions.

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.

usbh_musb_hw_init

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

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

Format

```
int usbh_musb_hw_init ( void )
```

Arguments

None.

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.

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

Format

```
int usbh_musb_hw_start ( void )
```

Arguments

None.

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 ( void )
```

Arguments

None.

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 ( void )
```

Arguments

None.

Return Values

Return value	Description
USBH_SUCCESS	Successful execution.
USBH_ERROR	Operation failed.

Memory Functions

These functions are provided by the PSP to perform memory management. You may need to port them to work with your hardware solution; they are designed to make porting easy.

The package includes samples in the **psp_hcc_mem.c** file.

Function	Description
hcc_mem_buf_alloc()	Allocates a memory area of the required size.
hcc_mem_buf_free()	Frees a previously allocated buffer.
hcc_mem_phy_addr()	Gets the physical address of an uncached address.
hcc_mem_uncached_addr()	Get the uncached address of a physical address.
hcc_mem_flush_cache()	Flushes the cache.
hcc_mem_invalidate_cache()	the cache.

These functions are described in the following sections.

hcc_mem_buf_alloc

This function is provided by the PSP to allocate a memory area of the required size, depending on the memory type.

Note: If an uncached area is requested and the MMU provides an uncached virtual area, after statically/dynamically allocating the area the uncached address must be returned.

Format

```
int hcc_mem_buf_alloc (
    uint8_t       b_uncached,
    uint32_t      size,
    uint32_t      align,
    uint8_t * *   pp_buf )
```

Arguments

Argument	Description	Type
b_uncached	TRUE if an uncached area is requested.	uint8_t
size	The amount of memory to allocate.	uint32_t
align	The requested alignment of the buffer.	uint32_t
pp_buf	A pointer to the buffer.	uint8_t * *

Return Values

Return value	Description
HCC_MEM_SUCCESS	Successful execution.
HCC_MEM_ERROR	Operation failed.

hcc_mem_buf_free

This function is provided by the PSP to release a previously allocated buffer.

Format

```
int hcc_mem_buf_free (
    uint8_t     b_uncached,
    uint8_t *   p_buf   )
```

Arguments

Argument	Description	Type
b_uncached	TRUE if this is an uncached area.	uint8_t
p_buf	A pointer to the buffer to free.	uint8_t *

Return Values

Return value	Description
HCC_MEM_SUCCESS	Successful execution.
HCC_MEM__ERROR	Operation failed.

hcc_mem_phy_addr

This function is provided by the PSP to get the physical address of an uncached address.

Note: If the uncached address is the physical address, the same address must be returned.

Format

```
void * hcc_mem_phy_addr ( void * addr )
```

Arguments

Argument	Description	Type
addr	The uncached address.	void *

Return Values

The physical address

hcc_mem_uncached_addr

This function is provided by the PSP to get the uncached address of a physical address.

Note: If the physical address is the uncached address, the same address must be returned.

Format

```
void * hcc_mem_uncached_addr ( void * addr )
```

Arguments

Argument	Description	Type
addr	The physical address.	void *

Return Values

The uncached address.

hcc_mem_flush_cache

This function is provided by the PSP to flush the cache.

Format

```
int hcc_mem_flush_cache (
    void *      addr,
    uint32_t    size )
```

Arguments

Argument	Description	Type
addr	The start address.	void *
size	The amount of memory to flush.	uint32_t

Return Values

Return value	Description
HCC_MEM_SUCCESS	Successful execution.
HCC_MEM_ERROR	Operation failed.

hcc_mem_invalidate_cache

This function is provided by the PSP to invalidate the cache.

Note: This function should only return HCC_MEM_SUCCESS if the buffer is aligned to the cache line, *size* is an exact multiple of the cache line size, and the buffer was successfully invalidated.

Format

```
int hcc_mem_invalidate_cache (
    void *     addr,
    uint32_t   size )
```

Arguments

Argument	Description	Type
addr	The start address.	void *
size	The amount of memory to invalidate.	uint32_t

Return Values

Return value	Description
HCC_MEM_SUCCESS	Successful execution.
HCC_MEM_ERROR	Operation failed.

6. Version

Version 2.10

For use with USBH MUSB DMA Host Controller versions 1.09 and above