SmartFusion2 MSS HPDMA Driver User's Guide

Version 2.2

Table of Contents

Introduction

The SmartFusion2™ Microcontroller Subsystem (MSS) includes a High Performance Direct Memory Access (HPDMA) controller which performs fast data transfer between any MSS memory connected to the AHB bus matrix and MSS DDR-Bridge memory. This software driver provides a set of functions for controlling the MSS HPDMA as part of a bare metal system where no operating system is available. The driver can be adapted for use as part of an operating system but the implementation of the adaptation layer between the driver and the operating system's driver model is outside the scope of the driver.

Features

The MSS HPDMA driver provides support for the following features:

- Initialization of the MSS HPDMA
- Starting and controlling HPDMA transfers
- Reading the status of a HPDMA transfer
- MSS HPDMA interrupt handling

The MSS HPDMA driver is provided as C source code.

Supported Hardware IP

The SmartFusion2 MSS HPDMA bare metal driver can be used with the SmartFusion2 MSS version 0.0.500 or higher.

Files Provided

The files provided as part of the MSS HPDMA driver fall into three main categories: documentation, driver source code and example projects. The driver is distributed via the Microsemi SoC Products Group's Firmware Catalog, which provides access to the documentation for the driver, generates the driver's source files into an application project, and generates example projects that illustrate how to use the driver. The Firmware Catalog is available from: www.microsemi.com/soc/products/software/firmwarecat/default.aspx.

Documentation

The Firmware Catalog provides access to these documents for the driver:

- User's guide (this document)
- Release notes

Driver Source Code

The Firmware Catalog generates the driver's source code into the *drivers/mss_hpdma* subdirectory of the selected software project directory. The files making up the driver are detailed below.

mss_hpdma.h

This header file contains the public application programming interface (API) of the MSS HPDMA software driver. This file should be included in any C source file that uses the MSS HPDMA software driver.

mss_hpdma.c

This C source file contains the implementation of the MSS HPDMA software driver.

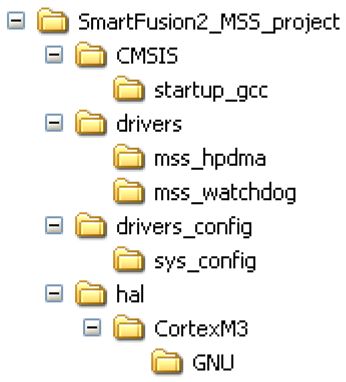
Example Code

The Firmware Catalog provides access to example projects illustrating the use of the driver. Each example project is self contained and is targeted at a specific processor and software toolchain combination. The example projects are targeted at the FPGA designs in the hardware development tutorials supplied with the SoC Products Group's development boards. The tutorial designs can be found on the Microsemi SoC Development Kit web page.

Driver Deployment

This driver is deployed from the Firmware Catalog into a software project by generating the driver's source files into the project directory. The driver uses the SmartFusion2 Cortex Microcontroller Software Interface Standard Hardware Abstraction Layer (CMSIS HAL) to access MSS hardware registers. You must ensure that the SmartFusion2 CMSIS HAL is included in the project settings of the software toolchain used to build your project and that it is generated into your project. The most up-to-date SmartFusion2 CMSIS HAL files can be obtained using the Firmware Catalog.

The following example shows the intended directory structure for a SoftConsole ARM® Cortex™-M3 project targeted at the SmartFusion2 MSS. This project uses the MSS HPDMA and MSS watchdog drivers. Both of these drivers rely on the SmartFusion2 CMSIS HAL for accessing the hardware. The contents of the *drivers* directory result from generating the source files for the driver into the project. The contents of the *CMSIS* and *hal* directories result from generating the source files for the SmartFusion2 CMSIS HAL into the project. The contents of the *drivers_config* directory are generated by the Libero project and must be copied into the into the software project.



```
SmartFusion2_MSS_project
    CMSIS
        startup_gcc
    drivers
        mss_hpdma
        mss_watchdog
    drivers_config
        sys_config
    hal
        CortexM3
            GNU
```

Figure 1 · SmartFusion2 MSS Project Example

Driver Configuration

The configuration of all features of the MSS HPDMA is covered by this driver. There are no dependencies on the hardware flow when configuring the MSS HPDMA.

The base address, register addresses and interrupt number assignment for the MSS HPDMA blocks are defined as constants in the SmartFusion2 CMSIS HAL. You must ensure that the latest SmartFusion2 CMSIS HAL is included in the project settings of the software tool chain used to build your project and that it is generated into your project.

Application Programming Interface

This section describes the driver’s API. The functions and related data structures described in this section are used by the application programmer to control the MSS HPDMA peripheral.

Theory of Operation

The MSS HPDMA driver functions are grouped into the following categories:

- Initialization of the MSS HPDMA driver and hardware
- Data transfer control
- Data transfer status
- Interrupt handling

Initialization

The MSS HPDMA driver is initialized through a call to the *MSS_HPDMA_init()* function. The *MSS_HPDMA_init()* function must be called before any other MSS HPDMA driver functions can be called.

Data Transfer Control

The driver provides the following functions to control HPDMA data transfers:

- *MSS_HPDMA_start()* – This function starts a HPDMA data transfer. You must provide the source and destination addresses along with transfer size and transfer direction.
- *MSS_HPDMA_pause()* – This function pauses the HPDMA transfer that is currently in progress.
- *MSS_HPDMA_resume()* – This function resumes a HPDMA transfer that was previously paused.
- *MSS_HPDMA_abort()* – This function aborts the HPDMA transfer that is currently in progress.

Data Transfer Status

The status of the HPDMA transfer initiated by the last call to *MSS_HPDMA_start()* can be retrieved using the *MSS_HPDMA_get_transfer_status()* function.

Interrupt Handling

The *MSS_HPDMA_set_handler()* function is used to register a handler function that will be called by the driver when the HPDMA transfer completes. The driver passes the outcome of the transfer to the completion handler in the form of a hpdma_status_t parameter indicating if the transfer was successful or the type of error that occurred during the transfer. You must create and register a transfer completion handler function to suit your application.

Types

hpdma_status_t

Prototype

```
typedef enum
{
    HPDMA_IN_PROGRESS,
    HPDMA_COMPLETED,
    HPDMA_SOURCE_ERROR,
    HPDMA_DESTINATION_ERROR,
    HPDMA_WORD_ALIGN_ERROR
} hpdma_status_t;
```

Description

The *hpdma_status_t* type is used to communicate the status of the HPDMA transfer initiated by the most recent call to *HPDMA_start()*. It indicates if a transfer is still currently in progress or the outcome of the latest transfer. This type is returned by the *MSS_HPDMA_get_transfer_status()* function and used as parameter to handler functions registered with the MSS HPDMA driver by the application.

Table 1 • HPDMA Transfer Status

Constant	Description
`HPDMA_IN_PROGRESS`	A HPDMA transfer is taking place.
`HPDMA_COMPLETED`	The most recent HPDMA transfer initiated by a call to *HPDMA_start()* has completed successfully.
`HPDMA_SOURCE_ERROR`	The most recent HPDMA transfer failed because of a problem with the source address passed to *HPDMA_start()*.
`HPDMA_DESTINATION_ERROR`	The most recent HPDMA transfer failed because of a problem with the destination address passed to *HPDMA_start()*.
`HPDMA_WORD_ALIGN_ERROR`	The most recent HPDMA transfer failed because the transfer size was not word aligned. This means that the transfer size passed to *HPDMA_start()* was not a multiple of four.

mss_hpdma_handler_t

Prototype

```
typedef void (*mss_hpdma_handler_t)(hpdma_status_t status);
```

Description

This type definition specifies the prototype of a function that can be registered with this driver as a HPDMA transfer completion handler function through a call to *MSS_HPDMA_set_handler()*. The HPDMA transfer completion handler will be called by the driver when a HPDMA transfer completes. The MSS HPDMA driver passes the outcome of the transfer to the completion handler in the form of a *hpdma_status_t* parameter indicating if the transfer was successful or the type of error that occurred during the transfer.

Constant Values

HPDMA_TO_DDR

The *HPDMA_TO_DDR* constant is used to specify the data transfer direction. It indicates a transfer from memory connected to the AHB bus matrix to MSS DDR-Bridge memory. It is used as a parameter by the *MSS_HPDMA_start()* function.

HPDMA_FROM_DDR

The *HPDMA_FROM_DDR* constant is used to specify the data transfer direction. It indicates a transfer from MSS DDR-Bridge memory to memory connected to the AHB bus matrix. It is used as a parameter by the *MSS_HPDMA_start()* function.

Functions

MSS_HPDMA_init

Prototype

```
void MSS_HPDMA_init(void);
```

Description

The *MSS_HPDMA_init()* function initializes the MSS HPDMA driver. It resets the HPDMA controller. It clears all pending HPDMA interrupts. It initializes internal data structures used by the MSS HPDMA driver. It disables interrupts related to HPDMA data transfers.

Parameters

This function has no parameters.

Return Value

This function does not return a value.

MSS_HPDMA_start

Prototype

```
void MSS_HPDMA_start
(
    uint32_t src_addr,
    uint32_t dest_addr,
    uint32_t transfer_size,
    uint8_t transfer_dir
);
```

Description

The *MSS_HPDMA_start()* function starts a HPDMA data transfer. Its parameters specify the source and destination addresses of the transfer as well as its size and direction. HPDMA data transfers always use DDR memory as the source or destination of the transfer. The HPDMA controller transfers data in 32-bit chunks located on 32-bit word aligned address boundaries.

Note: A call to MSS_HPDMA_start() while a transfer is in progress will not initiate a new transfer. Use the MSS_HPDMA_get_transfer_status() function or a completion handler registered by the MSS_HPDMA_set_handler() function to check the status of the current transfer before calling the MSS_HPDMA_start() function again.

Parameters

src_addr

The parameter *src_addr* is the address of the data that will be transferred by the HPDMA. This address must be 32-bit word aligned. The memory location specified by this address must be located in DDR memory if the *transfer_dir* parameter is set to *HPDM_FROM_DDR*.

dest_addr

The parameter *dest_addr* is the address of the location at which the data will be transferred. This address must be 32-bit word aligned. The memory location specified by this address must be located in DDR memory if the *transfer_dir* parameter is set to *HPDM_TO_DDR*.

transfer_size

The parameter *transfer_size* specifies the size in bytes of the requested transfer. The value of *transfer_size* must be a multiple of four.

transfer_dir

The parameter *transfer_dir* is used for specifying the data transfer direction. Allowed values for *transfer_dir* are as follows:

- HPDMA_TO_DDR
- HPDMA_FROM_DDR

Return Value

This function does not return a value.

MSS_HPDMA_pause

Prototype

```
void MSS_HPDMA_pause(void);
```

Description

The *MSS_HPDMA_pause()* function pauses the current HPDMA transfer. The HPDMA transfer can be resumed later by calling *MSS_HPDMA_resume()*.

Note: A HPDMA transfer can be paused by calling MSS_HPDMA_pause(). For a transfer that hasn't been completed yet, nor been terminated due to addressing errors, the MSS_HPDMA_get_transfer_status() function will return HPDMA_IN_PROGRESS as the status while the transfer is paused.

Parameters

This function has no parameters.

Return Value

This function does not return a value.

MSS_HPDMA_resume

Prototype

```
void MSS_HPDMA_resume(void);
```

Description

The *MSS_HPDMA_resume()* function resumes the current HPDMA transfer after it was previously paused through a call to *MSS_HPDMA_pause()*.

Note: A HPDMA transfer can be paused by calling MSS_HPDMA_pause(). For a transfer that hasn't been completed yet, nor been terminated due to addressing errors, the MSS_HPDMA_get_transfer_status() function will return HPDMA_IN_PROGRESS as the status while the transfer is paused.

Parameters

This function has no parameters.

Return Value

This function does not return a value.

MSS_HPDMA_abort

Prototype

```
void MSS_HPDMA_abort(void);
```

Description

The *MSS_HPDMA_abort()* function aborts the current HPDMA transfer.

Parameters

This function has no parameters.

Return Value

This function does not return a value.

MSS_HPDMA_get_transfer_status

Prototype

```
hpdma_status_t MSS_HPDMA_get_transfer_status(void);
```

Description

The *MSS_HPDMA_get_transfer_status()* function returns the status of the HPDMA transfer initiated by a call to *MSS_HPDMA_start()*.

Parameters

This function has no parameters.

Return Value

The *MSS_HPDMA_get_transfer_status()* function returns the status of the HPDMA transfer as a value of type *hpdma_status_t*. The possible return values are:

- HPDMA_IN_PROGRESS
- HPDMA_COMPLETED
- HPDMA_SOURCE_ERROR
- HPDMA_DESTINATION_ERROR
- HPDMA_WORD_ALIGN_ERROR

Example

The example code below demonstrates the use of the *MSS_HPDMA_get_transfer_status()* function to detect the completion of the transfer of the content of eNVM into MDDR memory.

```
void copy_envm_to_mddr(void)
    {
        MSS_HPDMA_start(ENVM_BASE_ADDR,
                        MDDR_BASE_ADDR,
                        ENVM_SIZE_BYTE,
                        HPDMA_TO_DDR);

        do {
            xfer_state = MSS_HPDMA_get_transfer_status();
        } while(HPDMA_IN_PROGRESS == xfer_state);
    }
```

MSS_HPDMA_set_handler

Prototype

```
void MSS_HPDMA_set_handler
(
    mss_hpdma_handler_t handler
);
```

Description

The *MSS_HPDMA_set_handler()* function is used to register a handler function that will be called by the driver when the HPDMA transfer completes. You must create and register a transfer completion handler function to suit your application. The MSS HPDMA driver passes the outcome of the transfer to the completion handler in the form of a *hpdma_status_t* parameter indicating if the transfer was successful or the type of error that occurred during the transfer if the transfer failed.

Parameters

handler

The handler parameter is a pointer to a handler function provided by your application. This handler is of type *mss_hpdma_handler_t*. The handler function must take one parameter of type *hpdma_status_t* and must not return a value.

Return Value

This function does not return a value.

Example

This example code demonstrates the use of the *MSS_HPDMA_set_handler()* function:

```
    void transfer_complete_handler(hpdma_status_t status);
    volatile uint32_t g_xfer_in_progress = 0;

    void demo_transfer(void)
    {
        MSS_HPDMA_init();

        MSS_HPDMA_set_handler(transfer_complete_handler);

        g_xfer_in_progress = 1;
        MSS_HPDMA_start((uint32_t)0x20000000,
                        (uint32_t)0x00000000,
                        20,
                        HPDMA_FROM_DDR);

        while(g_xfer_in_progress)
        {
            ;
        }
    }

    void transfer_complete_handler(hpdma_status_t status)
    {
        g_xfer_in_progress = 0;
        switch(status)
        {
            case HPDMA_COMPLETED:
                display("Transfer complete");
            break;

            case HPDMA_SOURCE_ERROR:
                display("Transfer failed: source error");
            break;

            case HPDMA_DESTINATION_ERROR:
                display("Transfer failed: destination error");
            break;

            case HPDMA_WORD_ALIGN_ERROR:
                display("Transfer failed: word alignment error");
            break;

            default:
                display("Unexpected status");
            break;
        }
    }
```

Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call **800.262.1060**
From the rest of the world, call **650.318.4460**
Fax, from anywhere in the world **408.643.6913**

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Microsemi SoC Products Group Customer Support website for more information and support (http://www.microsemi.com/soc/support/search/default.aspx). Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on website.

Website

You can browse a variety of technical and non-technical information on the Microsemi SoC Products Group home page, at http://www.microsemi.com/soc/.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

50200411-2/09.15