SmartFusion cSoC: Accessing External Memories Using the External Memory Controller

Table of Contents

Introduction

The SmartFusion® customizable system-on-chip (cSoC) device contains a hard embedded microcontroller subsystem (MSS), programmable analog circuitry and FPGA fabric consisting of logic tiles, static random access memory (SRAM), and phase-locked loops (PLLs). The MSS consists of 100 MHz ARM® Cortex™-M3 processor, advanced high-performance bus (AHB) matrix, system registers, Ethernet MAC, DMA engine, real-time counter (RTC), embedded nonvolatile memory (eNVM), embedded SRAM (eSRAM), fabric interface controller (FIC), the Philips Inter-Integrated Circuit (I^2C), serial peripheral interface (SPI) peripherals, and external memory controller (EMC).

The EMC provides a glueless interface to the external memory devices that can be addressed by the ARM Cortex-M3 processor or user logic in the FPGA fabric. The EMC is divided into two regions: Region 0 and Region1. Region 0 and Region 1 can be independently configured for three memory types supported by EMC (NOR FLASH, Asynchronous RAM, and Synchronous RAM). The MSS configuration tool provides a graphical user interface (GUI) to configure the memory type, port size, and latency.

Refer to the *SmartFusion Microcontroller Subsystem User's Guide* for more details on EMC.

This application note describes how to access external memories (SRAM and NOR Flash) using EMC on the SmartFusion Development Kit Board. This document also describes the detailed description of EMC configuration. A basic understanding of the SmartFusion design flow is assumed.

Refer to *Using UART with SmartFusion cSoC - Libero SoC and SoftConsole Flow Tutorial* to understand the SmartFusion design flow.

Design Example Overview

This design example demonstrates accessing the external SRAM and NOR flash using EMC on the SmartFusion Development Kit Board. It uses one UART peripheral and EMC in the MSS. The EMC connects to Cypress SRAM, CY7C1061DV33-10ZSXI, at region 0 and Numonyx NOR flash, JS28F640J3D-75, at region 1 in the SmartFusion Development Kit Board. The UART prints the memory locations and data on HyperTerminal. Figure 1 shows the system-level block diagram.

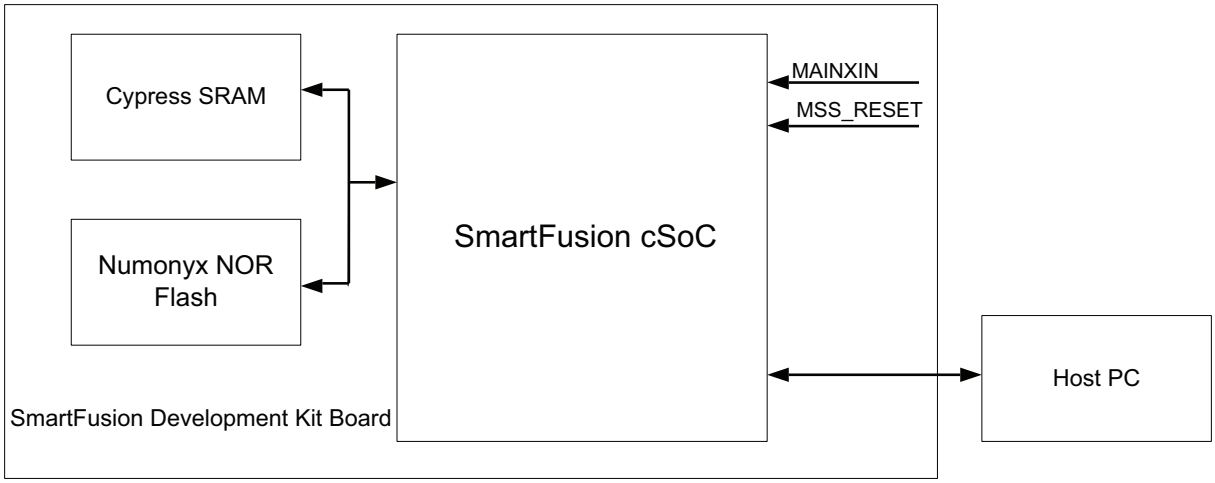

Figure 1 • **System Level Block Diagram**

Description of the Design Example

This design example mainly consists of EMC and a UART. Refer to the "Microcontroller Subsystem Configuration" on page 4 for a detailed description of EMC configuration.

The EMC is mapped into system address space from 0x70000000 to 0x77FFFFFF. The EMC memory space is divided into an upper and lower half. Each half memory space can be connected to a separate external memory device (EMD) and supports up to 64 MBytes of memory. The addresses to the EMDs are common. Access to each half memory space is determined by the assertion of the chip select signals (EMC_CS0_N and EMC_CS1_N). Table 1 illustrates the starting and ending address for each chip select.

Table 1 • **EMC Memory Regions**

Chip Select	Starting Address	Ending Address
EMC_CS0_N - Lower half (Region 0)	0x70000000	0x73FFFFFF
EMC_CS1_N - Upper half (Region 1)	0x74000000	0x77FFFFFF

This design example uses two 16-Mbit SRAM at region 0 and two 64-Mbit NOR flash at region 1 on the SmartFusion Development Kit.

Refer to the *SmartFusion Development Kit User's Guide*, the *Cypress CY7C1061DV33-10ZSXI* datasheet available at www.cypress.com, and the Numonyx JS28F640J3D-75 datasheet available at www.micron.com for more information on the SmartFusion Development Kit Board, SRAM, and NOR Flash.

The software design consists of setting the base address for SRAM and NOR flash, memory write and read operations, and printing the memory locations and data on HyperTerminal. It uses following functions in the main.c file.

```
static void emc_sram_wr(void);
static void emc_nor_wr(void);
```

The following sections explain the SRAM and NOR flash write and read operations.

SRAM Write and Read Operations

The following code sets the base address of EMC SRAM to 0x70000000.

```
/* Setting the base address for EMC SRAM*/
#define EXT_RAM_BASE_ADDR 0x70000000
```

The design example uses the following code for external SRAM write and read operations. It fills the first 16 memory locations with 0xFFFFFFFF, and then overwrites the first 10 memory locations with a pattern. It then reads back the first 16 location and prints the memory locations and data on HyperTerminal.

```
unsigned long * buff = (unsigned long *)EXT_RAM_BASE_ADDR;
int size;

/* Writing 0xFFFFFFFF to first 16 Memory locations*/
for ( size = 0; size < 16; size++)
{
     buff [size] = 0xFFFFFFFF;
}

 /* Writing 0, 4, 8, C, 10.. pattern to first 10 Memory locations*/
for( size = 0; size < 10; size++)
{
     buff[size] = size*4;
}

/* Reading data from first 16 Memory locations and prints on HyperTerminal*/
for( size = 0; size < 16; size++)
{
     printf("%p  ->   0x%08x \n\r", buff+size, ( unsigned int)buff[size]);
}
```

NOR Flash Write and Read Operations

The following code sets the base address of EMC NOR Flash to 0x74000000.

```
/* Setting the base address for EMC NOR Flash*/
#define EXT_FLASH_BASE_ADDR 0x74000000
```

The NOR Flash write and read operations mainly consist of four steps, as follows:

1. EMC Initialization
2. Erasing the memory content
3. NOR flash write
4. NOR flash read

The design example uses the following software driver APIs for NOR flash write and read operations.

emc_init ()

This function initializes the EMC controller with proper wait states for read and write access.

emc_flash_chip_erase ()

This function erases the content of the external NOR flash from the address provided to this function until the last address of the NOR flash.

emc_flash_write()

This function writes the content of the buffer passed as a parameter to external NOR flash. The data is written from the memory location specified by the first parameter. This address is the absolute address in the processor's memory space at which the external flash is located.

emc_flash_read()

This function reads content of the external NOR flash. The data is read from the memory location specified by the first parameter. This address is the absolute address in the processor's memory space at which the external flash is located.

Refer to "Appendix B – NOR Flash Driver APIs" on page 8 for more details on NOR flash driver APIs used in this example design to access NOR Flash Memory.

A Verilog Libero SoC project and SoftConsole project are provided in the design files attached with this design example.

Microcontroller Subsystem Configuration

The MSS is configured with one UART peripheral (UART_0) and EMC. The clock conditioning circuit (CCC) in the MSS generates an 80 MHz clock and acts as clock source for the peripherals.

The EMC is divided into two regions: Region 0 and Region1. The MSS configuration tool provides a graphical user interface to configure the Memory type, Port size and Latency for each region. The EMC Read/Write Latency values can be configured as per the user requirements. The minimum latency values at different MSS clock frequencies for the SmartFusion Development Kit Board are given in Table 2.

Table 2 • **EMC Read/Write Minimum Latency Values**

Parameters	MSS Clock – 100 MHz		MSS Clock – 80 MHz	
	Asynchronous RAM	NOR flash	Asynchronous RAM	NOR flash
Read Latency for First Access (HCLK cycles)	1	5	1	4
Read Latency for Remaining Accesses (HCLK cycles)	1	1	1	1
Write Latency (HCLK cycles)	0	0	0	0

Figure 2 shows the timing diagram of EMC for Asynchronous Read. For more detail on Read/Write latencies, refer to the *SmartFusion Microcontroller Subsystem User's Guide*.

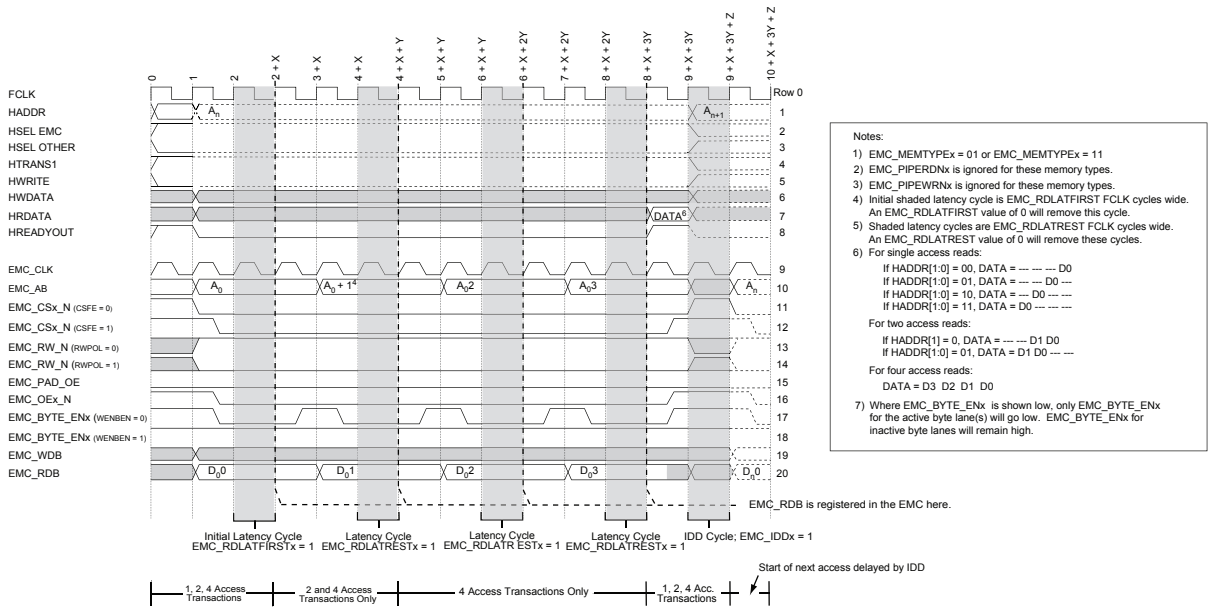

Figure 2 • **Timing Diagram of EMC for Asynchronous Read**

Figure 3 shows the EMC configuration for Region 0, which is configured as Asynchronous RAM and port size as half word. Similarly, Figure 4 shows the EMC configuration for Region 1, which is configured as NOR flash and port size as half word. Refer to the *EMC Configuration User's Guide* for more details on EMC configuration.

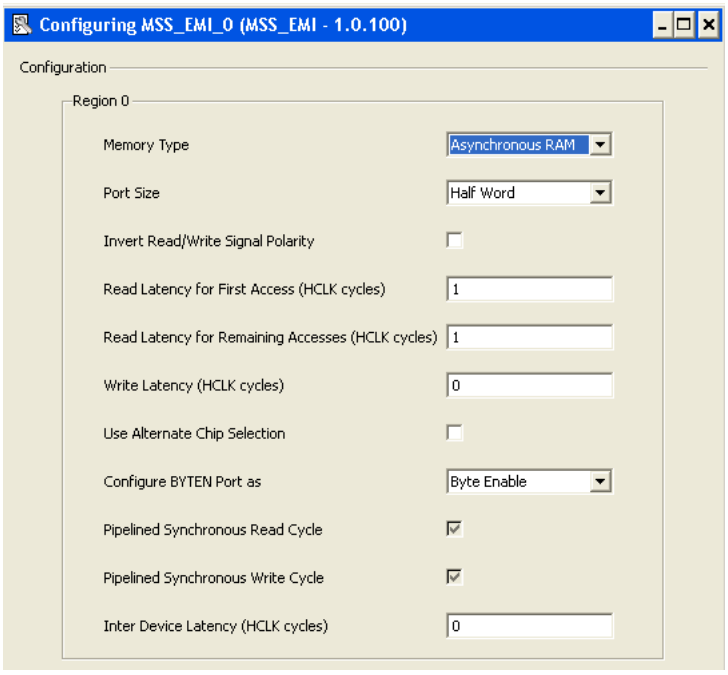

Figure 3 • **EMC Configuration for Region 0**

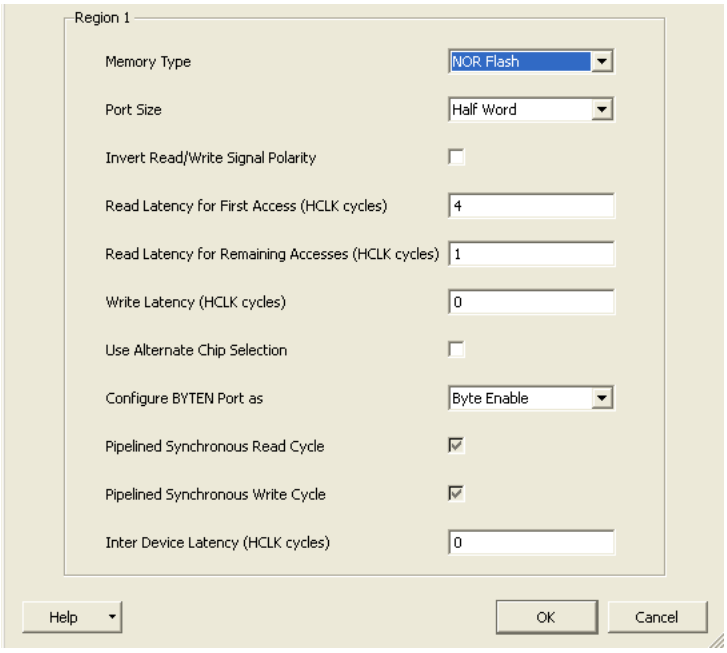

Figure 4 • **EMC Configuration for Region 1**

Board Specific Settings

Table 3 gives the jumper settings needed to access the SRAM and NOR flash.

Table 3 • **Jumper Settings to Interface EMC with SRAM and NOR Flash**

Jumper	Pin	Pin
JP17	2	3
JP19	2	3
JP24	1	2
JP16	2	3

Refer to the *SmartFusion Development Kit User's Guide* for more details.

Running the Design

Program the SmartFusion Development Kit Board with the generated/provided STP file (refer to "Appendix A – Design and Programming Files" on page 7) using FlashPro and then power cycle the board.

Invoke the SoftConsole IDE (refer to "Appendix A – Design and Programming Files" on page 7) by clicking **Write Application code** under Develop Firmware in Libero SoC and launch the debugger. Start a HyperTerminal session with 57,600 baud rate, 8 data bits, 1 stop bit, no parity, and no flow control. If your computer does not have the HyperTerminal program, use any free serial terminal emulation program such as PuTTY or Tera Term. Refer to the *Configuring Serial Terminal Emulation Programs* tutorial for configuring HyperTerminal, Tera Term, and PuTTY.

When you run the debugger in SoftConsole, the application starts printing the memory location and data on HyperTerminal. Figure 5 shows the screen shot of HyperTerminal with SRAM memory location and data.

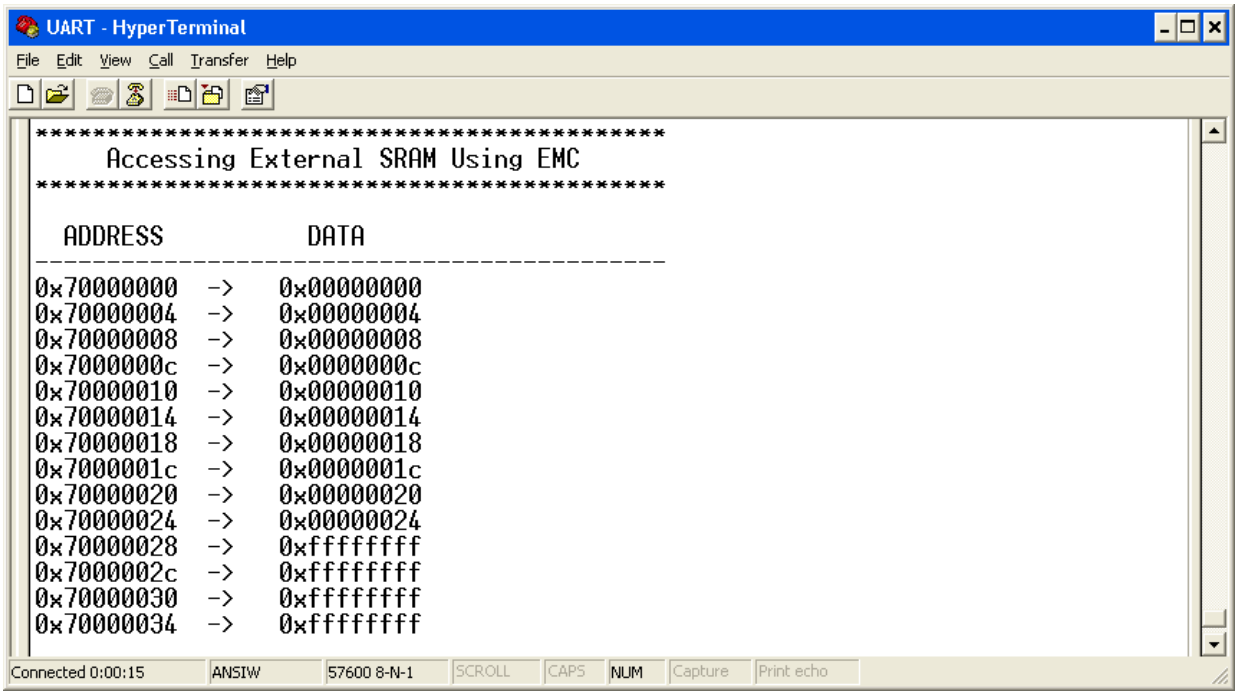

Figure 5 • **Screen Shot of HyperTerminal with SRAM Memory Location and Data**

Figure 6 shows the screen shot of HyperTerminal with NOR Flash memory location and data.

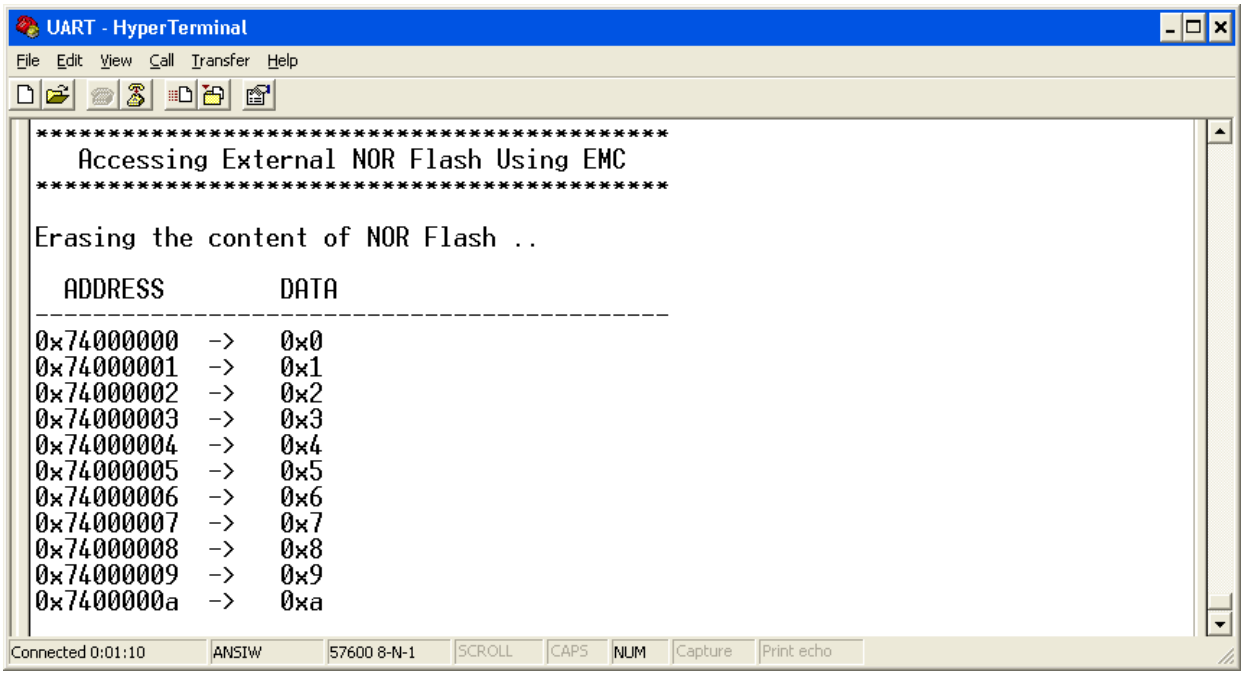

Figure 6 • **Screen Shot of HyperTerminal with NOR Flash Memory Location and Data**

Release Mode

The release mode programming file (STAPL) is also provided. Refer to the Readme.txt file included in the programming zip file for more information. Refer to the *Building Executable Image in Release Mode and Loading into eNVM* tutorial for more information on building an application in release mode.

Conclusion

The design example demonstrates accessing the external SRAM and NOR flash using EMC on the SmartFusion Development Kit Board. This document also describes the detailed description of EMC configuration and NOR flash driver APIs.

Appendix A – Design and Programming Files

You can download the design files from the Microsemi SoC Products Group website:

www.microsemi.com/soc/download/rsc/?f=A2F_AC348_DF.

The design file consists of Libero SoC Verilog, SoftConsole software project, and programming files (*.stp) for A2F500-DEV-KIT. Refer to the Readme.txt file included in the design file for the directory structure and description.

You can download the programming files (*.stp) in release mode from the Microsemi SoC Products Group website: www.microsemi.com/soc/download/rsc/?f=A2F_AC348_PF.

The programming zip file consists of a STAPL programming file (*.stp) for A2F500-DEV-KIT and a Readme.txt file.

Appendix B – NOR Flash Driver APIs

This section describes the software driver APIs used in this design to carry out transactions with NOR flash. These drivers are included in the design files with this design example.

Function Description of Data Structures

Enum : emc_flash_status_t

This enum represents the status of the different APIs, such as emc_flash_write, emc_flash_read, emc_flash_chip_erase, and emc_flash_block_erase.

```
typedef enum {
    NOR_SUCCESS = 0,
    NOR_BLOCK_LOCK_ERROR = 0x2,
    NOR_PROGRAM_SUSPEND = 0x4,
    NOR_GENERAL_ERROR = 0x08,
    NOR_PROGRAM_ERROR = 0x10,
    NOR_ERASE_ERROR   = 0x20,
    NOR_CMD_SEQ_ERROR = 0x30,
    NOR_ERASE_SUSPEND = 0x40,
    NOR_INVALID_ARGUMENTS,
    NOR_INVALID_ADDRESS,
    NOR_UNSUCCESS};
```

Function Description of Application Programming Interface (API)

emc_Init()

This function initializes the EMC controller with proper wait states for read and write access.

For example:

emc_init();

emc_flash_chip_erase (uint32_t start_addr)

This function erases the content of the external NOR flash from the address provided to this function until the last address of the NOR flash. For example:

```
emc_flash_chip_erase(0x74000000);
```

*emc_flash_block_erase (uint16_t *blockAddr)*

This function erases the content of a particular block of the external NOR flash based on the address provided to this function. For example:

```
emc_flash_block_erase (0x74000000)
```

*emc_flash_read(uint32_t start_addr, uint8_t * p_data, size_t nb_bytes)*

This function reads content of the SmartFusion EMC external NOR flash. The data is read from the memory location specified by the first parameter. This address is the absolute address in the processor's memory space at which the external flash is located.

- @param start_addr: This is the address at which data will be read. This address is the absolute address in the processor's memory space at which the external flash is located.
- @param p_data: This is a pointer to the buffer for holding the read data.
- @param nb_bytes: This is the number of bytes to be read from external flash.
- @return: The return value indicates if the read was successful. The possible values are:
  - NOR_SUCCESS: Describes that the NOR flash operation is correct and complete
  - NOR_INVALID_ADDRESS: Describes that the function has received an invalid address
  - NOR_UNSUCCESS: Describes that the NOR flash operation is incomplete

For example:

```
status = emc_flash_read(0x74000000 + ii, output_buffer, length);
```

*emc_flash_write (uint32_t start_addr, const uint8_t * p_data, size_t nb_bytes)*

This function writes the content of the buffer passed as a parameter to the external NOR flash. The data is written from the memory location specified by the first parameter. This address is the absolute address in the processor's memory space at which the external flash is located.

- @param start_addr: This is the address at which data will be written. This address is the absolute address in the processor memory space at which the external flash is located.
- @param p_data: This is a pointer to the buffer holding the data to be written into external flash.
- @param nb_bytes: This is the number of bytes to be written into external flash.
- @return: The return value indicates if the write was successful. The possible values are:
  - NOR_SUCCESS: Describes the NOR flash operation is correct and complete
  - NOR_INVALID_ADDRESS: Describes that function has received an invalid address
  - NOR_UNSUCCESS: Describes the NOR flash operation is incomplete

For example:

```
status = emc_flash_write(0x74000000 + ii, input_buffer, length);
```

List of Changes

The following table lists critical changes that were made in each revision of the document.

Revision*	Changes	Page
Revision 3 (February 2012)	Removed ".zip" extension in the Design and Programming files link (SAR 36763).	7
Revision 2 (January 2012)	Modified last two lines in "Introduction" section (SAR 35788).	1
	Modified first five lines in "Running the Design" section (SAR 35788).	6
	Added a new section called "Release Mode" (SAR 35788).	7
	Modified "Appendix A – Design and Programming Files" section (SAR 35788).	7
Revision 1 (August 2010)	Design files for this application note have been modified to support Libero SP2 and A2F500 based development board. Also removed and added few lines on page 6 under "Running the Design" (SAR 27474).	6
	Removed and added few lines on page 8 under "Appendix A – Design and Programming Files" (SAR 27474).	7
	Removed Table 4 on page 8 (SAR 27474).	8

*Note: *The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.*

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

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

51900216-3/02.12