# SmartFusion cSoC: Accessing EEPROM Using I$^2$C

## 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 a 100 MHz ARM® Cortex™-M3 processor, advanced high-performance bus (AHB) matrix, system registers, Ethernet MAC, DMA engine, real-time clock (RTC), embedded nonvolatile memory (eNVM), embedded SRAM (eSRAM), fabric interface controller (FIC), the Philips Inter-Integrated Circuit (I$^2$C), serial peripheral interface (SPI), and external memory controller (EMC).

The MSS has two identical I$^2$C peripherals that perform serial-to-parallel conversion on data originating from serial devices, and perform parallel-to-serial conversion on data from the ARM Cortex-M3 processor to these devices. The Cortex-M3 embedded processor controls the I$^2$C peripherals via the Advanced Peripheral Bus (APB) interface.

The I$^2$C peripherals in the SmartFusion cSoC device support I$^2$C, SMBus, and PMBus data transfers, which conform to the I$^2$C v2.1 specification and support the SMBus v2.0 and PMBus v1.1 specifications. The I$^2$C peripherals can operate as either a Master or a Slave. When operating in the Master mode, the I$^2$C peripherals generate the serial clock and data to the Slave device that needs to be accessed. The I$^2$C peripherals can generate serial clock from 104 KHz to 1.66 MHz by dividing MSS clock, which can be controlled by software. The I$^2$C peripherals use a 7-bit addressing format and run up to 400 Kbps (Fast mode) data rates nominally. Faster rates can be achieved depending on the external load.

Refer to the *SmartFusion Microcontroller Subsystem User's Guide* for more details on I$^2$C peripherals.

This application note describes how to use the I$^2$C interface on the SmartFusion cSoC device to access EEPROM. A basic understanding of SmartFusion design flow is assumed. Refer to the *Using UART with a SmartFusion cSoC - Libero SoC and SoftConsole Flow Tutorial* to understand the SmartFusion design flow.

## Design Example Overview

This design example demonstrates how to use I$^2$C EEPROM on the SmartFusion Development Kit Board. This kit has one EEPROM, ST M24512-WMN6TP, which is connected to I2C_1 on the SmartFusion Development Kit Board. Figure 1 shows the top level interface signals used in this design example. The UART in the MSS acts as a user interface for writing/reading string data into I$^2$C EEPROM through HyperTerminal.

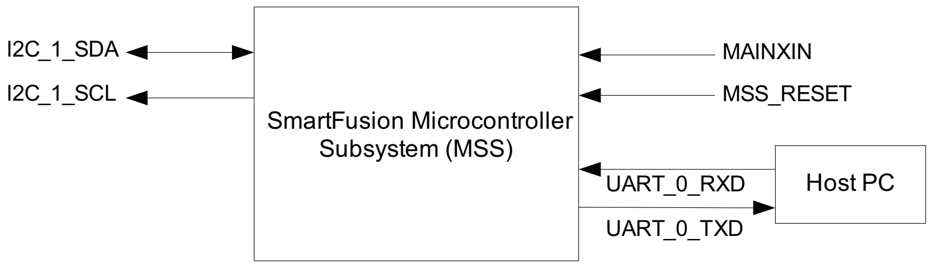


***Figure 1* • Top Level Interface Signals**

## Design Example Description

The MSS is configured to use one I$^2$C interface (I2C_1) and one UART interface (UART_0). I2C_1 is clocked by PCLK1 on APB bus 1 and UART_0 is clocked by PCLK0 on APB bus 0. PCLK0 and PCLK1 are derived from the clock conditioning circuit (CCC) in the MSS that generates an 80 MHz clock.

The I$^2$C peripheral in the MSS is configured as Master which operates in Fast mode. The APB bus clock (PCLK1) divider is set to 256 to generate 312.5 KHz serial clocks (I2C_1_SCL).

The I$^2$C Master device initiates a read/write transaction by sending a START bit as soon as the bus becomes free. The START bit is followed by the 7-bit serial address of the slave device and read/write bit. The slave acknowledges receipt of its address with an acknowledge bit. When master is in Write mode, the master sends data one byte at a time to the slave. When master is in Read mode, the slave sends data one byte at a time to the master.

This design example uses 512 Kbit EEPROM with serial address (device select code) set to "A0" on the SmartFusion Development Kit Board. The device select code consists of a 4-bit device type identifier (1010b), and a 3-bit Chip Enable "Address" (E2, E1, and E0). These chip enable pins are connected to ground (000b) in the SmartFusion Development Kit Board.

Refer to the *SmartFusion Development Kit User's Guide* and the ST *M24512-WMN6TP* datasheet for more information on the development kit board and EEPROM specification.

The design example uses UART as a user interface for writing string data into I$^2$C EEPROM and reading string data from I$^2$C EEPROM on HyperTerminal. You can enter maximum of 1 Kbyte string data through HyperTerminal. The application writes the string data into the I$^2$C EEPROM and reads string data from EEPROM after the write operation that prints on the HyperTerminal.

Figure 2 on page 3 shows the string data flow from HyperTerminal to I$^2$C EEPROM and from I$^2$C EEPROM to HyperTerminal.

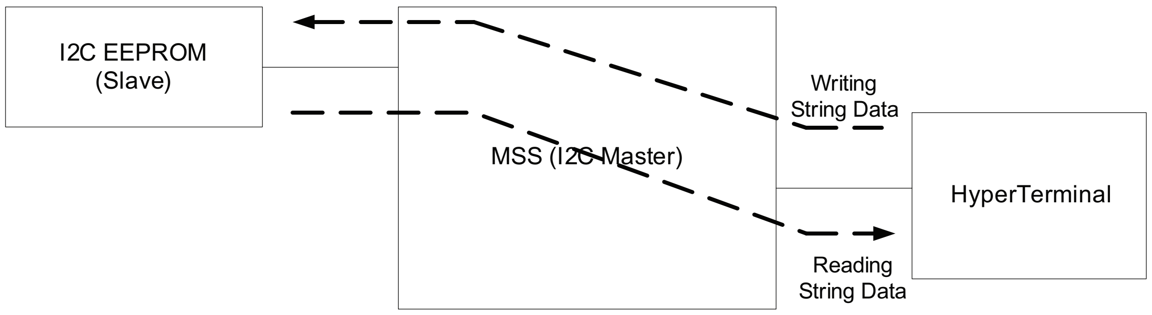


***Figure 2* • I2C EEPROM Read/Write Flow**

Libero® System-on-Chip (SoC) projects are provided in the design files attached with this design example.

## Interface Description

Table 1 shows the top level interface signal descriptions.

***Table 1* • Interface Description**

| Signal | Direction | Description |
|---|---|---|
| MSS_RESET_N | Input | Active low reset signal for the MSS. |
| MAINXIN | Input | Main crystal oscillator circuit.<br>Input to the crystal oscillator circuit. Pin for connecting an external crystal, ceramic resonator, or RC network. |
| UART_0_TXD | Output | UART Transmit data |
| UART_0_RXD | Input | UART Receive data |
| I2C_1_SDA | Input | Serial data |
| I2C_1_SCL | Output | Serial clock |

## Software Implementation

The following EEPROM driver APIs are used in the example design to access I$^2$C EEPROM.

### *EEPROM_init ()*

This function initializes and configures the I$^2$C peripheral for data transfer. It configures the I$^2$C controller with serial clock speed and EEPROM Slave address for EEPROM.

### *EEPROM_write ()*

This function writes the content of the buffer passed as parameter to EEPROM. The data is written to the EEPROM based on the start address and number of bytes specified. This address ranges from 0 to EEPROM size and not the processors absolute range.

### *EEPROM_read ()*

This function reads the content to the buffer passed as parameter to EEPROM. The data is read from the EEPROM based on the start address and number of bytes specified. This address ranges from 0 to EEPROM size and not the processors absolute range.

Refer to "Appendix B - $I^2C$ EEPROM Driver APIs" on page 6 for more details on $I^2C$ initialization and configuration, and EEPROM write and read operations.

### *Macro Settings*

The following macros to be used in EEPROM API file (EEPROM.h) to enable the communication between $I^2C$ peripheral and EEPROM.

- # define I2C_INSTANCE: This macro decides either I2C_0 or I2C_1 needs to be communicated with EEPROM.
- # define EEPROM_SLAVE_ADDRESS: This macro defines the serial address (device select code) of EEPROM device.
- # define EEPROM_I2C_CLK_FREQ: This macro defines the APB bus clock divider.
- # define PAGE_SIZE_IN_BYTES: This macro defines the maximum page size in bytes. It depends on EEPROM device. The EEPROM on the SmartFusion Development Kit Board allows up to or 128 bytes to be written in a single Write cycle.

Refer to the *SmartFusion MSS Firmware Drivers v2.0 User's Guide* for more details on the $I^2C$ driver's API.

# Running the Design

## Board Settings

The design example works on the SmartFusion Development Kit Board with default board settings. Refer to the following user’s guide for default board settings.

- *SmartFusion Development Kit User’s Guide*

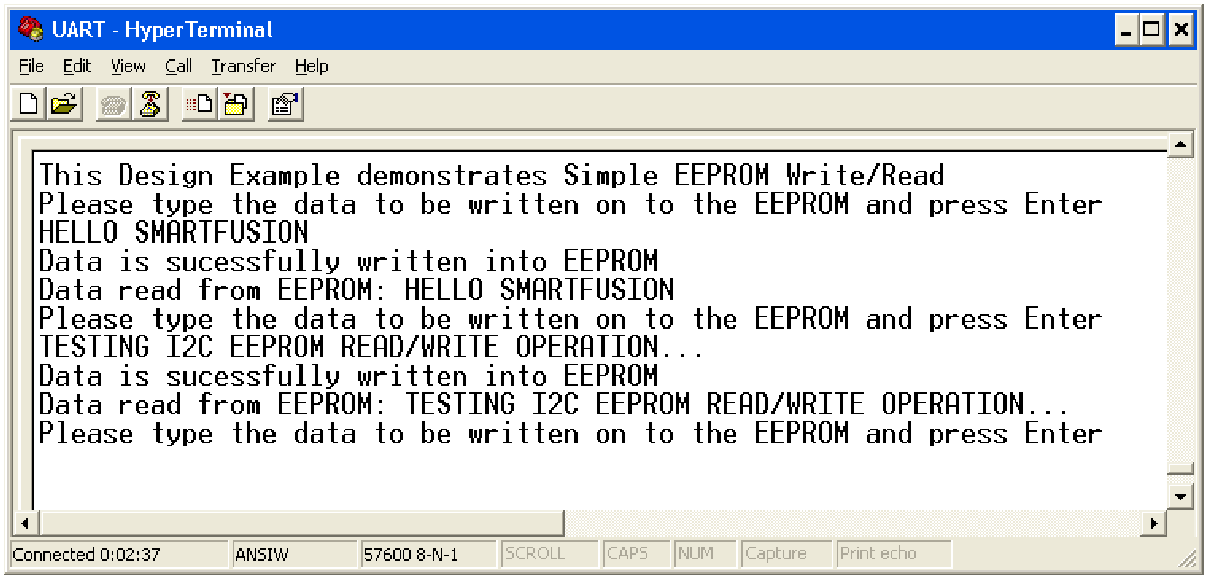
## Program the Design and Running the Application

Program the SmartFusion Development Kit Board with the generated/provided *.stp file (refer to "Appendix A – Design Files" on page 5) using FlashPro and then power cycle the board.

Invoke the SoftConsole IDE by clicking on the **Write Application code** under **Develop Firmware** in Libero SoC (refer to "Appendix A – Design Files" on page 5) and launch the debugger. Start HyperTerminal with 57600 baud rate, 8 data bits, 1 stop bit, no parity, and no flow control. If your computer does not have HyperTerminal program, use any free serial terminal emulation program like PuTTY or Tera Term. Refer to the *Configuring Serial Terminal Emulation Programs* tutorial for configuring the HyperTerminal, Tera Term, and PuTTY.

When you run the debugger in SoftConsole, the HyperTerminal window asks you to enter the string data to be written into the $I^2C$ EEPROM.

Once you enter the string data, the application reads data from EEPROM and prints on the HyperTerminal. Figure 3 shows the screen shot of HyperTerminal with I$^2$C EEPROM read/write operation.



```
This Design Example demonstrates Simple EEPROM Write/Read
Please type the data to be written on to the EEPROM and press Enter
HELLO SMARTFUSION
Data is sucessfully written into EEPROM
Data read from EEPROM: HELLO SMARTFUSION
Please type the data to be written on to the EEPROM and press Enter
TESTING I2C EEPROM READ/WRITE OPERATION...
Data is sucessfully written into EEPROM
Data read from EEPROM: TESTING I2C EEPROM READ/WRITE OPERATION...
Please type the data to be written on to the EEPROM and press Enter
```

***Figure 3* • Screen Shot of HyperTerminal**

### Release Mode

The release mode programming file (STAPL) is also provided. Refer to the Readme.txt file included in the programming 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

This application note introduces the features of SmartFusion cSoC FPGA devices and highlights the features of I$^2$C peripherals. The design example demonstrated the usage of I$^2$C peripherals on SmartFusion cSoC to access EEPROM with sample software. It also explains the usage of I$^2$C EEPROM driver APIs to implement EEPROM write/read operations.

## Appendix A – Design Files

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

www.microsemi.com/soc/download/rsc/?f=A2F_AC345_DF.

The design file consists of Libero SoC Verilog, SoftConsole software project, and programming files (*.stp) for A2F500-DEV-KIT and A2F-EVAL-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_AC345_PF.

The programming file consists of STAPL programming file (*.stp) for A2F500-DEV-KIT and A2F-EVAL-KIT, and a Readme.txt file.

# Appendix B - I$^2$C EEPROM Driver APIs

This section describes the software driver APIs used in this design to carry out transactions with I$^2$C EEPROM. These drivers are included in the design files with this design example.

## Function Description Of Data Structures

### *Enum : typedef enum*

This enum represents the status of the EEPROM Read/Write. EEPROM_ADDRESS_FAULT represents the address fault if the user provided address is out of range of the EEPROM size.

```
typedef enum
{
    EEPROM_WRITE_SUCCESS = 0,
    EEPROM_READ_SUCCESS,
    EEPROM_WRITE_UNSUCCESS,
    EEPROM_READ_UNSUCCESS,
    EEPROM_ADDRESS_FAULT
} EEPROM_status_t;
```

## Function Description Of Application Programming Interface (API)

### *void EEPROM_init (void);*

This function initializes the I2C_0 and I2C_1 for data transfer, with the provided I$^2$C clock division factor, the slave address, and the I$^2$C instance as macros.

### *EEPROM_status_t EEPROM_write (uint16_t start_address, uint16_t size_in_bytes, uint8_t * write_buffer);*

This function performs a write operation to the EEPROM based on the start address passed as first parameter and the number of bytes passed as second parameter. If the number of bytes to be written is more than 128 bytes, this function divides the write buffer into pages, each of 128 bytes size. Then, from the specified start address, it writes page-by-page to the EEPROM.

- @param start_address: The start_address parameter specifies the start address of the EEPROM, where data has to be written.
- @param size_in_bytes: The size_in_bytes parameter specifies the number of bytes to be written to EEPROM from write_buffer.
- @param write_buffer: The write_buffer parameter is a pointer to the buffer from where the data has to be transmitted.

For example:

```
EEPROM_write(start_address,300,write_data);
```

### *EEPROM_status_t EEPROM_read (uint16_t start_address, uint16_t size_in_bytes, uint8_t * read_buffer);*

This function reads the content from the EEPROM. The data is read from the memory location specified by the first parameter and number of bytes specified by the second parameter. This address ranges from 0 to EEPROM size and not the processors absolute range.

- @param start_address: The start_address parameter specifies the start address of the EEPROM, from where the data has to be received.
- @param size_in_bytes: The size_in_bytes parameter specifies the number of bytes to be read from EEPROM to read_buffer.
- @param write_buffer: The write_buffer parameter is a pointer to the buffer from where the data has to be transmitted.

For example:

```
EEPROM_read(start_address,300,read_data);
```

### *void delay_btw_transfers (volatile uint32_t n);*

This function provides the delay between two I$^2$C operations, that is, delay between stop and start conditions. This delay can be 2 to 3 cycles of I$^2$C serial clock.

Note: The summation of start_address and size_in_bytes should be in the range of 1 to 65536. The start_address should be in the range of 0 to 65536 and the size_in_bytes should be in the range of 1 to 65536.

## List of Changes

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

| **Revision*** | **Changes** | **Page** |
|---|---|---|
| Revision 4 (January 2013) | Added "Board Settings" section and "Program the Design and Running the Application" section (SAR 43469). | 4 |
| Revision 3 (February 2012) | Removed ".zip" extension in the Design files link (SAR 36763). | 5 |
| Revision 2 (January 2012) | Modified the "Running the Design" section (SAR 35799). | 4 |
| | Added the "Release Mode" section (SAR 35799). | 5 |
| | Modified the "Appendix A – Design Files" section (SAR 35799). | 5 |
| Revision 1 (August 2010) | Modified the section "Running the Design" (SAR 27471). | 4 |
| | Modified the section "Appendix A – Design Files" (SAR 27471). | 5 |
| | Removed Table 2 • Design Files (SAR 27471). | |

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

51900215-4/01.13