SmartFusion2 I^{2}C Reference Design using Multiple Masters and Multiple Slaves - Libero SoC v11.7

Table of Contents

Purpose

SmartFusion®2 system-on-chip (SoC) field programmable gate array (FPGA) device contains two Philips inter-integrated circuit (I^{2}C) peripherals available in the microcontroller subsystem (MSS). In addition, a number of I^{2}C peripherals can be implemented in the FPGA fabric using CoreI2C IP. This application note describes the I^{2}C transaction types (Write, Read, and Write-Read) with a reference design, which implements multiple Masters and Slaves using the SmartFusion2 Security Evaluation Kit.

Introduction

I^{2}C is a two-wire serial bus interface that provides data transfer between several devices. The MSS has two identical I^{2}C peripherals that perform serial-to-parallel data conversion originating from the serial devices, and parallel-to-serial conversion of data from the ARM® Cortex®-M3 processor. The Cortex-M3 embedded processor controls the I^{2}C peripherals through the advanced peripheral bus (APB) interface.

The I^2C peripherals in the SmartFusion2 SoC FPGA device support I^2C, system management bus (SMBus), and power management bus (PMBus) data transfers, which conform to the I^2C v2.1 specifications and support the SMBus v2.0 and PMBus v1.1 specifications. The I^2C peripherals can operate as either a Master or a Slave, and can be configured independently. When operating in Master mode, the I^2C peripherals generate the serial clock and data to the Slave device that needs to be accessed. The I^2C peripheral generates the serial clock by dividing MSS clock which is controlled by a software. The I^2C peripherals use a 7-bit addressing format and run up to 400 kbps (Fast mode) data rates. Faster rates can be achieved depending on the external load. For more details about I^2C peripherals, see the *UG0331: SmartFusion2 Microcontroller Subsystem User Guide*.

If the system requires more than two I^2C peripherals, additional I^2C peripherals have to be implemented in the FPGA fabric. Microsemi provides CoreI2C IP to fulfill the design requirement. CoreI2C is available in the Libero® System-on-Chip (SoC) IP catalog.

This application note describes the I^2C transaction types with a reference design which implements two Masters and two Slaves using the SmartFusion2 Security Evaluation Kit. MSS I^2C0 and CoreI2C_0 are configured as I^2C MASTER1 and I^2C MASTER2. The MSS I^2C1 and CoreI2C_1 are configured as I^2C SLAVE1 and I^2C SLAVE2 as shown in Figure 1. The reference design package has a graphical user interface (GUI) that runs on a host PC to communicate with the SmartFusion2 Security Evaluation Kit board. The GUI allows the user to select the Master and Slave combinations, serial clock, Slave addresses, number of bytes to read, and the I^2C transaction types. To communicate between the Masters and Slaves, MSS I^2C0 SDA, MSS I^2C1 SDA, CoreI2C_0 SDA, and CoreI2C_1 SDA are connected together, and MSS I^2C0 SCL, MSS I^2C1 SCL, CoreI2C_0 SCL, and CoreI2C_1 SCL are connected together on the SmartFusion2 Security Evaluation Kit board.

Note: SDA: Serial data access and SCL: Serial clock line.

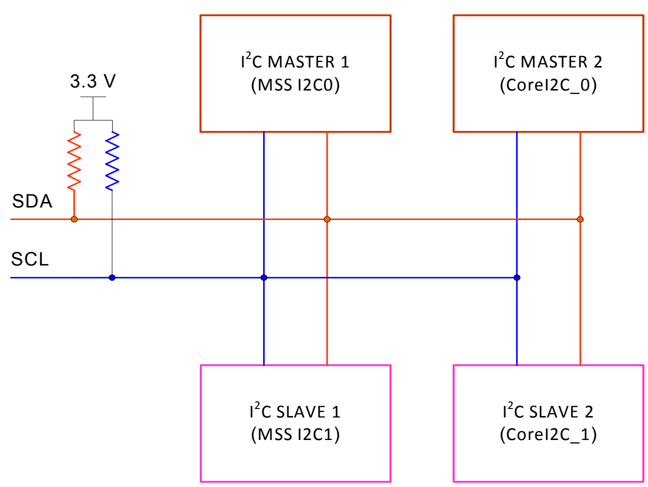

***Figure 1* • I^2C Bus with Multiple Masters and Slaves**

References

The following documents are referenced in this document. The references complement and help in understanding the relevant Microsemi SmartFusion2 FPGA device flows and features.

- *UG0331: SmartFusion2 Microcontroller Subsystem User Guide*
- *SmartFusion2 System Builder User Guide*
- *SmartFusion2 MSS I2C Configuration Guide*
- *SmartFusion2 MSS MMUART Configuration Guide*
- *UG0445: IGLOO2 FPGA and SmartFusion2 SoC FPGA Fabric User Guide*
- *UG0594: M2S090TS-EVAL-KIT SmartFusion2 Security Evaluation Kit User Guide*

Design Requirements

Table 1 lists the design requirements.

Table 1 • **Design Requirements**

Design Requirements	**Description**
Hardware Requirements	
SmartFusion2 Security Evaluation Kit: • FlashPro4 programmer (provided along with the kit)	Rev D or later
Desktop or Laptop	Any 64-bit Windows Operating System
Flying leads	To connect all I^2C SDA and SCL lines together (See Figure 14)
Software Requirements	
Libero® System-on-Chip (SoC)	v11.7
Microsoft .NET Framework 4 Client Profile	–

Features

The following features are implemented in the reference design.

- Write, Read, and Write-Read I^2C transaction types
- Two I^2C Masters (MSS I^2C and CoreI2C)
- Two I^2C Slaves (MSS I^2C and CoreI2C)
- Error detection
- Timeout

I^2C Transaction Types

The MSS I^2C and CoreI2C drivers are designed to handle the following three types of I^2C transactions:

- Write Transaction
- Read Transaction
- Write-Read Transaction

Write Transaction

The Master I^2C device initiates a Write transaction by sending a START bit when the bus is free. It continuously monitors the SDA line to determine the bus status. The START bit is followed by the 7-bit serial address of the target Slave device followed by the read/write bit indicating the direction of the transaction. The Slave acknowledges the receipt of its Slave address with an acknowledge bit. The Master sends one byte of data at a time to the Slave must acknowledge the receipt of each byte for the next byte to be sent. The Master sends a STOP bit to complete the transaction. Figure 2 on page 4 shows the I^2C write transaction.

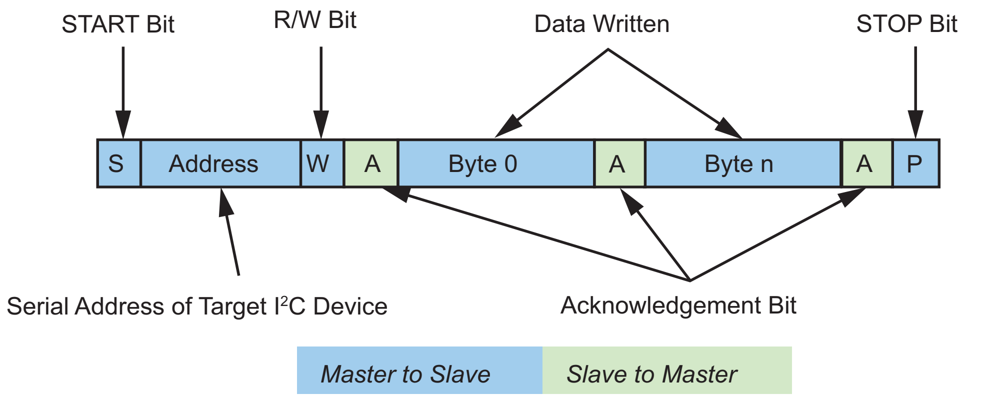

Figure 2 • **I^{2}C Write Transaction**

The Slave can abort the transaction by sending a non-acknowledge bit instead of an acknowledge bit. If the application programmer chooses not to send a STOP bit at the end of the transaction, the next transaction to begin with a repeated START bit.

Read Transaction

The Master I^{2}C device initiates a Read transaction by sending a START bit when the bus is free. The START bit is followed by the 7-bit serial address of the target Slave device followed by the read/write bit indicating the direction of the transaction. The Slave acknowledges the receipt of its Slave address with an acknowledge bit. The Slave sends one byte of data at a time to the Master. The Master must acknowledge the receipt of each byte for the next byte to be sent. The Master sends a non-acknowledge bit following the last byte it wishes to read. The Master sends a STOP bit to complete the transaction.

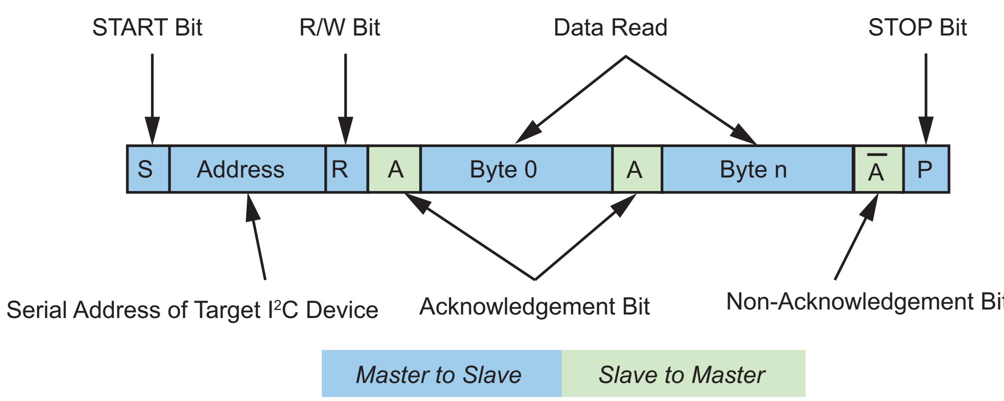

Figure 3 • **I^{2}C Read Transaction**

If the application programmer chooses not to send a STOP bit at the end of the transaction, the next transaction to begin with a repeated START bit.

Write-Read Transaction

The Write-Read transaction is a combination of a write transaction immediately followed by a read transaction. There is no STOP bit between the write and read phases of a Write-Read transaction. A repeated START bit is sent between the write and read phases.

The Write-Read transaction is typically used to send a command or offset in the write transaction specifying the logical data to be transferred during the read phase. Figure 4 shows the I^2C Write-Read transaction.

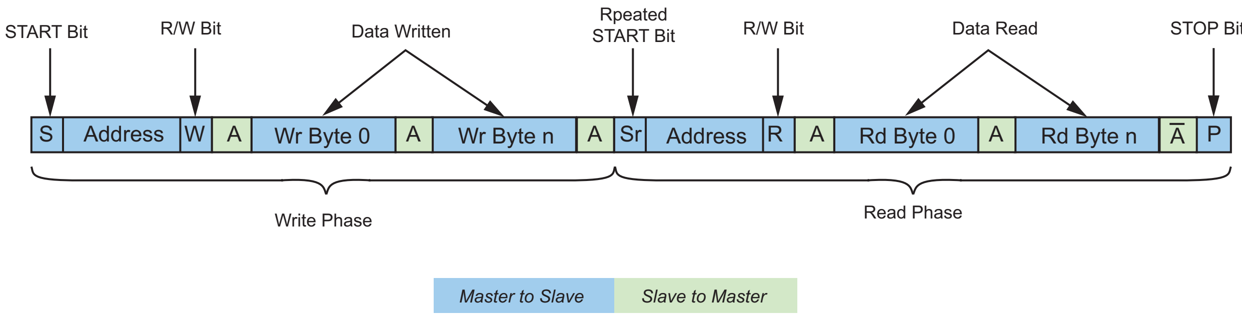

Figure 4 **• I^2C Write-Read Transaction**

If the application programmer chooses not to send a STOP bit at the end of the transaction, the next transaction to begin with a repeated START bit.

Implementation on SmartFusion2 Device

The I^2C transaction types (Write, Read, and Write-Read) have been implemented and validated using the SmartFusion2 Security Evaluation Kit board. This section describes the following:

- Design Description
- Hardware Implementation
- Software Implementation
- Running the Design

Design Description

The design consists of MSS, CoreAPB3 IP, and CoreI2C IP. Figure 5 shows the block diagram of the design.

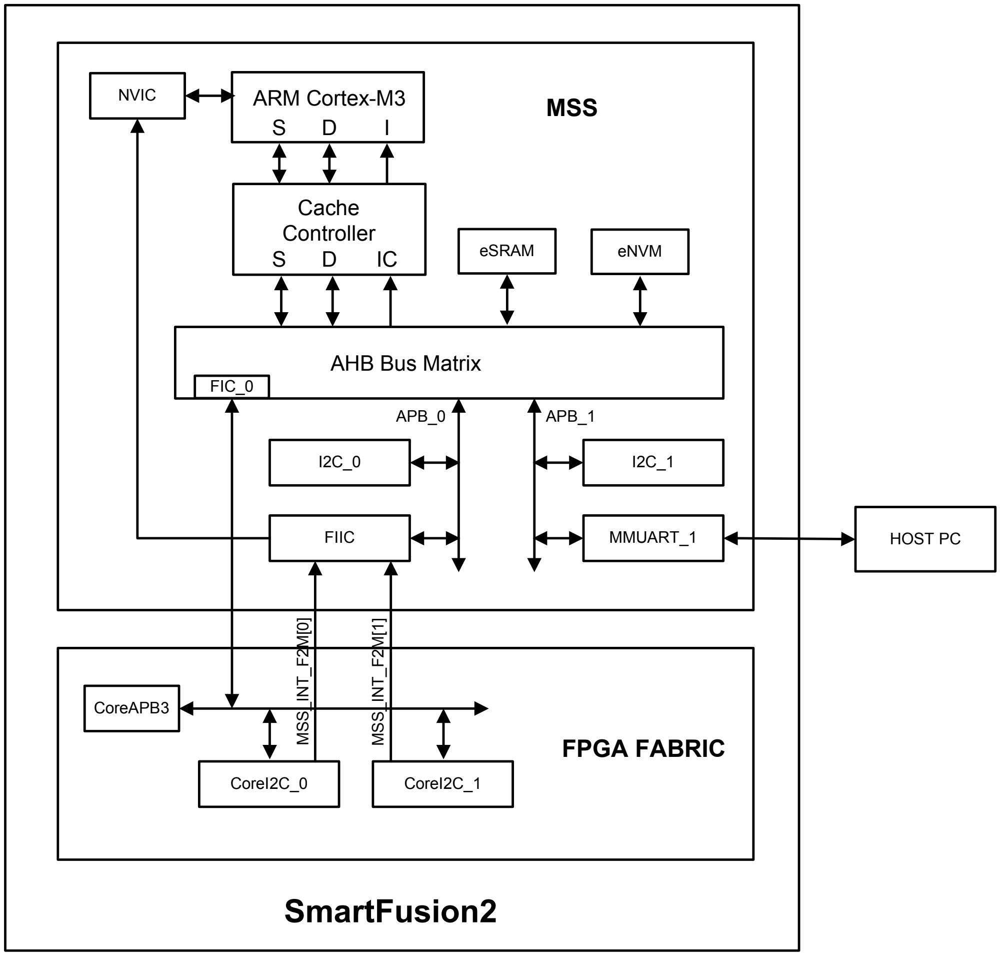

Figure 5 **• Top-Level Block Diagram of Design**

MSS is configured to use I2C_0, I2C_1, MMUART_1, fabric interface interrupt controller (FIIC), and a fabric interface controller (FIC_0). FIIC is configured to use fabric to MSS interrupt and FIC_0 is configured to use APB3 Master interface. CoreI2C_0 and CoreI2C_1 are connected to FIC_0 through a CoreAPB3 bridge and interrupt lines are connected to FIIC. For more information about MSS (ARM Cortex-M3, Cache controller, NVIC, AHB bus matrix, FIC, FIIC, I^2C, and MMUART),
see the *UG0331: SmartFusion2 Microcontroller Subsystem User Guide*.

The application code runs on the Cortex-M3 processor interfaces with the host PC through MMUART_1, and initiates the I^2C transactions.

Hardware Implementation

The System Builder is used to implement the hardware. Figure 6 shows the top-level SmartDesign of the reference design.

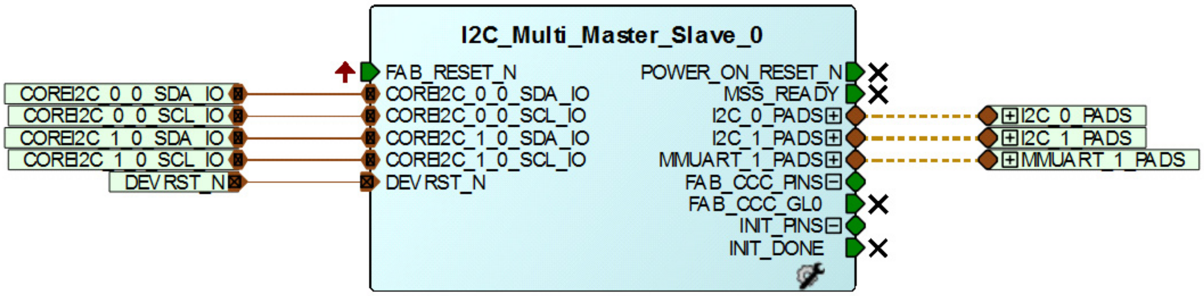

Figure 6 **• Top-Level SmartDesign**

Figure 7 shows the connections of MSS and IPs when the System Builder generated components are opened as SmartDesign.

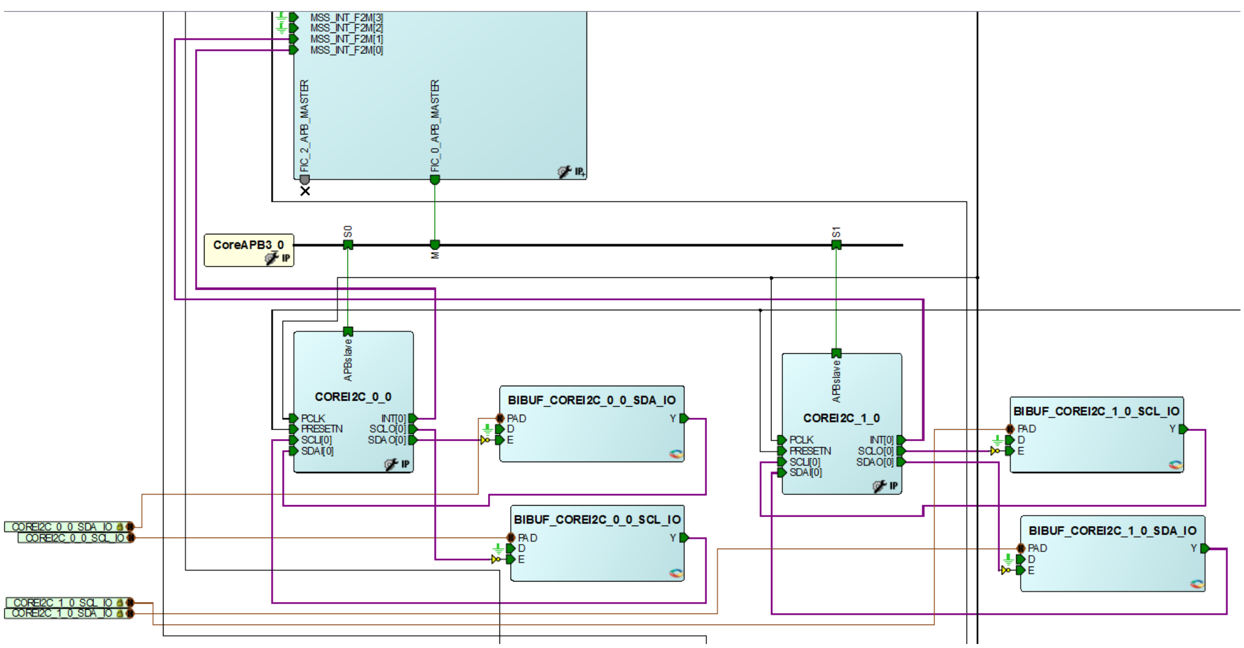

Figure 7 **• System Builder Opened as SmartDesign**

Configuring System Builder

This section describes how to configure device features and build a complete system using the **System Builder** graphical design wizard in the Libero SoC software. For more information about how to launch the **System Builder** wizard, see the *SmartFusion2 System Builder User Guide*.

The following steps describe how to configure the system builder for the reference design:

1. The **System Builder** window is displayed with **Device Features** page by default. Click **Next**, the **System Builder - Peripherals** page is displayed. Drag two instances of CoreI2C and drop on to the **MSS FIC_0 - MSS Master Subsystem**. Figure 8 shows the **Peripherals** page.

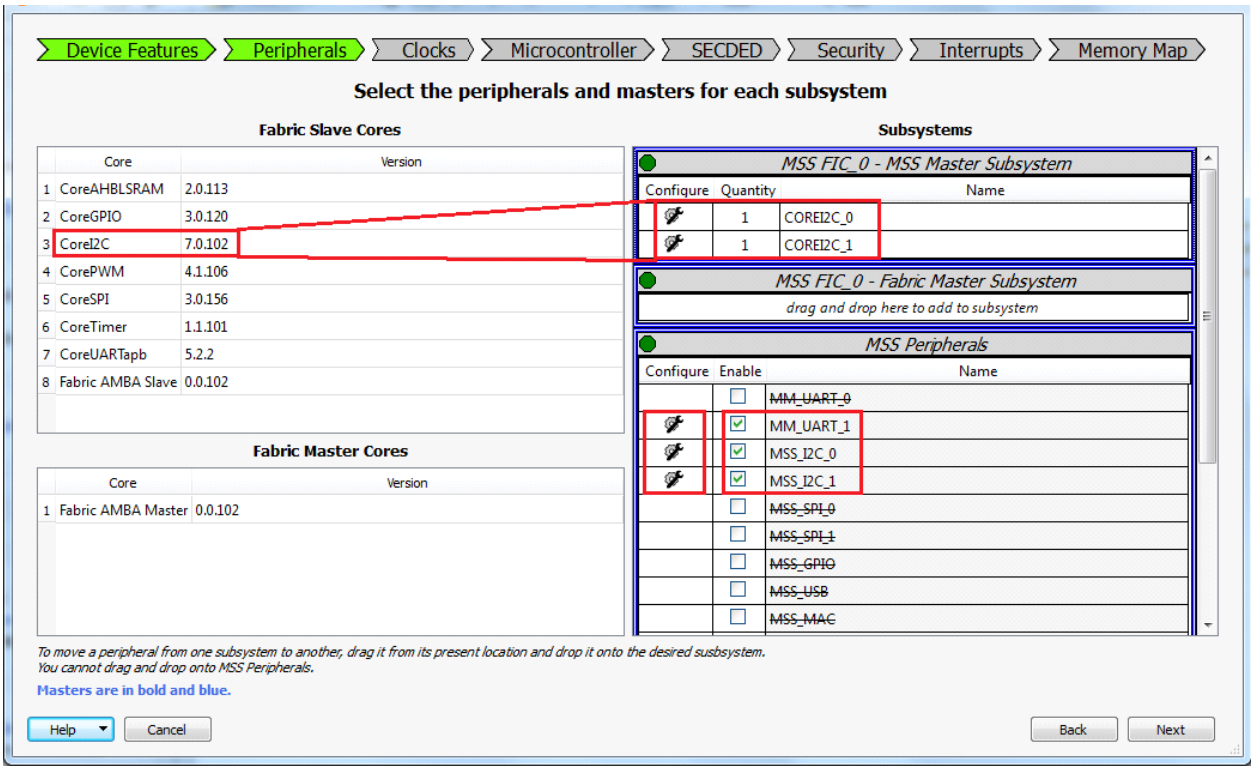

***Figure 8* • System Builder - Peripherals Page**

2. Configure two instances of CoreI2C by clicking **Configure** as shown in Figure 9.

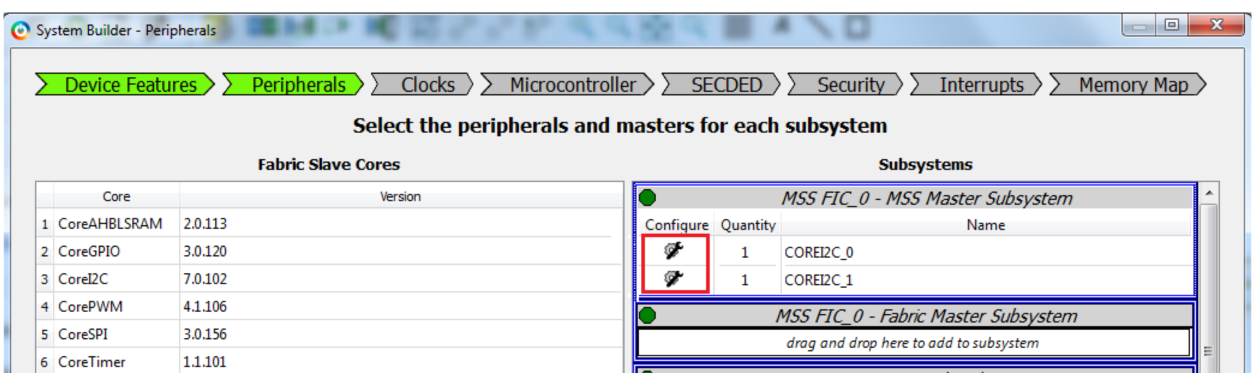

***Figure 9* • CoreI2C Configure Icon**

Use settings as shown in the Figure 10.

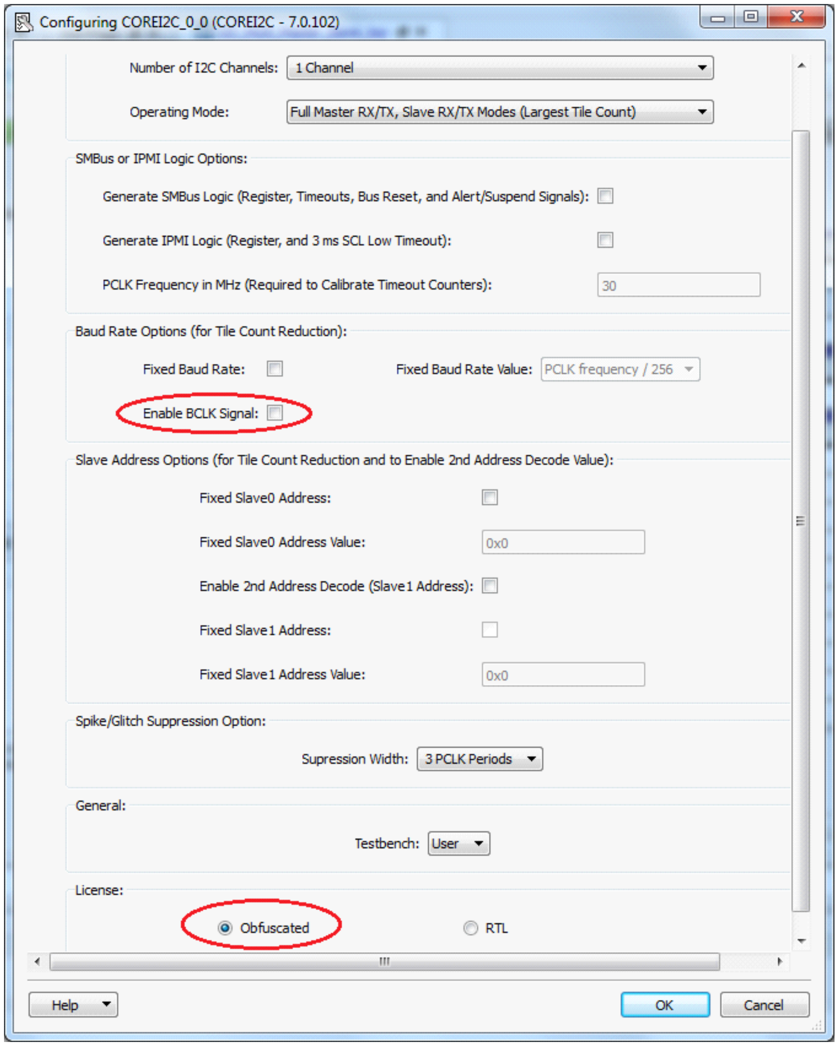

Figure 10 • **CoreI2C Configurator**

3. The design uses MMUART and I^{2}C MSS peripherals. Select **MM_UART_1**, **MSS_I2C_0**, **MSS_I2C_1** and uncheck all other peripherals (see Figure 8).
4. Click **Next**. Figure 11 on page 10 shows the **System Builder- Clock Settings** tab. Configure the System and Subsystem clocks in the **Clocks** page as listed in Table 2.

Table 2 • **System and Subsystem Clocks**

Clock Name	Frequency in MHz
System Clock	On-chip 25 MHz/50 MHz RC oscillator
M3_CLK	96
APB_0_CLK	24
APB_1_CLK	24
FIC_0_CLK	24

Figure 11 shows the **Clocks Configuration** dialog.

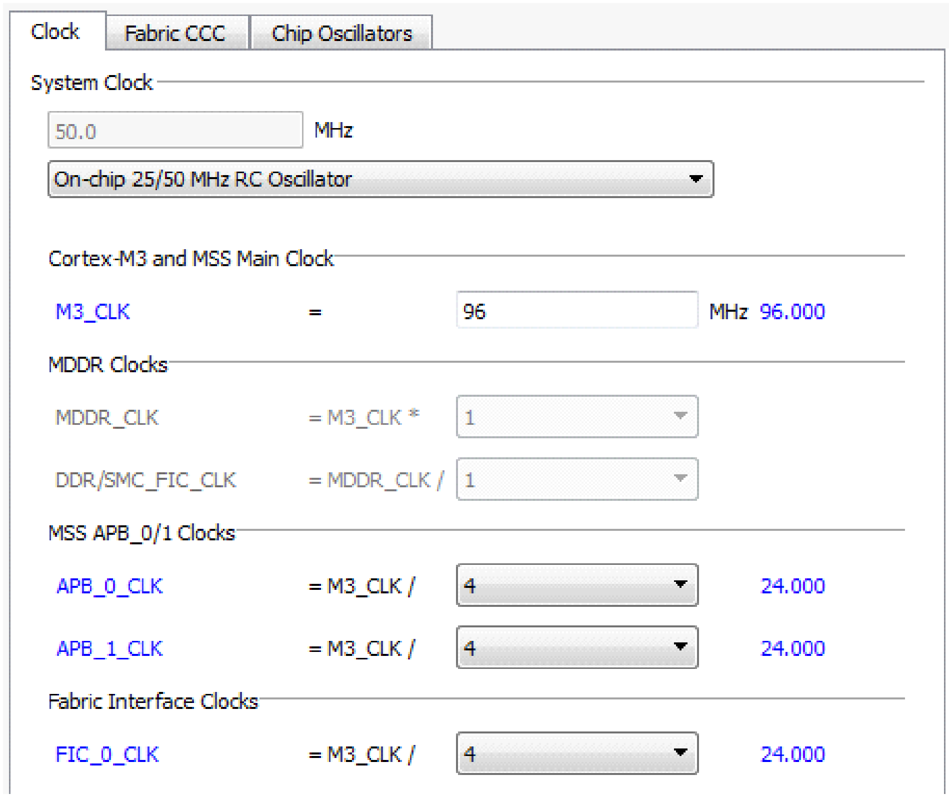

***Figure 11* • System and Subsystem Clocks Configuration**

5. Click **Next** to go to the **System Builder – Microcontroller** page. Retain the default values.
6. Click **Next** to go to the **System Builder – SECDED** page. Retain the default values.
7. Click **Next** to go to the **System Builder – Security** page. Retain the default values.
8. Click **Next** to go to the **System Builder – Interrupts** page. Check **Lock** check-boxes, as shown in Figure 12.

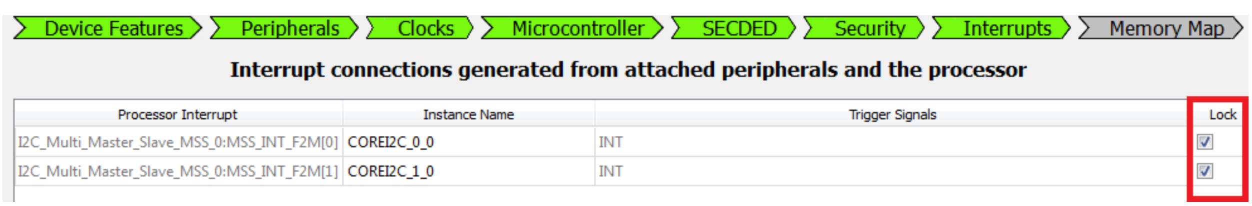

***Figure 12* • CoreI2C Interrupts**

9. Click **Next** and **Finish** to generate the design. Figure 13 shows the **Memory Map** page with CoreI2C memory map.

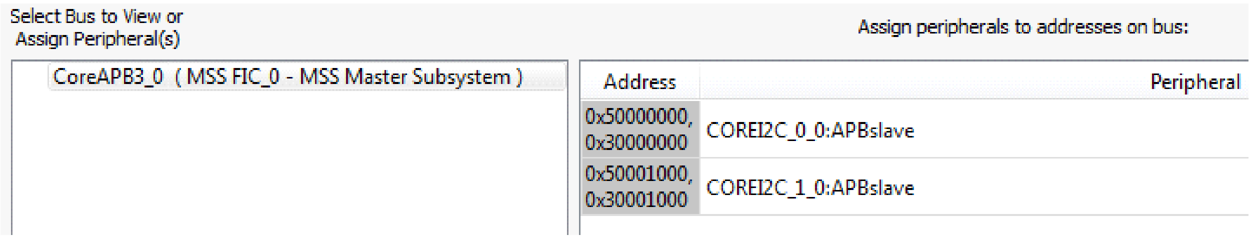

***Figure 13* • CoreI2C Memory Map**

Software Implementation

The software design performs the I^{2}C transaction types (Write, Read, and Write-Read) on receiving commands from user through GUI. All I^{2}C buffer (Master/Slave transmit/receive buffer) sizes are 1024 bytes. An I^{2}C Master (MSS I^{2}C Master/CoreI2C Master) writes up to 1024 bytes of data to an I^{2}C Slave (MSS I^{2}C Slave/CoreI2C Slave). The data received by the Slave is written to the Slave transmit buffer and overwrites some or all of the default contents. The default contents of MSS I^{2}C Slave is **<<---MSS Slave Tx data ------->>** and CoreI2C Slave is **<<---COREI2C Slave Tx data --->>.** During the read operation, the I^{2}C Master reads the content from the Slave transmit buffer and displays it on the GUI. The I^{2}C Master writes up to 1024 bytes of data to the Slave, and reads it back in the same operation, while performing the Write-Read transaction. It uses a repeated START bit between the write and read phases. Software design also performs the error detection and time out features.

The software design performs the following operations:

- Initialization of UART
- Initialization of MSS I^{2}C Master and CoreI2C Master with its I^{2}C serial address
  – MSS I^{2}C Master serial address - 0x20
  – CoreI2C Master serial address - 0x30
- Initialization of MSS I^{2}C Slave and CoreI2C Slave with its I^{2}C serial address
  – MSS I^{2}C Slave serial address - 0x21
  – CoreI2C Slave serial address - 0x31
- Performing the following I^{2}C transactions based on the command from the GUI:
  – MSS I^{2}C Master Perform Master Transmit - MSS I^{2}C Slave Receive
  – MSS I^{2}C Master Perform Master Receive - MSS I^{2}C Slave Transmit
  – MSS I^{2}C Master Perform Write-Read (MSS I^{2}C Slave) operation
  – MSS I^{2}C Master Perform Master Transmit - CoreI2C Slave Receive
  – MSS I^{2}C Master Perform Master Receive - CoreI2C Slave Transmit
  – MSS I^{2}C Master Perform Write-Read (CoreI2C Slave) operation
  – CoreI2C Master Perform Master Transmit - MSS I^{2}C Slave Receive
  – CoreI2C Master Perform Master Receive - MSS I^{2}C Slave Transmit
  – CoreI2C Master Perform Write-Read (MSS I^{2}C Slave) operation
  – CoreI2C Master Perform Master Transmit - CoreI2C Slave Receive
  – CoreI2C Master Perform Master Receive - CoreI2C Slave Transmit
  – CoreI2C Master Perform Write-Read (CoreI2C Slave) operation

Firmware Drivers

The following firmware drivers are used in this application:

- MSS MMUART driver: To communicate with GUI on the host PC
- MSS I^2C driver
- CoreI2C driver

For more information about the description of driver APIs and usage, see the respective driver user guide. See the "Appendix B: Updating Firmware Catalog For Latest Drivers" section on page 21 to update the drivers for latest version.

Application Program Interface (APIs)

Table 3 lists the APIs that are implemented in the software design.

Table 3 • **APIs for I^2C Transaction Types**

API	Description
UART_Polled_Rx	Receives data. It receives the contents of the UART receiver FIFO. It returns when the full content of the UART's receive FIFO has been transferred to the receive data buffer.
mss_read_transaction	MSS I^2C Master perform Read transaction
mss_write_transaction	MSS I^2C Master perform Write transaction
mss_write_read_transaction	MSS I^2C Master perform Write-Read transaction
mss_slave_write_handler	Stores the received data in Slave transmit buffer
corei2c_read_transaction	CoreI2C Master perform read transaction
corei2c_write_transaction	CoreI2C Master perform write transaction
corei2c_write_read_transaction	CoreI2C Master perform write-read transaction
corei2c_slave_write_handler	Stores the received data in Slave transmit buffer
SysTick_Handler	Service the I^2C timeout functionality
FabricIrq0_IRQHandler	CoreI2C 0 Fabric Interrupt handler
FabricIrq1_IRQHandler	CoreI2C 1 Fabric Interrupt handler

If the design is re-generating, the eNVM memory content file path to be updated. See the "Appendix C: Updating eNVM Memory Content File Path" section on page 23 to update the eNVM memory client in SmartDesign flow.

Running the Design

The reference design runs on the SmartFusion2 Security Evaluation Kit board. For more information about the SmartFusion2 Security Evaluation Kit board, see the *SmartFusion2 Security Evaluation Kit*.

Setting-up the Hardware

The following steps describe how to setup the hardware:

1. Connect the jumpers on the SmartFusion2 Security Evaluation Kit board as listed in Table 4.

Table 4 • **SmartFusion2 Security Evaluation Kit Jumper Settings**

Jumper	Pin (From)	Pin (To)	Comments
J3	1	2	Default
J8	1	2	Default

CAUTION: Ensure that power supply switch **SW7** is switched off while connecting the jumpers on the SmartFusion2 Security Evaluation kit.

2. Connect the Power supply to the J6 connector.
3. Switch on the power supply switch **SW7**.
4. Connect the FlashPro4 programmer to the J5 connector (JTAG Programming Header) of the SmartFusion2 Security Evaluation Kit board.
5. Connect the host PC USB port to the SmartFusion2 Security Evaluation Kit board's J18 (FTDI) USB connector using the USB mini-B cable. Ensure that the USB to UART bridge drivers are automatically detected. This can be verified in the **Device Manager** of the host PC.
6. If the USB to UART bridge drivers are not installed, download and install the drivers from *www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip*.
7. Program the SmartFusion2 Security Evaluation Kit board with the generated or provided *.stp file (see "Appendix A: Design Files" section on page 20) using FlashPro4.
8. Switch **OFF** the power supply switch **SW7.**
9. Connect the I^2C header pins and general purpose input-output (GPIO) header pins together using flying leads as listed in Table 5.

Table 5 • **I^2C SDA and SCL Connections**

I^2C Signal Name	I^2C Header - H1	GPIO Header - J1
SCL	6, 10	55, 57
SDA	7, 11	60, 62

Figure 14 shows the I^2C SDA and SCL connection using flying leads connectors. The wires are joined together to connect all the SDA lines and SCL lines.

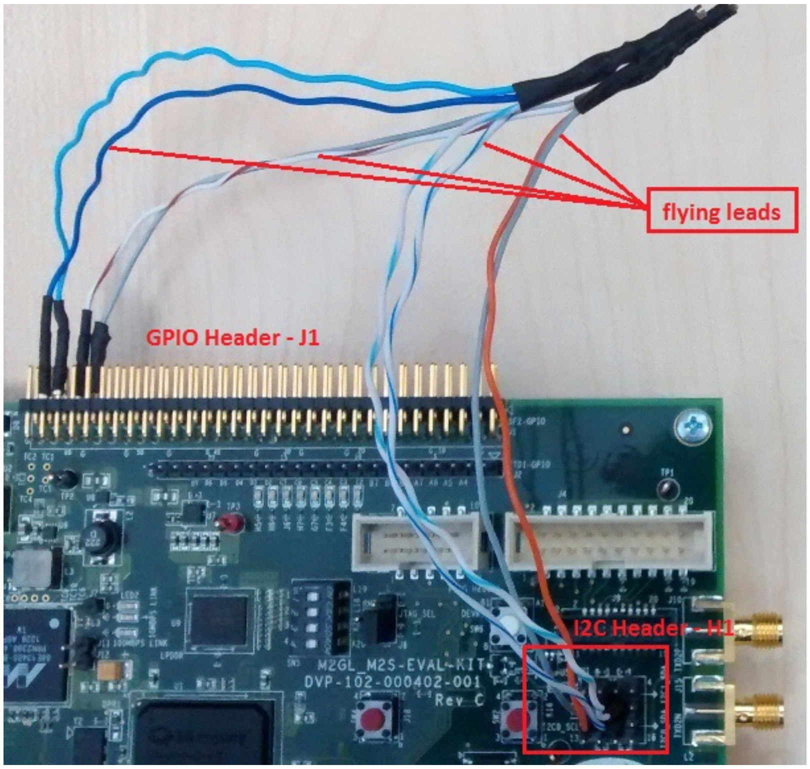

***Figure 14* • I^2C SDA and SCL Connections**

10. Switch **ON** the power supply switch, **SW7**.

Windows Application

The reference design provides a windows GUI, `M2S_I2C.exe` that runs on the host PC to communicate with the SmartFusion2 Security Evaluation Kit board. The UART protocol is used as communication protocol between the host PC and SmartFusion2 Security Evaluation Kit board. Figure 15 shows the initial screen of the GUI.

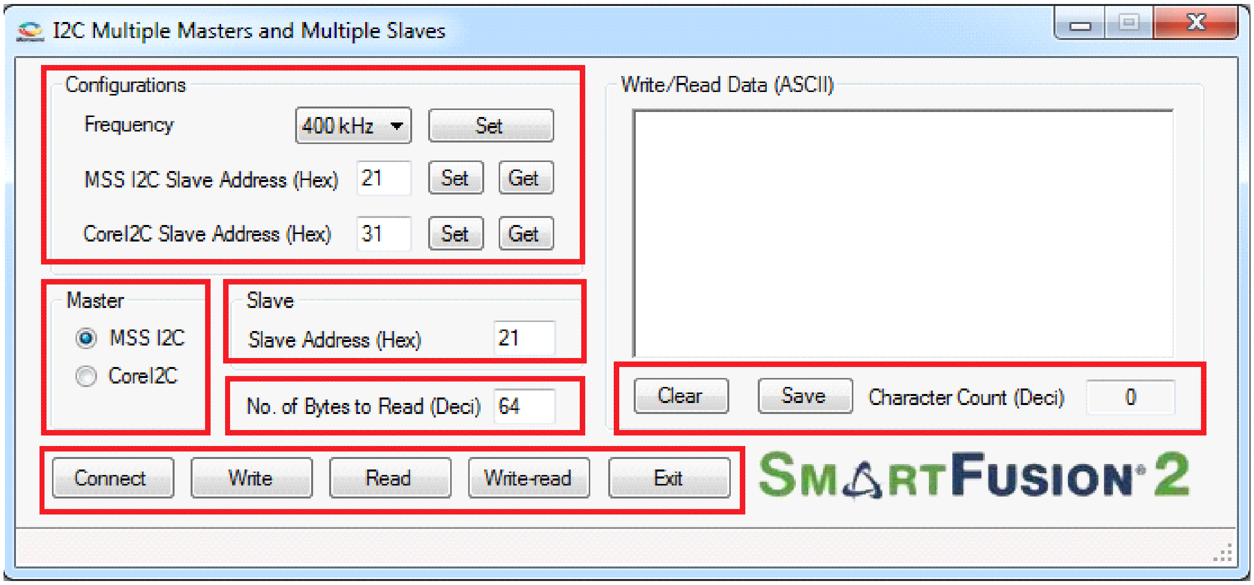

***Figure 15* • M2S_I2C GUI**

The M2S_I2C GUI consists of the following:

- **Configurations:** Consists of Frequency (serial clock), MSS I^2C Slave address, and CoreI2C Slave address.
  - Frequency: Select a serial clock from the drop-down menu and click **Set**.
  - MSS I^2C Slave Address (Hex): Enter (2-digit Hexadecimal) Slave address as per the I^2C specification and click **Set**. Click **Get** to view the already assigned Slave address.
  - CoreI2C Slave Address (Hex): Enter (2-digit Hexadecimal) Slave address as per the I^2C specification and click **Set**. Click **Get** to view the already assigned Slave address.
- **Master:** Select the following I^2C Masters:
  - MSS I^2C
  - CoreI2C
- **Slave:** Enter the Slave address of the I^2C Slave peripheral.
- **No.of Bytes to Read (Deci):** Enter the number of bytes to be read.
- **Buttons:**
  - **Connect:** Connects or disconnects the serial port communication between the host PC and the SmartFusion2 Security Evaluation Kit board
  - **Write:** Starts the Write transaction
  - **Read:** Starts the Read transaction
  - **Write-Read:** Starts the Write-Read transaction
  - **Exit:** Exits the application

- **Write/Read Data (ASCII):**
  - **Write Data:** Enter up to 1024 characters as write data during the write or Write-Read transaction.
  - **Read Data:** Displays received data during the read or write-read transaction.
  - **Clear:** Clears the text box.
  - **Save:** Saves the content as a text file.
  - **Character Count (Deci):** Displays the numbers of characters in the text box.

Running the GUI

The following steps describe how to run the GUI:

1. Launch the GUI. The default location is:
   <download_folder>\m2s_ac430_liberov11p7_df\M2S_I2C_DF \Windows_Utility\M2S_I2C.exe
2. Click **Connect** and wait for few seconds to connect the proper FDTI COM port. The connection status along with the COM Port and Baud rate is highlighted in Figure 16. Figure 16 shows the connection status.

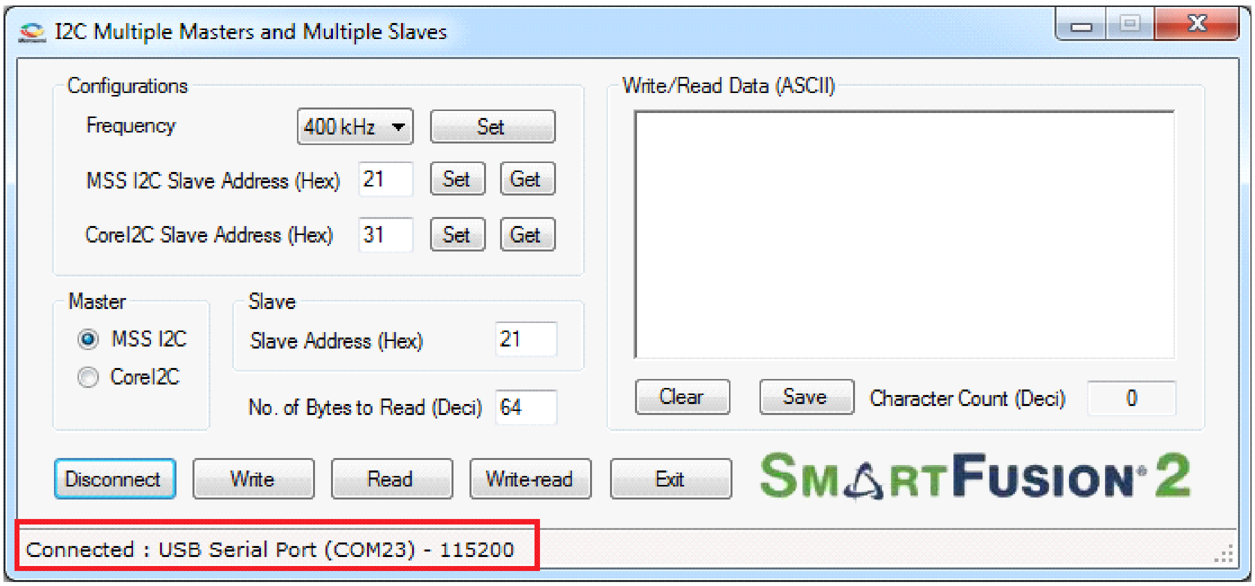

Figure 16 • **M2S_I2C Connection Status**

If the board is not connected, or programmed with incorrect `.stp` file, the GUI shows an error message as shown in Figure 17.

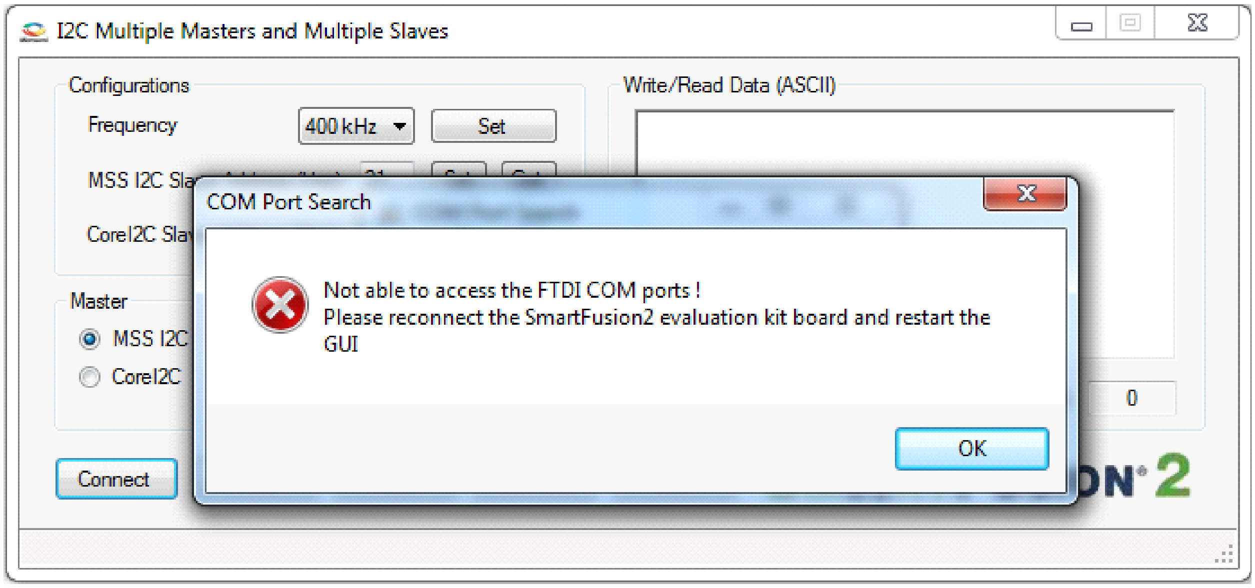

***Figure 17* • Connection Status - Error Message**

The following steps describe each I^2C transaction types (Write, Read, and Write-Read). All possible use cases are listed in Table 6 on page 19.

- **Write:**
  – Select a Master from the Master section
  – Enter a Slave address in the Slave section
  – Enter the write data
  – Click **Write**
- **Read:**
  – Select a Master from the Master section
  – Enter a Slave address in the Slave section
  – Enter the number of bytes to be read
  – Click **Read**
- **Write-Read:**
  – Select a Master from the Master section
  – Enter a Slave address in the Slave section
  – Enter the write data
  – Enter the number of bytes to be read
  – Click **Write-read**

Figure 18 shows the Read transaction type. The MSS I^2C Master reads from CoreI2C Slave. Write/Read Data section shows the default CoreI2C Slave read data.

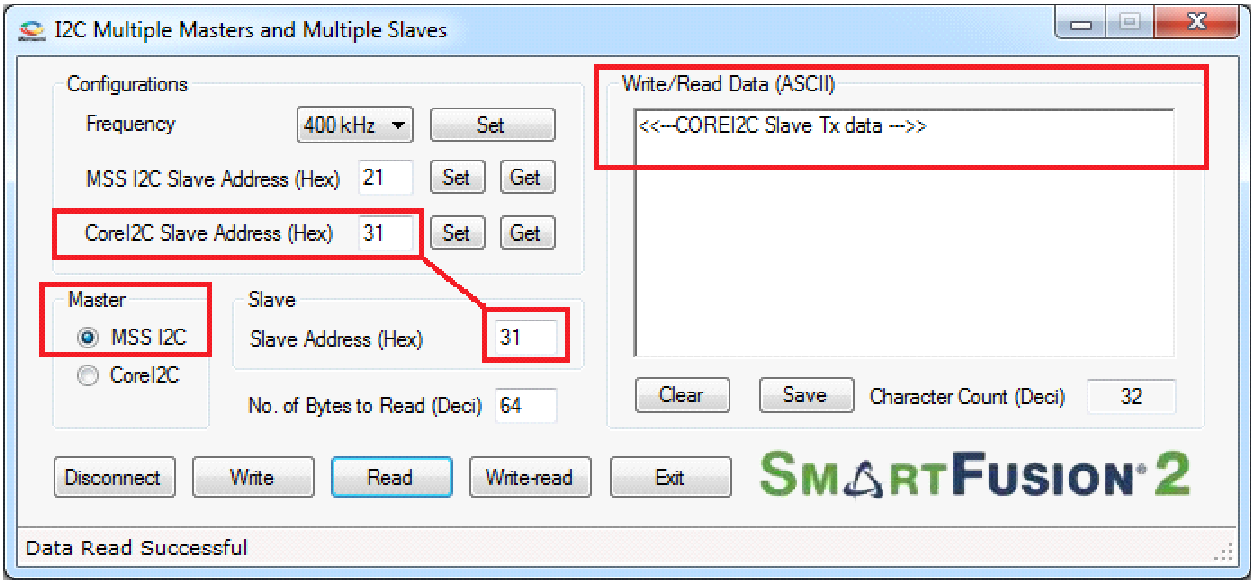

***Figure 18* • Read Transaction Type**

3. Read or write error occurs due to the non-availability of the selected Slave or due to connection problem. To validate error detection, one of the I^2C Slaves SDA line must be removed from the SmartFusion2 Security Evaluation Kit board. Remove either I^2C Header - H1 (7) or GPIO Header - J1 (62) pin and perform an I^2C transaction. Figure 19 shows the read error message when the MSS I^2C Master tries to read from the CoreI2C Slave.

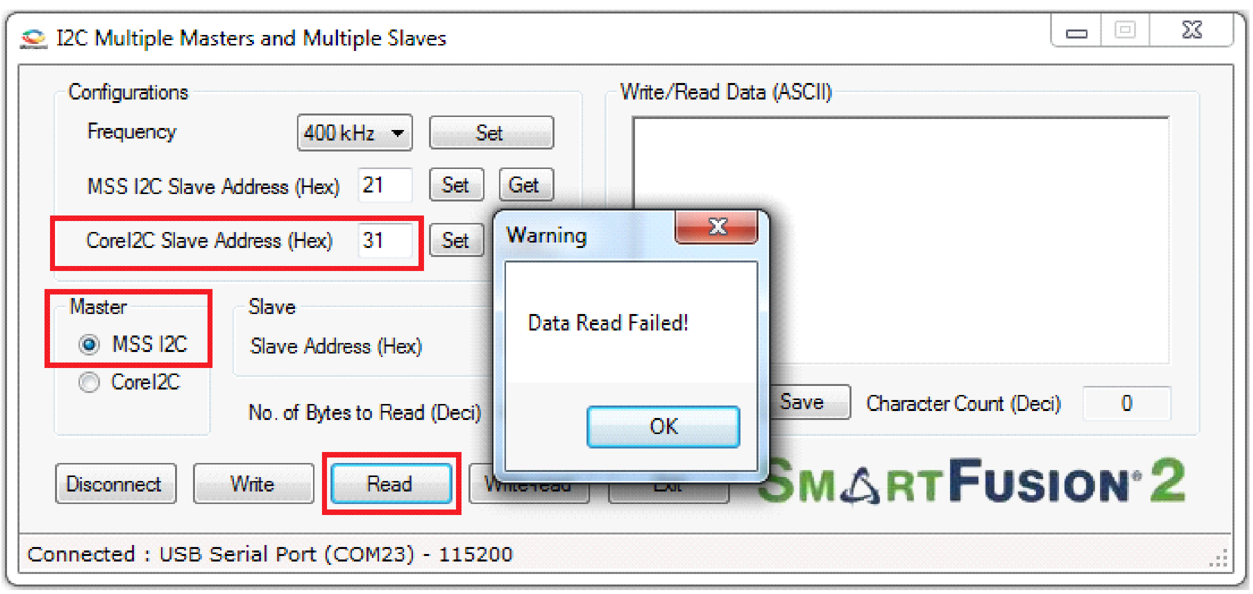

***Figure 19* • Read Error**

4. Connect the removed flying lead to GND and perform an I^2C transaction to test the time out. Figure 19 shows the time out message when the MSS I^2C Master tries to read from the CoreI2C Slave.

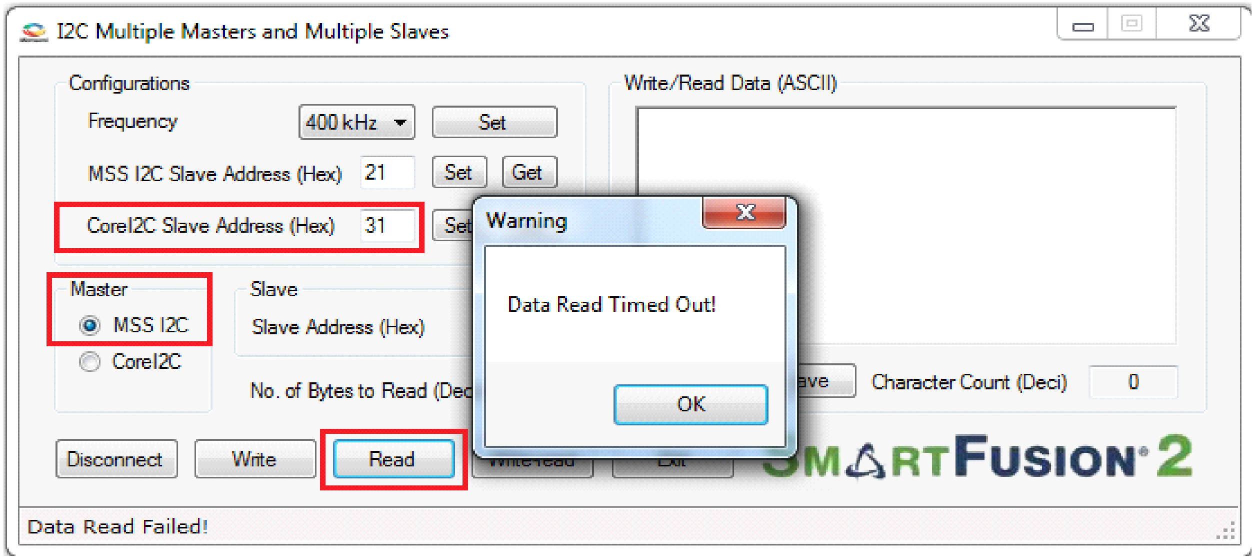

Figure 20 **• Read Time Out**

Use Cases

Table 6 lists the use cases.

Table 6 **• Use Cases**

I^2C Master	I^2C Slave	I^2C Transaction Type
MSS I^2C Master	MSS I^2C Slave	Write
		Read
		Write-Read
	CoreI2C Slave	Write
		Read
		Write-Read
CoreI2C Master	MSS I^2C Slave	Write
		Read
		Write-Read
	CoreI2C Slave	Write
		Read
		Write-Read

Conclusion

This application note describes the I^2C transaction types (Write, Read, and Write-Read) with a reference design, which implements multiple Masters and Slaves using the SmartFusion2 Security Evaluation Kit.

Appendix A: Design Files

The design files can be downloaded from the Microsemi SoC Products Group website:

http://soc.microsemi.com/download/rsc/?f=m2s_ac430_liberov11p7_df.

The design file consists of Libero SoC Verilog project, SoftConsole software project, and programming files (`*.stp`) for SmartFusion2 Security Evaluation Kit board. See the `Readme.txt` file included in the design file for the directory structure and description.

Appendix B: Updating Firmware Catalog For Latest Drivers

The following steps describe how to update firmware catalog for latest drivers.

1. Expand Handoff Design for Firmware Development in the **Design Flow** tab as shown in Figure 21. Right-click **Configure Firmware Cores** and click **Open Interactively**.

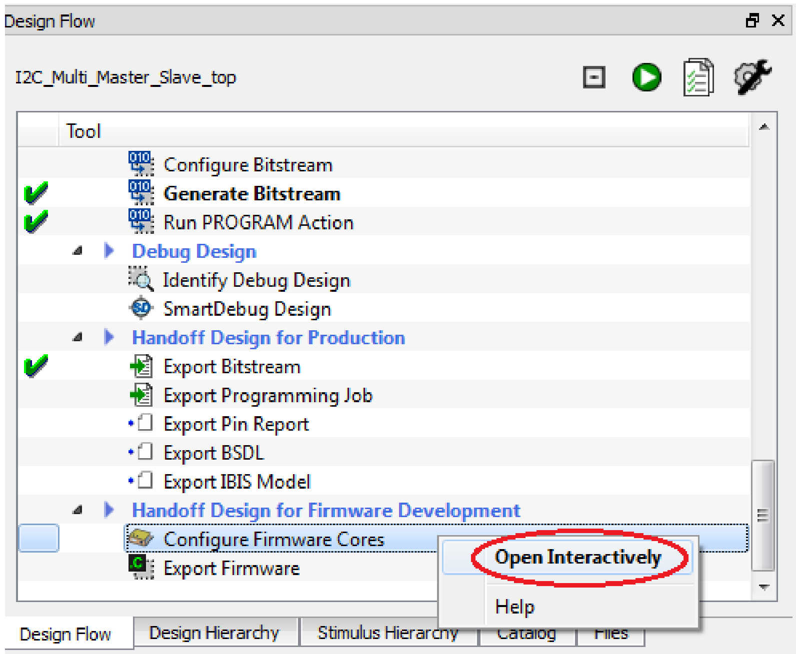

***Figure 21* • Invoking Configure Firmware Cores**

2. **DESIGN_FIRMWARE** tab displays MSS peripherals and CoreI2C drivers. Click **Download all firmware** as shown in Figure 22.

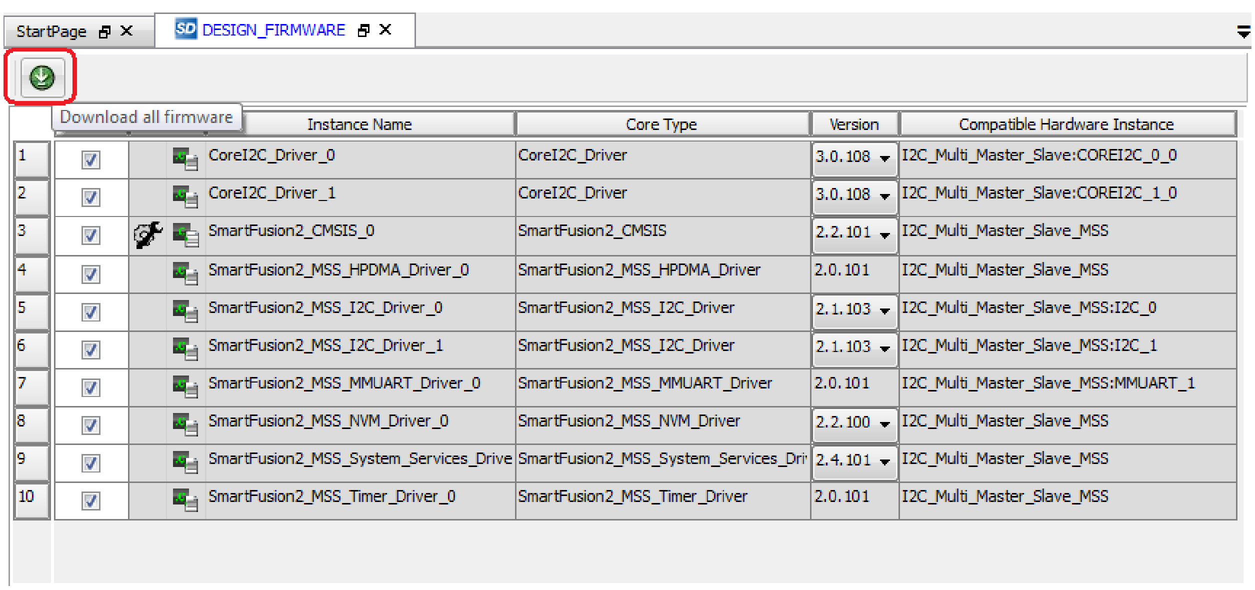

***Figure 22* • Download All Firmware**

Log - Messages window shows the firmware update status as shown in Figure 23.

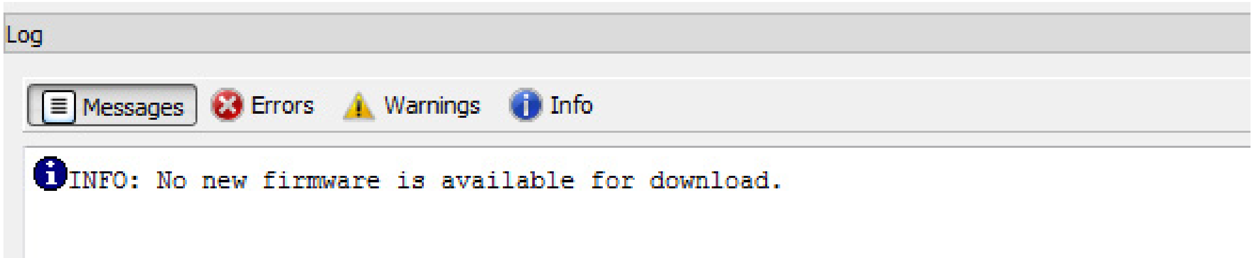

Figure 23 **• Download All Firmware**

Appendix C: Updating eNVM Memory Content File Path

Libero stores the eNVM Memory Content file path as absolute path where it is developed. When re-generating the design, the **Memory** window in the System Builder displays an error message as shown in Figure 24.

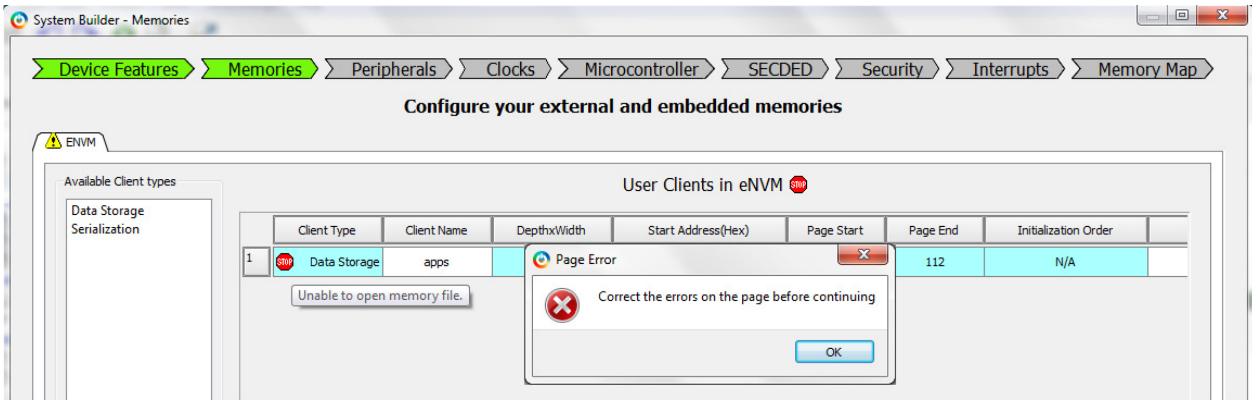

***Figure 24* • Memory File Path Error**

The following steps describe how to update eNVM memory content file path in SmartDesign flow:

1. Expand **I2C_Multi_Master_Slave_top** in the **Design Hierarchy** tab as shown in Figure 25. Right-click **I2C_Multi_Master_Slave** and then click **Open as SmartDesign**.

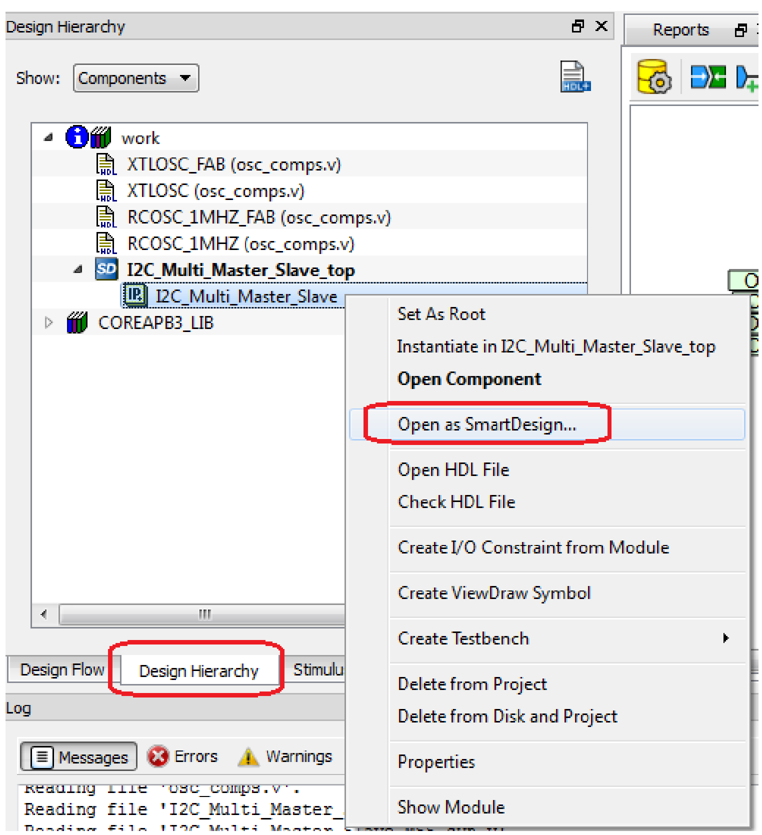

***Figure 25* • System Builder Opens as SmartDesign**

I2C_Multi_Master_Slave is opened as SmartDesign as shown in Figure 26.

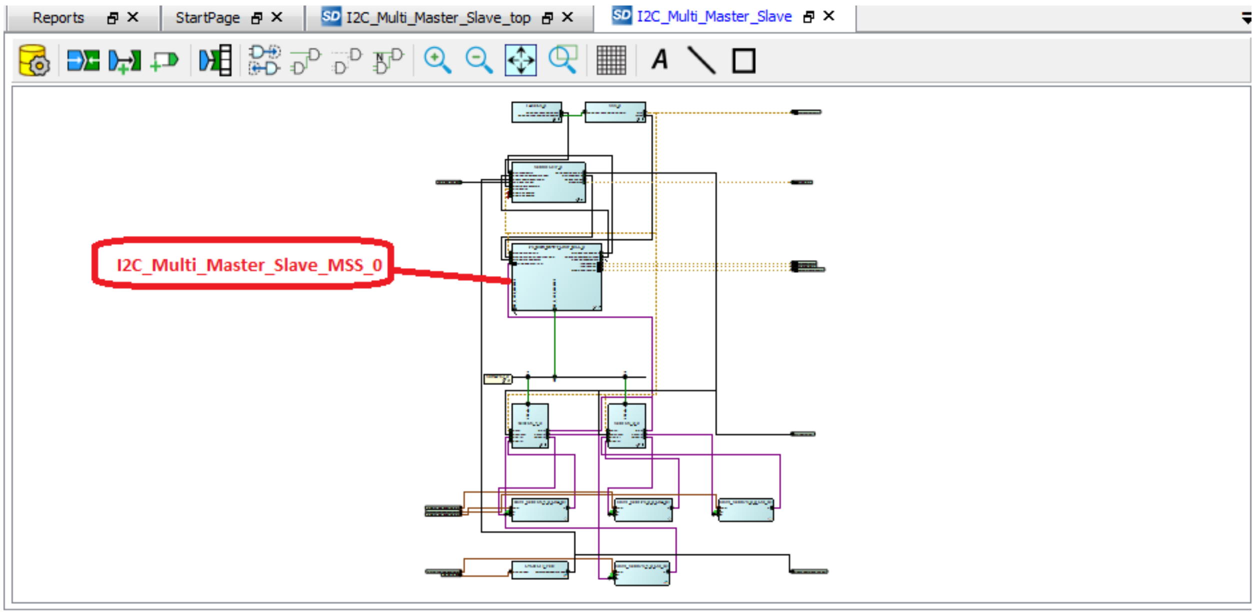

Figure 26 • **I2C_Multi_Master_Slave SmartDesign**

2. Double-click **I2C_Multi_Master_Slave_MSS_0** instance. **I2C_Multi_Master_Slave_MSS** is opened, as shown in Figure 27.

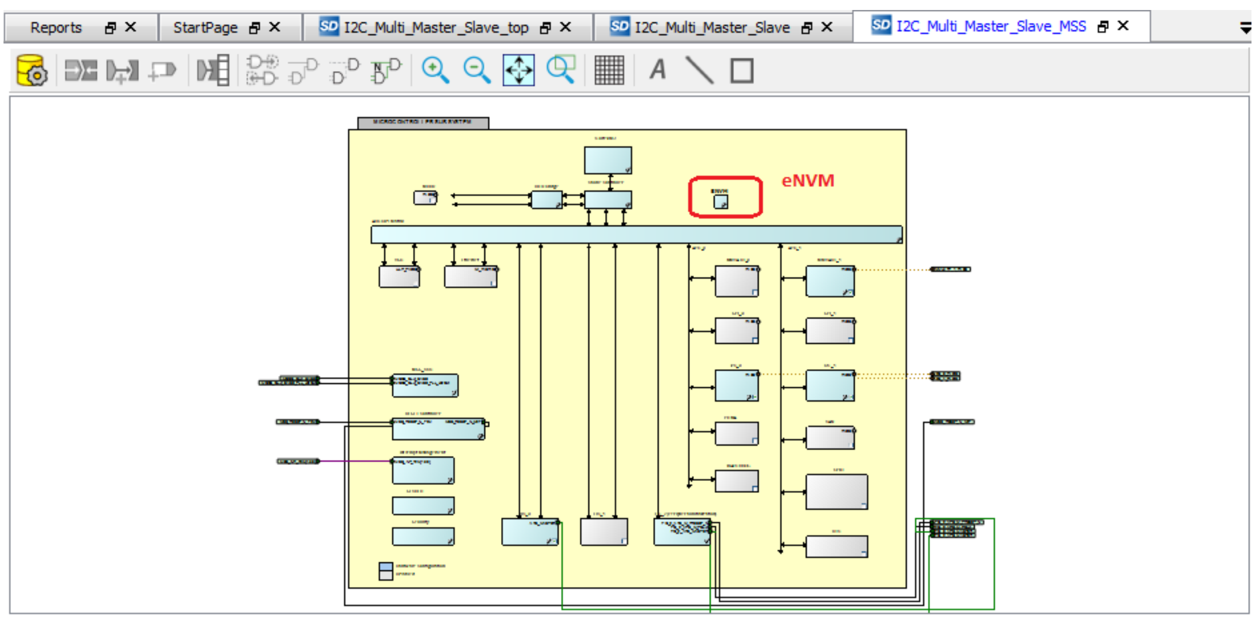

Figure 27 • **MSS Component**

3. Double-click **eNVM**. The **eNVM Configurator** window is opened as shown in Figure 28.

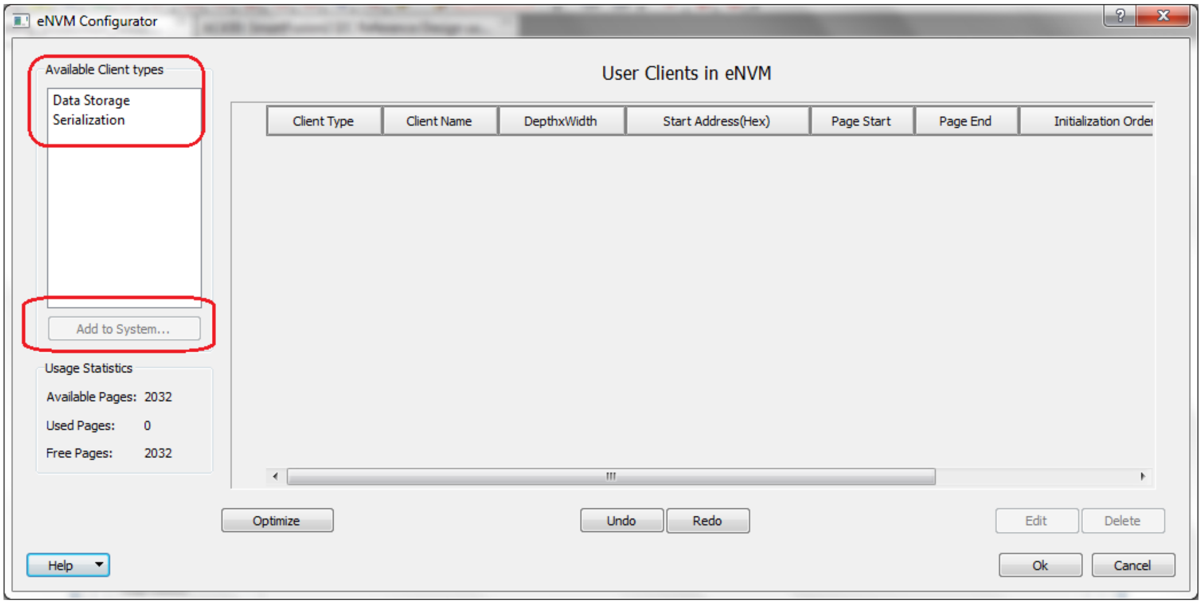

***Figure 28* • eNVM Configurator**

4. Select **Data Storage** under **Available Client Types** tab (see Figure 28) and then click **Add to System**. This opens **Add Data Storage Client** window, as shown in Figure 29.

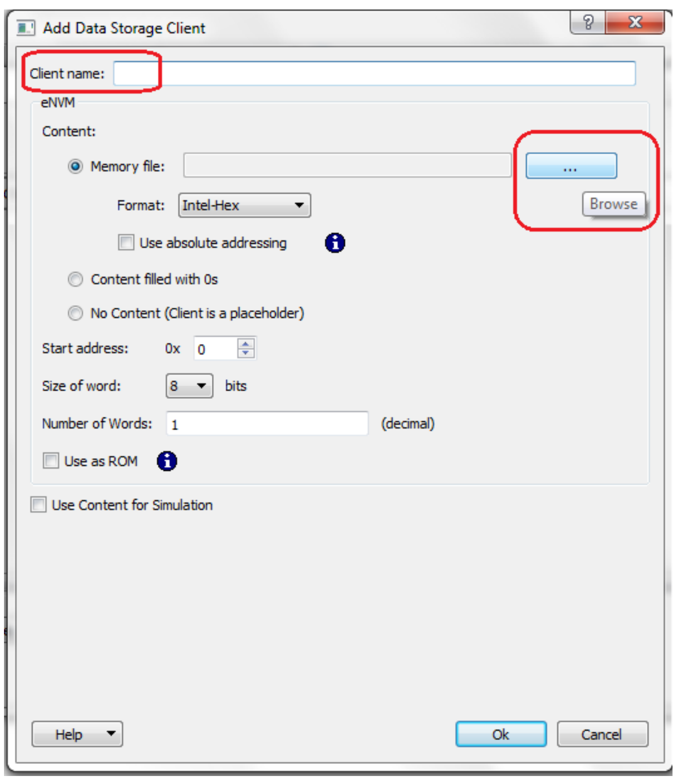

***Figure 29* • Add Data Storage Client Window**

5. Enter a client name and click **Memory file Browse**.
6. Enter the following in the **Open File** dialog box and then click **Open**:
   – Look in: *<download_folder>\m2s_ac430_liberov11p7_df\M2S_I2C_DF\Libero_Project\I2C_Multi_Master_Slave\SoftConsole\I2C_Multi_Master_Slave_MSS_CM3\I2C_Multi_Master_Slave_MSS_CM3_app\Release*
   – Files type: Intel-Hex Files (*.hex, *.ihx)
   – File name: I2C_Multi_Master_Slave_MSS_CM3_app.hex
7. Click **Ok** in the **Add Data Storage Client** window (see Figure 30).

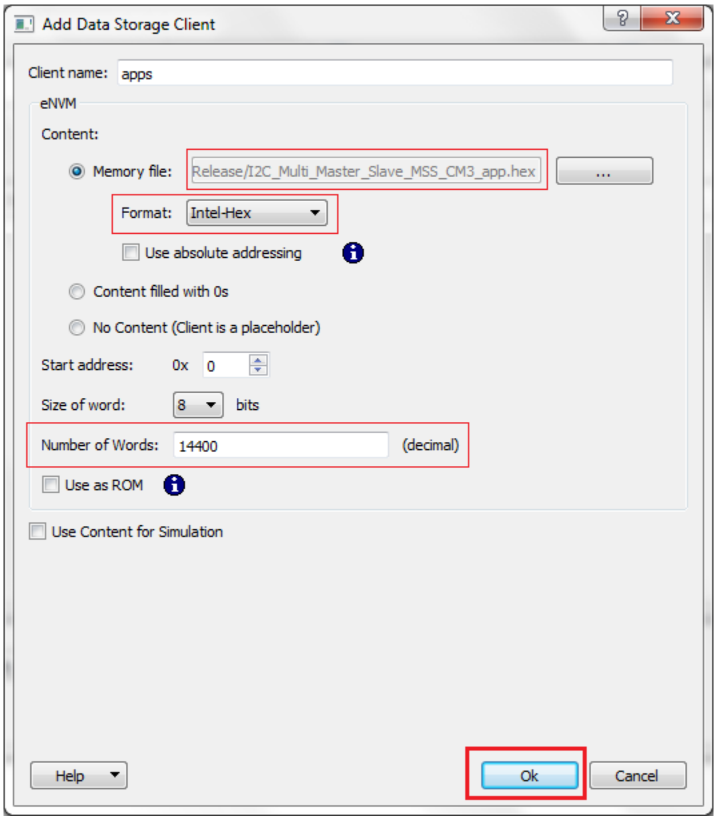

***Figure 30* • Add Data Storage Client Window**

8. Click **Ok** to close the **eNVM Configurator**.
9. Generate the following SmartDesigns by clicking **SmartDesign > Generate Component** or by clicking the **Generate Component** icon on the SmartDesign toolbar.
   – I2C_Multi_Master_Slave_MSS
   – I2C_Multi_Master_Slave
   – I2C_Multi_Master_Slave_top

10. Click the **Generate Bitstream** in the **Design Flow** tab (highlighted in Figure 31) or select **Design > Generate Bitstream** to synthesize the design, run layout using the I/O constraints and generate the programming file (bitstream file).

***Figure 31* • Generate Bitstream Icon**

The design implementation tools run in batch mode. Successful completion of a design step is indicated by a green check mark next to the Implement Design in the **Design Flow** tab.

If the design implementation tools run without updating System Builder component (without updating eNVM client), **Generate Bitstream** fails with an error message as shown in Figure 32.

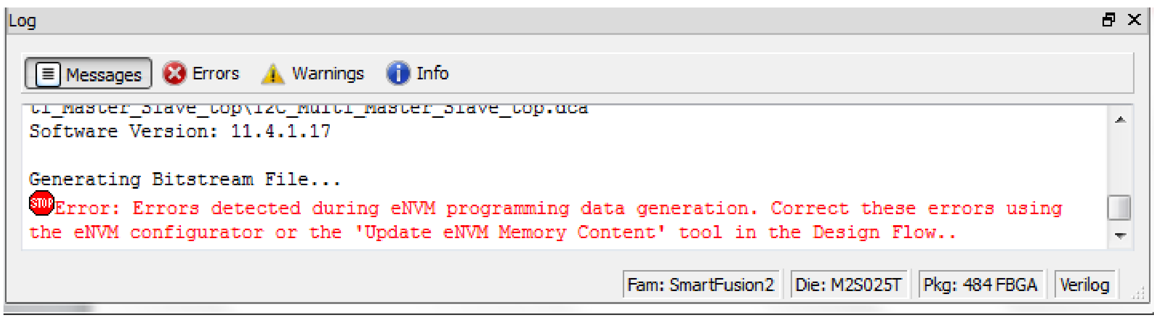

***Figure 32* • Log Window**

To generate Bitstream, eNVM memory content to be updated. See “Updating eNVM Memory Content” section in *AC426: Implementing Production Release Mode Programming for SmartFusion2 Application Note*.

List of Changes

The following table shows important changes made in this document for each revision.

Revision*	**Changes**	**Page**
Revision 4 (February 2016)	Updated the document for Libero SoC v11.7 software release (SAR 76663).	NA
Revision 3 (September 2015)	Updated the document for Libero SoC v11.6 software release (SAR 71268).	NA
Revision 2 (February 2015)	Updated the document for Libero SoC v11.5 software release (SAR 62801).	NA
Revision 1 (October 2014)	Initial release.	NA

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 Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; Enterprise Storage and Communication solutions, security technologies and scalable anti-tamper products; Ethernet Solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 4,800 employees globally. Learn more at **www.microsemi.com**.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

E-mail: sales.support@microsemi.com

51900298-4/02.16