

# **Table of Contents**

Purpose	· · · · 1 · · · · 1 · · · 2
Design Requirements	3
Features	3
I2C Transaction Types	
Implementation on SmartFusion2 Device	6
Design Description	
Appendix A: Design Files	19
Appendix B: Updating Firmware Catalog For Latest Drivers	19
Appendix C: Updating eNVM Memory Content File Path	21
List of Changes	25

## **Purpose**

SmartFusion<sup>®</sup>2 system-on-chip (SoC) field programmable gate array (FPGA) contains two inter-integrated circuit (I2C) peripherals available in the microcontroller subsystem (MSS). In addition, a number of I2C peripherals can be implemented in the FPGA fabric using CoreI2C IP. This application note describes the I2C transaction types (Write, Read, and Write-Read) with a reference design, which implements multiple Masters and Slaves using the SmartFusion2 Evaluation Kit.

# Introduction

12C is a two-wire serial bus interface that provides data transfer between several devices. The MSS has two identical I2C peripherals that perform serial-to-parallel data conversion originating from the serial devices, and parallel-to-serial conversion of data from the ARM<sup>®</sup> Cortex<sup>®</sup>-M3 processor. The Cortex-M3 embedded processor controls the I2C peripherals through the advanced peripheral bus (APB) interface.

The I2C peripherals in the SmartFusion2 SoC FPGA support I2C, system management bus (SMBus), and power management bus (PMBus) data transfers, which conform to the I2C v2.1 specification and support the SMBus v2.0 and PMBus v1.1 specifications. The I2C peripherals can operate as either a Master or a Slave, and can be configured independently. When operating in Master mode, the I2C peripherals generate the serial clock and data to the Slave device that needs to be accessed. The I2C peripheral generates the serial clock by dividing MSS clock which is controlled by a software. The I2C 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 I2C peripherals, refer to the *SmartFusion2 Microcontroller Subsystem User Guide*.



If the system requires more than two I2C peripherals, additional I2C peripherals have to be implemented in the FPGA fabric. Microsemi<sup>®</sup> provides CoreI2C IP to fulfill the design requirement. CoreI2C is available in the Libero<sup>®</sup> System-on-Chip (SoC) IP catalog.

This application note describes the I2C transaction types with a reference design which implements two Masters and two Slaves using the SmartFusion2 Evaluation Kit. MSS I2C0 and CoreI2C\_0 are configured as I2C MASTER1 and I2C MASTER2. The MSS I2C1 and CoreI2C\_1 are configured as I2C SLAVE1 and I2C 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 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 I2C transaction types. To communicate between the Masters and Slaves, MSS I2C0 SDA, MSS I2C1 SDA, CoreI2C\_0 SDA, and CoreI2C\_1 SDA are connected together, and MSS I2C0 SCL, MSS I2C1 SCL, CoreI2C\_0 SCL, and CoreI2C\_1 SCL are connected together on the SmartFusion2 Evaluation Kit board.

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



Figure 1 • I2C Bus with Multiple Masters and Slaves

# References

The following list of references is used in this document. These references complement and help in understanding the relevant Microsemi SmartFusion2 device features and flows that are described in this document:

- SmartFusion2 Microcontroller Subsystem User Guide
- SmartFusion2 System Builder User Guide
- SmartFusion2 MSS I2C Configuration
- SmartFusion2 MSS MMUART Configuration
- SmartFusion2 SoC FPGA Fabric User Guide
- SmartFusion2 SoC FPGA Evaluation Kit User Guide



# **Design Requirements**

Table 1 lists the design requirements.

#### Table 1 • Design Requirements

Design Requirements	Description	
Hardware Requirements		
SmartFusion2 Evaluation Kit	Rev C or later	
<ul> <li>FlashPro4 programmer (provided along with the kit)</li> </ul>		
Desktop or Laptop	Any 64-bit Windows Operating System	
Flying leads	To connect all I2C SDA and SCL lines together (Refer Figure 14)	
Software Requirements		
Libero SoC	v11.4 SP1	
Microsoft .NET Framework 4 Client Profile		

# **Features**

The following features are implemented in the reference design

- Write, Read, and Write-Read I2C transaction types
- Two I2C Masters (MSS I2C and CoreI2C)
- Two I2C Slaves (MSS I2C and CoreI2C)
- Error detection
- Time out

# **I2C Transaction Types**

The MSS I2C and CoreI2C drivers are designed to handle the following three types of I2C transactions:

- Write Transaction
- Read Transaction
- Write-Read Transaction

## Write Transaction

The Master I2C 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 I2C write transaction.





#### Figure 2 • I2C 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 I2C 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.





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 I2C Write-Read transaction.





# Implementation on SmartFusion2 Device

The I2C transaction types (Write, Read, and Write-Read) have been implemented and validated using the SmartFusion2 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, I2C, and MMUART) refer to the *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 I2C transactions.



## Hardware Implementation

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



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, refer to the *SmartFusion2 System Builder User Guide*.

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

 The System Builder window is displayed with Device Features tab by default. Click Next, the System Builder - Peripherals tab is displayed. Drag two instances of Corel2C and drop on to the MSS FIC\_0 - MSS Master Subsystem. Figure 8 shows the Peripherals tab.

	Subsystems
Core Version	MSS FIC_0 - MSS Master Subsystem
CoreAHBLSRAM 2.0.113	Configure Quantity Name
CoreGPIO 3.0.120	1 COREI2C_D
Corel2C 7.0.102	1 COREI2C_1
CorePWM 4.1.106	MSS FIC_0 - Fabric Master Subsystem
CoreSPI 3.0.156	drag and drop here to add to subsystem
CoreTimer 1.1.101	MSS Perinherals
CoreUARTapb 5.2.2	Configure Enable
Fabric AMBA Slave 0.0.102	MM UART 0
	M UART 1
Fabric Master Cores	Ø MSS_I2C_0
Core Version	Ø MSS_12C_1
Fabric AMBA Master 0.0.102	MSS_SPI_0
	MSS_SPI_1
	MSS_GPIO
	MSS-USB

#### Figure 8 • System Builder - Peripherals Tab

2. Configure two instances of Corel2C by clicking Configure as shown in Figure 9.

Sy	stem Builder - Perip	herals		
Device Features Peripherals Clocks Microcontroller SECDED Security Memory Map Select the peripherals and masters for each subsystem				
		Fabric Slave Cores	Subsystems	
	Core	Version	MSS FIC_0 - MSS Master Subsystem	
1	CoreAHBLSRAM	2.0.113	Configure Quantity Name	
2	CoreGPIO	3.0.120		
3	CoreI2C	7.0.102	I COREI2C_1	
4	CorePWM	4.1.106	MSS FIC 0 - Fabric Master Subsystem	
5	CoreSPI	3.0.156	draa and drop here to add to subsystem	
	Constinues	1 1 101		

Figure 9 • Corel2C Configure Icon



Use the settings as shown as shown in the Figure 10.

Number of I2C Channels	: 1 Channel			•	<b>^</b>
Operating Mode:	Full Master RX/TX, Sl	ave RX/TX Modes (Large	est Tile Count)	•	
SMBus or IPMI Logic Options:					
Generate SMBus Logic (Regis	ster, Timeouts, Bus Rese	t, and Alert/Suspend Sig	nals): 📃		
Generate IPMI Logic (Registe	er, and 3 ms SCL Low Tim	eout):			
PCLK Frequency in MHz (Req	uired to Calibrate Timeo	ut Counters):	30		
Baud Rate Options (for Tile Count	t Reduction):				
Fixed Baud Rate:	Fix	ed Baud Rate Value:	CLK frequency / 256		
Enable BCLK Signal:	>				
Slave Address Options (for Tile Co	ount Reduction and to E	nable 2nd Address Deco	de Value):		
Fixed Slave0	) Address:				
Fixed Slave0	) Address Value:	0x0			
Enable 2nd /	Address Decode (Slave1	Address): 🔲			
Fixed Slave 1	1 Address:	F C			
Fixed Slave 1	1 Address Value:	0x0			
Spike/Glitch Suppression Option:					
	Supression Width:	3 PCLK Periods			
General:					
	Testbench	: User 🔻			
License:					
© Obfu	scated	© RTL			-
					_

## Figure 10 • Corel2C Configurator

4

3. The design uses MMUART and I2C MSS peripherals. Select MM\_UART\_1, MSS\_I2C\_0, MSS\_I2C\_1 and uncheck all other peripherals (refer Figure 8).

Click Next. Figure 11 on page 10 shows the System Builder- Clock Settings tab is displayed. Configure the System and Subsystem clocks in the Clocks tab as listed in Table 2.

## Table 2 • System and Subsystem Clocks

Clock Name	Frequency in MHz	
System Clock	On-chip 25/50 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.

50.0	MHz		
On-chip 25/50 MHz RC	Oscillator		-
Cortex-M3 and MSS Mai	n Clock		
M3_CLK	=	96	MHz 96.000
MDDR Clocks			
MDDR_CLK	= M3_CLK *	1	
DDR/SMC_FIC_CLK	= MDDR_CLK	/ 1	
ISS APB_0/1 Clocks			
APB_0_CLK	= M3_CLK /	4	24.000
APB_1_CLK	= M3_CLK /	4	24.000
Fabric Interface Clocks			
57.0 0 OV	- M2 CLK /		24.000

Figure 11 • System and Subsystem Clocks Configuration

- 5. Click Next to go to the System Builder Microcontroller tab. Leave all the Default Selections.
- 6. Click **Next** to go to the **System Builder SECDED** tab. Leave all the Default Selections.
- 7. Click Next to go to the System Builder Security tab. Leave all the Default Selections.
- 8. Click **Next** to go to the **System Builder Interrupts** tab. Check **Lock** check-boxes as shown in Figure 12.

Device Features Peripherals	Clocks Microcom	troller > SECDED > Security > Interrupts > Memory om attached peripherals and the processor	Map >
Processor Interrupt	Instance Name	Trigger Signals	Lock
I2C_Multi_Master_Slave_MSS_0:MSS_INT_F2M[0]	COREI2C_0_0	INT	<b>V</b>
I2C_Multi_Master_Slave_MSS_0:MSS_INT_F2M[1]	COREI2C_1_0	INT	

Figure 12 • Corel2C Interrupts



9. Click **Next** and **Finish** to generate the design. Figure 13 shows the **Memory Map** tab with Corel2C memory map.

Select Bus to View or Assign Peripheral(s)		Assign peripherals to addresses on bus:
CoreAPB3_0 (MSS FIC_0 - MSS Master Subsystem)	Address	Peripheral
	0x50000000, 0x30000000	COREI2C_0_0:APBslave
	0x50001000, 0x30001000	COREI2C_1_0:APBslave
1		
Figure 13 • Corel2C Memory Map		

#### igure 13 • Coreizo Merriory Map

### **Software Implementation**

The software design performs the I2C transaction types (Write, Read, and Write-Read) on receiving commands from user through GUI. All I2C buffer (Master/Slave transmit/receive buffer) sizes are 1024 bytes. An I2C Master (MSS I2C Master/CoreI2C Master) writes up to 1024 bytes of data to an I2C Slave (MSS I2C 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 I2C Slave is <---MSS Slave Tx data ---->> and CoreI2C Slave is <---COREI2C Slave Tx data --->>. During the read operation, the I2C Master reads the content from the Slave transmit buffer and displays it on the GUI. The I2C 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 I2C Master and CoreI2C Master with its I2C serial address
  - MSS I2C Master serial address 0x20
  - Corel2C Master serial address 0x30
- Initialization of MSS I2C Slave and CoreI2C Slave with its I2C serial address
  - MSS I2C Slave serial address 0x21
  - Corel2C Slave serial address 0x31
- Performing the following I2C transactions based on the command from the GUI:
  - MSS I2C Master Perform Master Transmit MSS I2C Slave Receive
  - MSS I2C Master Perform Master Receive MSS I2C Slave Transmit
  - MSS I2C Master Perform Write-Read (MSS I2C Slave) operation
  - MSS I2C Master Perform Master Transmit CoreI2C Slave Receive
  - MSS I2C Master Perform Master Receive CoreI2C Slave Transmit
  - MSS I2C Master Perform Write-Read (CoreI2C Slave) operation
  - Corel2C Master Perform Master Transmit MSS I2C Slave Receive
  - Corel2C Master Perform Master Receive MSS I2C Slave Transmit
  - Corel2C Master Perform Write-Read (MSS I2C Slave) operation
  - Corel2C Master Perform Master Transmit Corel2C Slave Receive
  - Corel2C Master Perform Master Receive Corel2C Slave Transmit
  - Corel2C Master Perform Write-Read (Corel2C 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 I2C driver
- Corel2C driver

For more information about the description of driver APIs and usage, refer to the respective driver user guide. Refer to the "Appendix B: Updating Firmware Catalog For Latest Drivers" section on page 19 to update the drivers for latest version.

#### APIs

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

#### Table 3 • APIs for I2C 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 I2C Master perform Read transaction
mss_write_transaction	MSS I2C Master perform Write transaction
mss_write_read_transaction	MSS I2C Master perform Write-Read transaction
mss_slave_write_handler	Stores the received data in Slave transmit buffer
corei2c_read_transaction	Corel2C Master perform read transaction
corei2c_write_transaction	Corel2C Master perform write transaction
corei2c_write_read_transaction	Corel2C Master perform write-read transaction
corei2c_slave_write_handler	Stores the received data in Slave transmit buffer
SysTick_Handler	Service the I2C timeout functionality
FabricIrq0_IRQHandler	Corel2C 0 Fabric Interrupt handler
FabricIrq1_IRQHandler	Corel2C 1 Fabric Interrupt handler

If the design is re-generating, the eNVM memory content file path to be updated. Refer to the "Appendix C: Updating eNVM Memory Content File Path" section on page 21 to update the eNVM memory client in SmartDesign flow.

# **Running the Design**

The reference design runs on the SmartFusion2 Evaluation Kit board. For more information about the SmartFusion2 Evaluation Kit board, refer to SmartFusion2 Evaluation Kit.

### Setting up the Hardware

The following steps describe how to setup the hardware:

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

#### Table 4 • SmartFusion2 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 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 Evaluation Kit board.
- 5. Connect the Host PC USB port to the SmartFusion2 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 Evaluation Kit board with the generated or provided \*.stp file (Refer to "Appendix A: Design Files" section on page 19) using FlashPro4.
- 8. Switch OFF the power supply switch SW7.
- 9. Connect the I2C header pins and general purpose input-output (GPIO) header pins together using flying leads as listed in Table 5.

Table 5 • I2C SDA and SCL Connections

I2C Signal Name	I2C Header - H1	GPIO Header - J1
SCL	6,10	55, 57
SDA	7,11	60, 62

Figure 14 shows the I2C SDA and SCL connection using flying leads connectors. The wires are joined together to connect all the SDA lines and SCL lines.



#### Figure 14 • I2C 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 Evaluation Kit board. The UART protocol is used as communication protocol between the Host PC and SmartFusion2 Evaluation Kit board. Figure 15 shows the initial screen of the GUI.

오 I2C Multiple Mas	ters and Multiple Slaves	
Configurations Frequency MSS I2C Slave CoreI2C Slave	400 kHz ▼ Set Address (Hex) 21 Set Get Address (Hex) 31 Set Get	Write/Read Data (ASCII)
Master MSS I2C CoreI2C	Slave Slave Address (Hex) 21	Clear Save Character Count (Deci) 0
Connect	Write Read Write-read	SMARTFUSION <sup>®</sup> 2

#### Figure 15 • M2S\_I2C GUI

The M2S\_I2C GUI consists of the following:

- Configurations: Consists of Frequency (serial clock), MSS I2C Slave address, and Corel2C Slave address.
  - Frequency: Select a serial clock from the drop-down menu and click Set.
  - MSS I2C Slave Address (Hex): Enter (2-digit Hexadecimal) Slave address as per the I2C specification and click Set. Click Get to view the already assigned Slave address.
  - Corel2C Slave Address (Hex): Enter (2-digit Hexadecimal) Slave address as per the I2C specification and click Set. Click Get to view the already assigned Slave address.
  - Master: Select the following I2C Masters:
    - MSS I2C
    - Corel2C
- Slave: Enter the Slave address of the I2C 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 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 textbox.
  - Save: Saves the content as a text file.
  - Character Count (Deci): Displays the numbers of characters in the textbox.

#### Running the GUI

The Following steps describe how to run the GUI:

- 1. Launch the GUI. The default location is: <download\_folder>\M2S\_AC430\_liberov11p4\_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.

2 I2C Multiple Masters and Multiple Slaves	
Configurations         Frequency       400 kHz          MSS I2C Slave Address (Hex)       21         Corel2C Slave Address (Hex)       31	Write/Read Data (ASCII)
Master     Slave       Image: Slave Address (Hex)     Slave Address (Hex)       Image: Slave Address (Hex)     Slave Address (Hex)       Image: Slave Address (Hex)     Slave Address (Hex)	Clear Save Character Count (Deci) 0
Disconnect Write Read Write-read	EM SMARTFUSION 2

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.

Configurations	Write/Read Data (ASCII)
Frequency	400 kHz  Set
MSS I2C Slar	
Corel2C Slav	OM Port Search
Master MSS I2C	Not able to access the FTDI COM ports ! Please reconnect the SmartFusion2 evaluation kit board and restart the GUI
Corel2C	

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

The Following steps describe each I2C transaction types (Write, Read and Write-Read). All possible use cases are listed in Table 6 on page 18.

- 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 I2C Master reads from CoreI2C Slave. Write/Read Data section shows the default CoreI2C Slave read data.

See I2C Multiple Masters and Multiple Slaves	
Configurations Frequency 400 kHz V Set	Write/Read Data (ASCII) <corei2c data="" slave="" tx="">&gt;</corei2c>
MSS I2C Slave Address (Hex) 21 Set Get CoreI2C Slave Address (Hex) 31 Set Get	
Master Slave Slave Slave Address (Hex) 31	
No. of Bytes to Read (Deci) 64	Clear Save Character Count (Deci) 32
Disconnect Write Read Write-read	Ext SMARTFUSION <sup>®</sup> 2

### 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 I2C Slaves SDA line must be removed from the SmartFusion2 Evaluation Kit board. Remove either I2C Header - H1 (7) or GPIO Header - J1 (62) pin and perform an I2C transaction. Figure 19 shows the read error message when the MSS I2C Master tries to read from the CoreI2C Slave.

I2C Multiple Masters and Multiple Slaves		3
Configurations Write/Read Data (ASCII)		
Frequency 400 kHz - Set		
MSS I2C Slave Address (Hex) 21 Set Get		
Corel2C Slave Address (Hex) 31 Set Warning		
Master Slave Data Read Failed!		
Corel2C		
No. of Bytes to Read (Deci) OK Save Character Count (De	ci) ()	
	<b>10N°2</b>	
Connected : USB Serial Port (COM23) - 115200		.:

Figure 19 • Read Error



 Connect the removed flying lead to GND and perform an I2C transaction to test the time out. Figure 19 shows the time out message when the MSS I2C Master tries to read from the CoreI2C Slave.





## **Use Cases**

Table 6 lists the use cases.

## Table 6 • Use Cases

SI.No	I2C Master	I2C Slave	I2C Transaction Type
1	MSS I2C Master	MSS I2C Slave	Write
2			Read
3			Write-Read
4		Corel2C Slave	Write
5			Read
6			Write-Read
7	Corel2C Master	MSS I2C Slave	Write
8			Read
9			Write-Read
10		Corel2C Slave	Write
11			Read
12			Write-Read



# **Appendix A: Design Files**

You can download the SmartFusion2 design files from the Microsemi SoC Products Group website: *http://soc.microsemi.com/download/rsc/?f=M2S\_AC430\_liberov11p4\_DF*. The design file consists of a Libero Verilog project. Refer to the Readme.txt file included in the design file for the directory structure and description.

# Appendix B: Updating 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**.

sign How	
2C_Multi_Master_Slave_top	
Tool	
🥮 Configure Bitstream	
V Generate Bitstream	
V 🚆 Run PROGRAM Action	
Debug Design	
💐 Identify Debug Design	
🕸 SmartDebug Design	
Handoff Design for Production	
🖌 🛛 🛃 Export Bitstream	
🛃 Export Programming Job	
• 🗋 Export Pin Report	
• 🗋 Export BSDL	
• 🗋 Export IBIS Model	
Handoff Design for Firmware Develop	ment E
Configure Firmware Cores	One I show the her
🖳 Export Firmware	Open Interactively
	Help
Design Flow Design Hierarchy Stimulus Hierarchy	

Figure 21 • Invoking Configure Firmware Cores





2. **DESIGN\_FIRMWARE** tab displays MSS peripherals and Corel2C drivers. Click **Download all firmware** as shown in Figure 22.

Start	Page 🗗 🗙	(	SD D	SIGN_FIRMWARE 🗗 🗙			
	Downloa	d all fi	irmwa	re Instance Name	Core Type	Version	Compatible Hardware Instance
1	<b>V</b>			oreI2C_Driver_0	CoreI2C_Driver	3.0.108 👻	I2C_Multi_Master_Slave:COREI2C_0_0
2	<b>V</b>		- C	oreI2C_Driver_1	CoreI2C_Driver	3.0.108 👻	I2C_Multi_Master_Slave:COREI2C_1_0
3	<b>V</b>	Ø,	<b>-</b> 5	martFusion2_CMSIS_0	SmartFusion2_CMSIS	2.2.101 👻	I2C_Multi_Master_Slave_MSS
4	<b>V</b>		<b>.</b> S	martFusion2_MSS_HPDMA_Driver_0	SmartFusion2_MSS_HPDMA_Driver	2.0.101	I2C_Multi_Master_Slave_MSS
5	<b>V</b>		<b>-</b> 5	martFusion2_MSS_I2C_Driver_0	SmartFusion2_MSS_I2C_Driver	2.1.103 👻	I2C_Multi_Master_Slave_MSS:I2C_0
6	<b>V</b>		<b>-</b> 5	martFusion2_MSS_I2C_Driver_1	SmartFusion2_MSS_I2C_Driver	2.1.103 👻	I2C_Multi_Master_Slave_MSS:I2C_1
7	<b>V</b>		- S	martFusion2_MSS_MMUART_Driver_0	SmartFusion2_MSS_MMUART_Driver	2.0.101	I2C_Multi_Master_Slave_MSS:MMUART_1
8	<b>V</b>		<b>-</b> 5	martFusion2_MSS_NVM_Driver_0	SmartFusion2_MSS_NVM_Driver	2,2.100 👻	I2C_Multi_Master_Slave_MSS
9	<b>V</b>		<b>e</b> 1	martFusion2_MSS_System_Services_Drive	SmartFusion2_MSS_System_Services_Driv	2.4.101 👻	I2C_Multi_Master_Slave_MSS
10	<b>V</b>		<b>-</b>	martFusion2_MSS_Timer_Driver_0	SmartFusion2_MSS_Timer_Driver	2.0.101	I2C_Multi_Master_Slave_MSS

#### Figure 22 • Download All Firmware

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

Log			
🔳 Messages 🛛 🔀 Errors	🚹 Warnings 👔 Info		
OINFO: No new firmwa	are is available for d	download.	
Figure 23 • Download All Firm	Iware		
$\sim$			

# Appendix C: Updating eNVM Memory Content File Path

Libero stores the eNVM Memory Content file path as absolute path where it was developed. So while re-generating the design, the **Memory** tab in the system builder throws an error as shown in Figure 24.

System Builder - Memories	Memories >> Periph	herals >> Clocks >> Mid	rrocontroller >>> SECC	DED >> Secu	urity >> Inte	errupts >> Memory Map
		Configure your externa	l and embedded mei	mories		
Available Client types User Clients in eNVM 😂						
Serialization	Client Type	Client Name DepthxWidth	Start Address(Hex)	Page Start	Page End	Initialization Order
	1 Data Storage	apps 💿 Page Erro	or	×	112	N/A
	Unable to open m	memory file.	orrect the errors on the page be	fore continuing	C	

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 click Open as SmartDesign.

Design Hierarchy	E × Reports E
Show: Components	
	Set As Root Instantiate in I2C_Multi_Master_Slave_top
C	Open as SmartDesign Open HDL File Check HDL File Create I/O Constraint from Module
< <u> </u>	Create ViewDraw Symbol Create Testbench
Design Flow Design Hierarchy Stimulu Log	Delete from Project Delete from Disk and Project
Messages       Services         Reading       The osc_comps.vi.	Properties
Reading file 'I2C_Multi_Master_	Show Module

Figure 25 • System Builder Opens as SmartDesign





I2C\_Multi\_Master\_Slave is opened as SmartDesign as shown in Figure 26.



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



Available Client types			Us	er Clients in eNVM			
Data Storage Serialization	Client Type	Client Name	DepthxWidth	Start Address(Hex)	Page Start	Page End	Initialization Order
Add to System Jsage Statistics							
Available Pages: 2032 Jsed Pages: 0							
Free Pages: 2032	•		m	•			•
	Optimize		Und	io Redo			Edit Delete
Help 🔻							Ok Cancel

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

### Figure 28 • eNVM Configurator

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

Add Data Storage Client
Not onto storing client     Client name:     eNVM   Content:             Memory file:      Format:   Intel+Hex     Intel+Hex     Browse     Image: Content filed with 0s               One Content filed with 0s                No Content (Client is a placeholder)         Start address:   0x             Size of word:                Dits
Number of Words: 1 (decimal)
Use as ROM 🚯
Use Content for Simulation
Help  Ok Cancel

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\_liberov11p4\_DF\M2S\_I2C\_DF\SoftConsole\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 (refer Figure 30).

Add Data Storage Client
Client name: apps
eNVM
Memory file: Release/I2C_Multi_Master_Slave_MSS_CM3_app.hex
Format: Intel-Hex
Use absolute addressing
Content filled with 0s
No Contant (Clantia a placeholder)
• No content (client is a placeholder)
Start address: 0x 0 荣
Size of word: 8 v bits
Number of Words: 14400 (decimal)
Use as ROM
Use Content for Simulation
Help  Cancel

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





 Click the Generate Bitstream icon in the Design Flow window (circled 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).

esign Flow	5 ×
I2C_Multi_Master	
Tool	Generate Bitstream

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** window.

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

og					8,
🔳 Messages 🔞 Errors 🗼 Warnings 🍈 In	fo				
Software Version: 11.4.1.17	_stave_cop.dd	a	1		*
Generating Bitstream File					
Serror: Errors detected during eNVM	programming	data generation	. Correct these	errors using	
the eNVM configurator or the 'Update	eNVM Memory	Content' tool i	n the Design Flo	ow	-
		Fam: SmartFusi	on2 Die: M2S025T	Pkg: 484 FBGA Veri	log

#### Figure 32 • Log Window

To generate Bitstream, eNVM memory content to be updated. Refer "Updating eNVM Memory Content" section in *Implementing Production Release Mode Programming for SmartFusion2 - Libero SoC v11.4 Application Note*.

# List of Changes

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

Revision*	Changes	Page
Revision 1 (October 2014)	Initial release.	NA
Note: *The revision of the last page	number is located in the part number after the hyphen. The part number is displayed at th e of the document. The digits following the slash indicate the month and year of publicati	ne bottom ion.



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 Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense and 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; security technologies and scalable anti-tamper products; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif. and has approximately 3,400 employees globally. Learn more at **www.microsemi.com**.

© 2014 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.