SmartFusion2 SoC FPGA Demo: Code Shadowing from SPI Flash to DDR Memory

User's Guide

Microsemi

Table of Contents

SmartFusion2 SoC FPGA - Code Shadowing from SPI Flash to DDR Memory

Introduction

This demo design shows SmartFusion®2 system-on-chip (SoC) field programmable gate array (FPGA) device capabilities for code shadowing from serial peripheral interface (SPI) flash memory to double data rate (DDR) synchronous dynamic random access memory (SDRAM) and executing the code from DDR SDRAM. Figure 1 shows the top level block diagram for code shadowing from SPI flash to DDR demo.

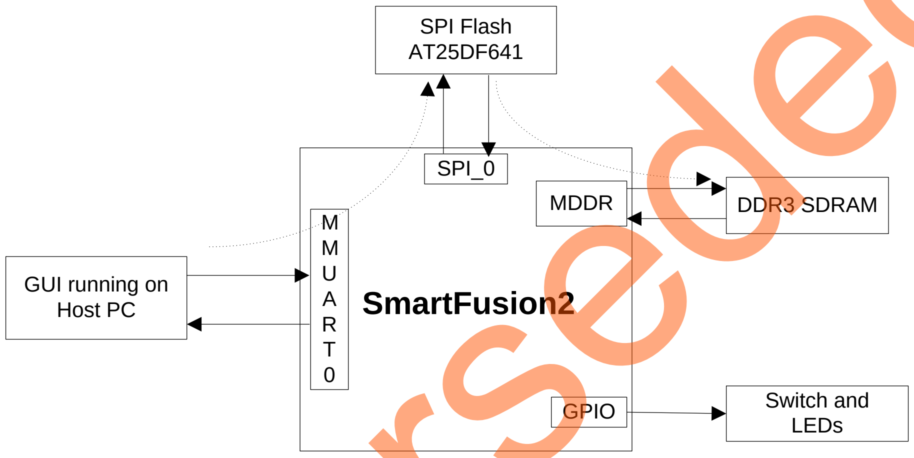


Figure 1. Top Level Block Diagram of the Demo

Code shadowing is a booting method that is used to execute an image from external, faster, volatile memories (DRAM). It is the process of copying the code from non-volatile memory to volatile memory for execution.

Code shadowing is required when the non-volatile memory associated with the processor does not support random access to the code for execute-in-place, or there is insufficient nonvolatile random access memory. In performance critical applications, execution speed can be improved by code shadowing where code is copied to higher throughput RAM for faster execution.

Single data rate (SDR)/DDR SDRAM memories are used in applications that have a large application executable image and require higher performance. Typically, the large executable images are stored in nonvolatile memory such as NAND flash or SPI flash and copied to volatile memory such as SDR/DDR SDRAM memory at power up for execution.

SmartFusion2 SoC FPGA device integrates fourth generation flash-based FPGA fabric, an ARM® Cortex™-M3 processor, and high performance communication interfaces on a single chip. The high speed memory controllers in the SmartFusion2 SoC FPGA device are used to interface with the external DDR2/DDR3/LPDDR memories. The DDR2/DDR3 memories can be operated at maximum speed of 333 MHz. The Cortex-M3 processor can directly execute the instructions from external DDR memory through the microcontroller subsystem (MSS) DDR (MDDR). The FPGA cache controller and MSS DDR bridge handles the data flow for a better performance.

Demo Requirements

Hardware and Software Requirements

The hardware and software required to run the demo are listed in the below Table 1.

Table 1. Required Hardware and Software to Run the Demo

Hardware	**Version**
SmartFusion2 Development Kit	Rev C or later
FlashPro4 programmer	
USB A to Mini - B USB cable	
12 V Adapter	
Software	
FlashPro Programming Software	11.3
USB to UART drivers	
Microsoft .NET Framework 4 client for launching demo GUI	
Operating system	Windows XP SP2 – 32-bit/64-bit Windows 7 – 32-bit/64-bit

Design Files

The design files for this demo can be downloaded from the Microsemi website:
http://soc.microsemi.com/download/rsc/?f=SF2_CodeShadowing_DDR3_DF

Design files include:

1. Libero® System-on-Chip (SoC)
2. Programming files
3. GUI executable
4. Sample application images
5. Linker scripts
6. Readme file

Refer to the Readme.txt file provided in the design files for the complete directory structure.

Demo Design Description

This demo design implements code shadowing technique to boot the application image from DDR memory. This demo design also provides host interface over SmartFusion2 SoC FPGA multi-mode universal asynchronous/synchronous receiver/transmitter (MMUART) to load the target application executable image into SPI flash connected to the MSS SPI0 interface.

The code shadowing is implemented in two methods:

1. Multi-stage boot process method using Cortex-M3 processor
2. Hardware boot engine method using FPGA fabric

Multi-stage Boot Process Method

In this method, the application image is executed from external DDR memories in two boot stages. In the first boot stage Cortex-M3 processor boots the soft boot loader from eNVM which performs the code image transfer from SPI flash to DDR memory. In the second boot stage, Cortex-M3 processor boots the application image from DDR memory.

This demo design implements a boot loader program to load the target application executable image from SPI flash to DDR memory for execution. The boot loader program running from the embedded nonvolatile memory (eNVM) jumps to the target application in the DDR memory once the target application image is copied to DDR memory. Figure 2 shows the detailed block diagram of the demo design.

The MDDR is configured for DDR3 at 320 MHz speed. The Appendix A – DDR Configurations shows the important register values that need to be configured for accessing DDR3. In the design, the DDR is configured before executing the main() function.

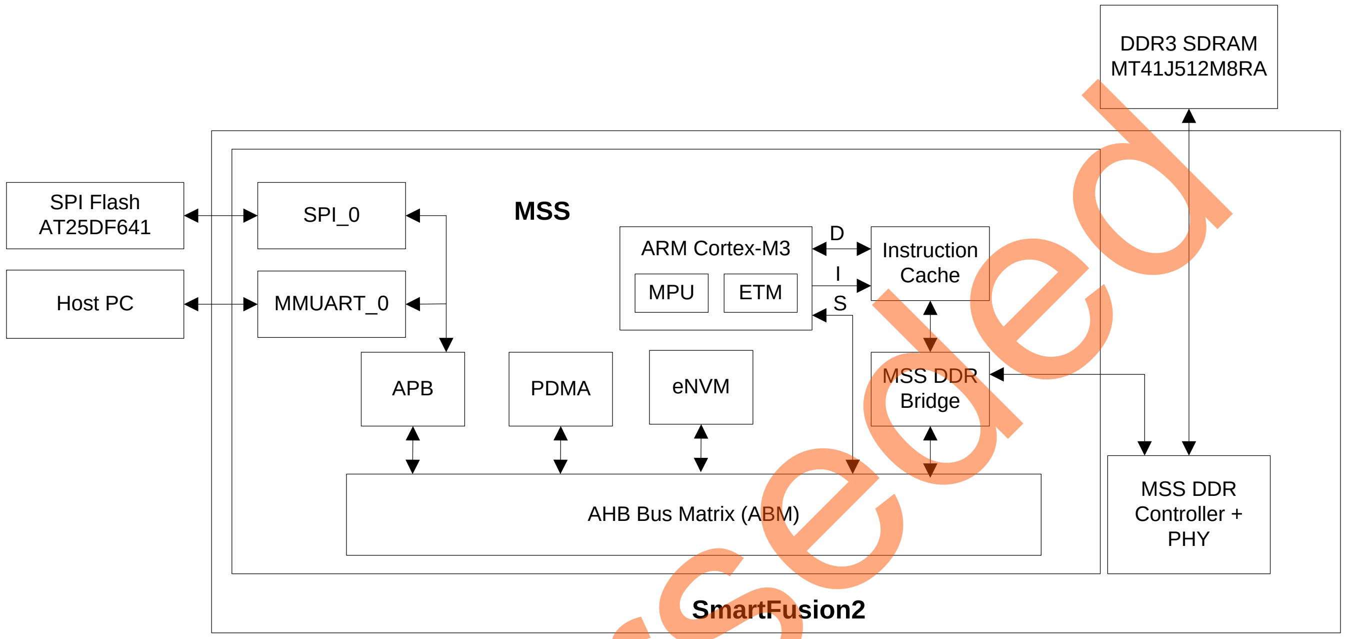


Figure 2. Code Shadowing – Multi Stage Boot Process Demo Block Diagram

Bootloader

This is the major part of the multi-stage boot process code shadowing demo that copies and boots the code from the DDR memory. The boot loader performs the following operations:

1. Copying the target application image from SPI flash memory to DDR
2. Remapping the DDR memory starting address from 0xA0000000 to 0x00000000 by configuring to the DDR_CR register.
3. Initializing the Cortex-M3 processor stack pointer as per the target application. The first location of the target application vector table contains the stack pointer value. The vector table of the target application is available starting from the address 0x00000000.
4. Loads the Program Counter (PC) to reset handler of the target application for executing the target application image from the DDR memory. Reset handler address of the target application is available in the vector table at the address 0x00000004.

The demo design architecture is described in Figure 3.

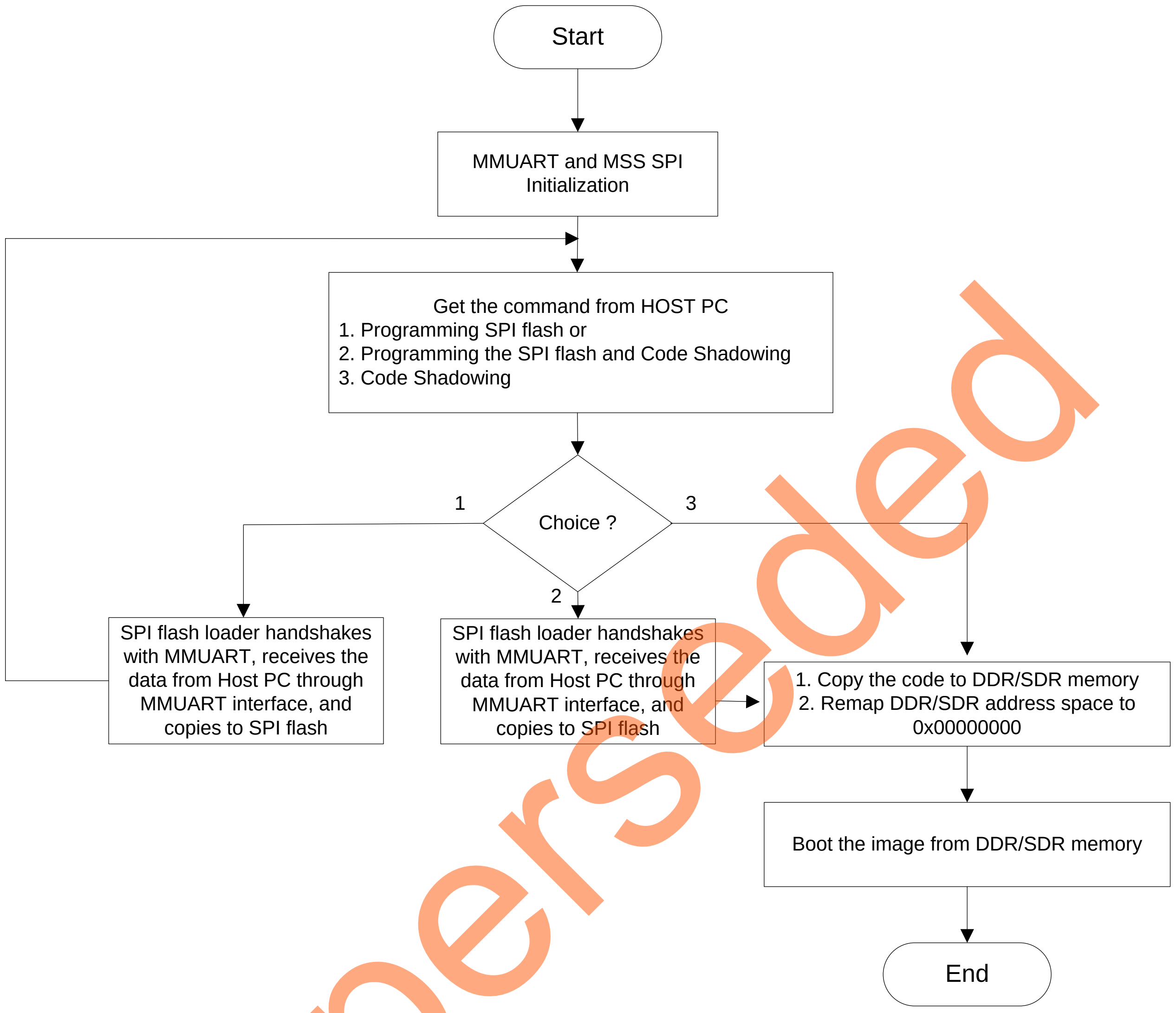

Start
MMUART and MSS SPI Initialization
Get the command from HOST PC
1. Programming SPI flash or
2. Programming the SPI flash and Code Shadowing
3. Code Shadowing
Choice ?
1
2
3
SPI flash loader handshakes with MMUART, receives the data from Host PC through MMUART interface, and copies to SPI flash
SPI flash loader handshakes with MMUART, receives the data from Host PC through MMUART interface, and copies to SPI flash
1. Copy the code to DDR/SDR memory
2. Remap DDR/SDR address space to 0x00000000
Boot the image from DDR/SDR memory
End

Figure 3. Design Flow for Multi-Stage Boot Process Method

Hardware Boot Engine Method

In this method, the Cortex-M3 directly boots the application image from external DDR memories. The hardware boot engine copies the application image from SPI flash to DDR memory before releasing the Cortex-M3 processor reset. After releasing the reset, the Cortex-M3 processor can boot directly from DDR memory. This method requires less boot-up time than multi-stage boot process as it avoids multiple boot stages and copies application image to DDR memory in less time.

This demo design implements boot engine logic in FPGA fabric to copy the target application executable image from SPI flash to the DDR memory for execution. This demo design also implements SPI flash loader, which can be executed by Cortex-M3 processor to load the target application executable image into SPI flash using the provided host interface over SmartFusion2 SoC FPGA MMUART_0. The DIP switch1 on the SmartFusion2 development kit can be used to select whether to program the SPI flash or to execute the code from DDR memory.

If the executable target application is available in SPI flash, the code shadowing from SPI flash to DDR memory is started on device power-up. The boot engine initializes the MDDR, copies the Image from SPI flash to DDR memory, and remaps the DDR memory space to 0x00000000 by keeping the Cortex-M3 processor in reset. After boot engine releases the Cortex-M3 reset, the Cortex-M3 executes the target application from DDR memory.

Figure 4 shows the detailed block diagram of the demo design. The FIC_0 is configured in Slave mode to access the MSS SPI_0 from FPGA fabric AHB master. The MDDR AXI interface (DDR_FIC) is enabled to access the DDR memory from FPGA fabric AXI master.

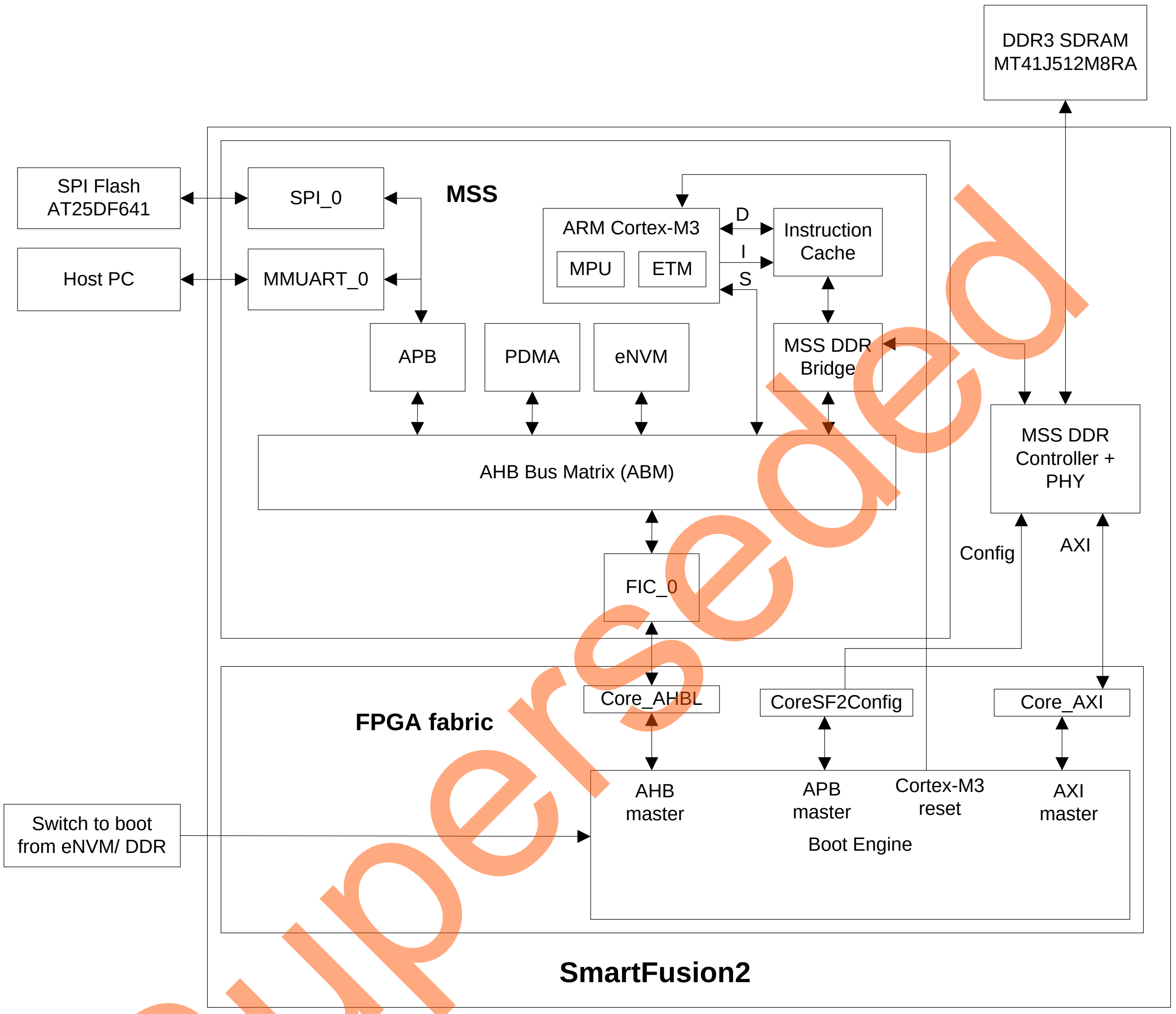


Figure 4. Code Shadowing – Hardware Boot Engine Demo Block Diagram

Boot Engine

This is the major part of the code shadowing demo that copies the application image from SPI flash to the DDR memory. The boot engine performs the following operations:

1. Initializing MDDR for accessing DDR3 at 320 MHz by keeping the Cortex-M3 processor in reset.
2. Copying the target application image from SPI flash memory to DDR memory using the AXI master in the FPGA fabric through MDDR AXI interface.
3. Remapping the DDR memory starting address from 0xA0000000 to 0x00000000 by writing to the DDR_CR register.
4. Releasing reset to Cortex-M3 processor to boot from DDR memory.

The demo design architecture is described in Figure 5.

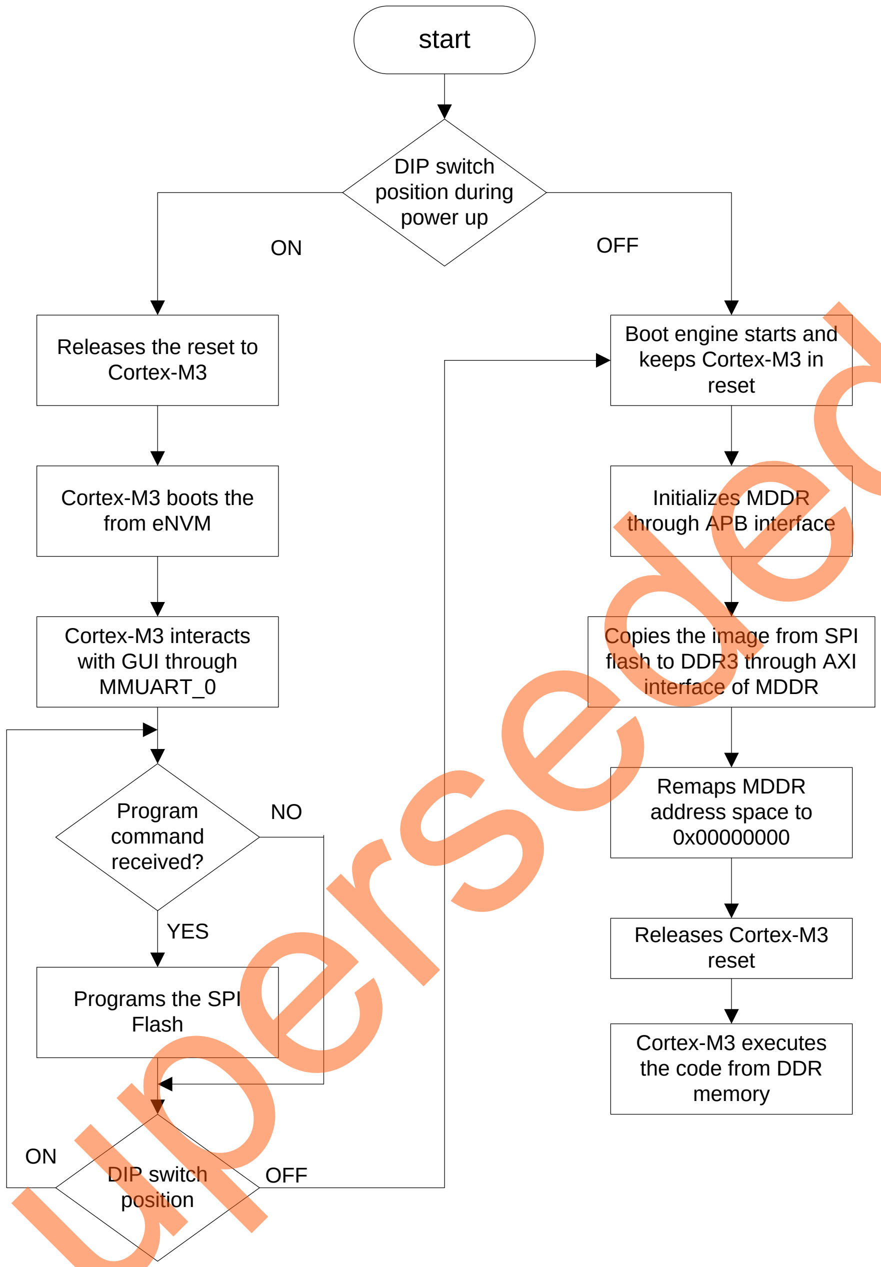

start
DIP switch position during power up
ON
OFF
Releases the reset to Cortex-M3
Cortex-M3 boots the from eNVM
Cortex-M3 interacts with GUI through MMUART_0
Program command received?
NO
YES
Programs the SPI Flash
ON
DIP switch position
OFF
Boot engine starts and keeps Cortex-M3 in reset
Initializes MDDR through APB interface
Copies the image from SPI flash to DDR3 through AXI interface of MDDR
Remaps MDDR address space to 0x00000000
Releases Cortex-M3 reset
Cortex-M3 executes the code from DDR memory

Figure 5. Design Flow for Hardware Boot Engine Method

Creating Application Image Targeting for DDR Memory

An image that can be executed from the DDR memory is required to run the demo. You need to use the "production-execute-in-place-externalDDR.ld" linker description file that is included in the design files to build the application image. This linker description file defines the DDR memory starting address as 0x00000000 since the bootloader/boot engine performs DDR memory remapping from 0xA0000000 to 0x00000000. This linker script creates an application image with instructions, data, and BSS sections in memory whose starting address is 0x00000000. A simple light-emitting diode (LED) blinking, timer and switch based interrupt generation application image file is provided for this demo.

SPI Flash Loader

The SPI flash loader is implemented to load the on-board SPI flash memory with the executable target application image from the host PC through the MMUART_0 interface. The Cortex-M3 processor makes a buffer for the data coming over the MMUART_0 interface and initiates the peripheral DMA (PDMA) to write the buffered data into SPI flash through the MSS_SPI0.

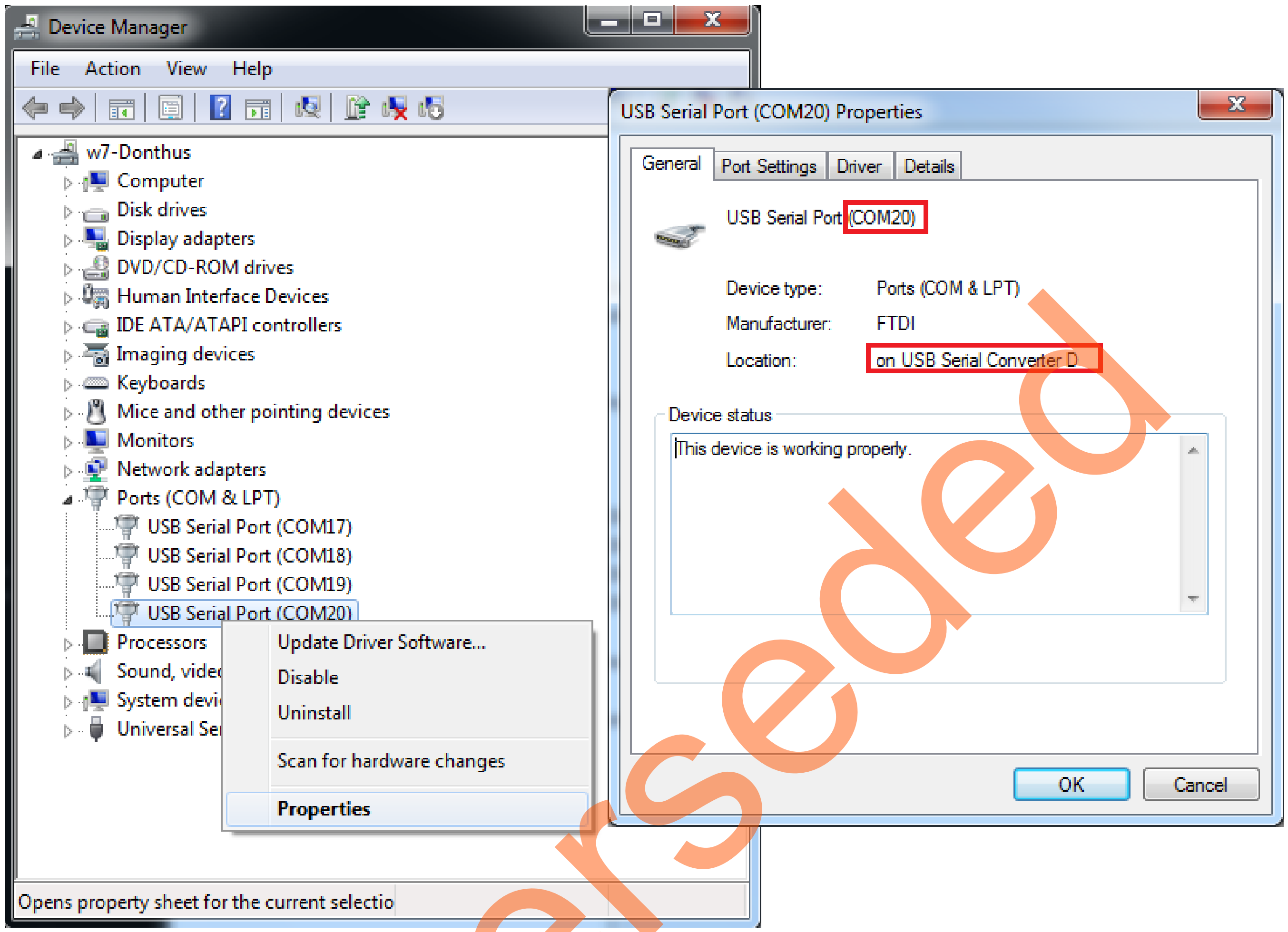
Running the Demo

The demo shows how to load the application image in the SPI flash and execute that application image from external DDR memories. This demo provides an example application image “sample_image_DDR3.bin”. This image shows the welcome messages and timer interrupt message on the serial console and blinks LED1 to LED8 on the SmartFusion2 Development Kit. To see the GPIO interrupt messages on the serial console, press **SW2** or **SW5** switch.

Demo Setup

1. Connect the FlashPro4 programmer to the J59 connector of SmartFusion2 SoC FPGA Development Kit.
2. Connect one end of the USB mini-B cable to the J24 connector provided on the SmartFusion2 SoC FPGA Development Kit. Connect the other end of the USB cable to the host PC.

   Make sure that the USB to UART bridge drivers are automatically detected (can be verified in the Device Manager), as shown in Figure 6. From the detected four COM ports select the one which location on its properties window should be as “on USB Serial Converter D”. Note down the COM port number for serial port configuration and ensure that the COM port Location is specified as 'on USB serial Converter D' as shown in Figure 6.



USB Serial Port (COM20)

Device type: Ports (COM & LPT)

Manufacturer: FTDI

Location: on USB Serial Converter D

Device status

This device is working properly.

Figure 6. USB to UART Bridge Drivers

3. If USB to UART bridge drivers are not installed, download and install the drivers from http://www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip.
4. Connect the jumpers on the SmartFusion2 SoC FPGA Development Kit, as shown in Table 2. While making the jumper connections the power supply switch SW7 on the board should be in **OFF** position.

Table 2. SmartFusion2 SoC FPGA Development Kit Jumper Settings

Jumper	**Pin (from)**	**Pin (to)**
J70, J93, J94, J117, J123, J142, J157, J160, J167, J225, J226, J227	1 (default)	2
J2	1 (default)	3
J23,	2 (default)	3
For UART Communication		
J129, J133	2	3
For SPI to SPI Flash Connection		
J110, J118, J119, J121	1 (default)	2

5. Connect the power supply to the J18 connector.

Figure 7 shows the board setup for running the code shadowing from SPI flash to DDR3 demo on the SmartFusion2 SoC FPGA Development Kit.

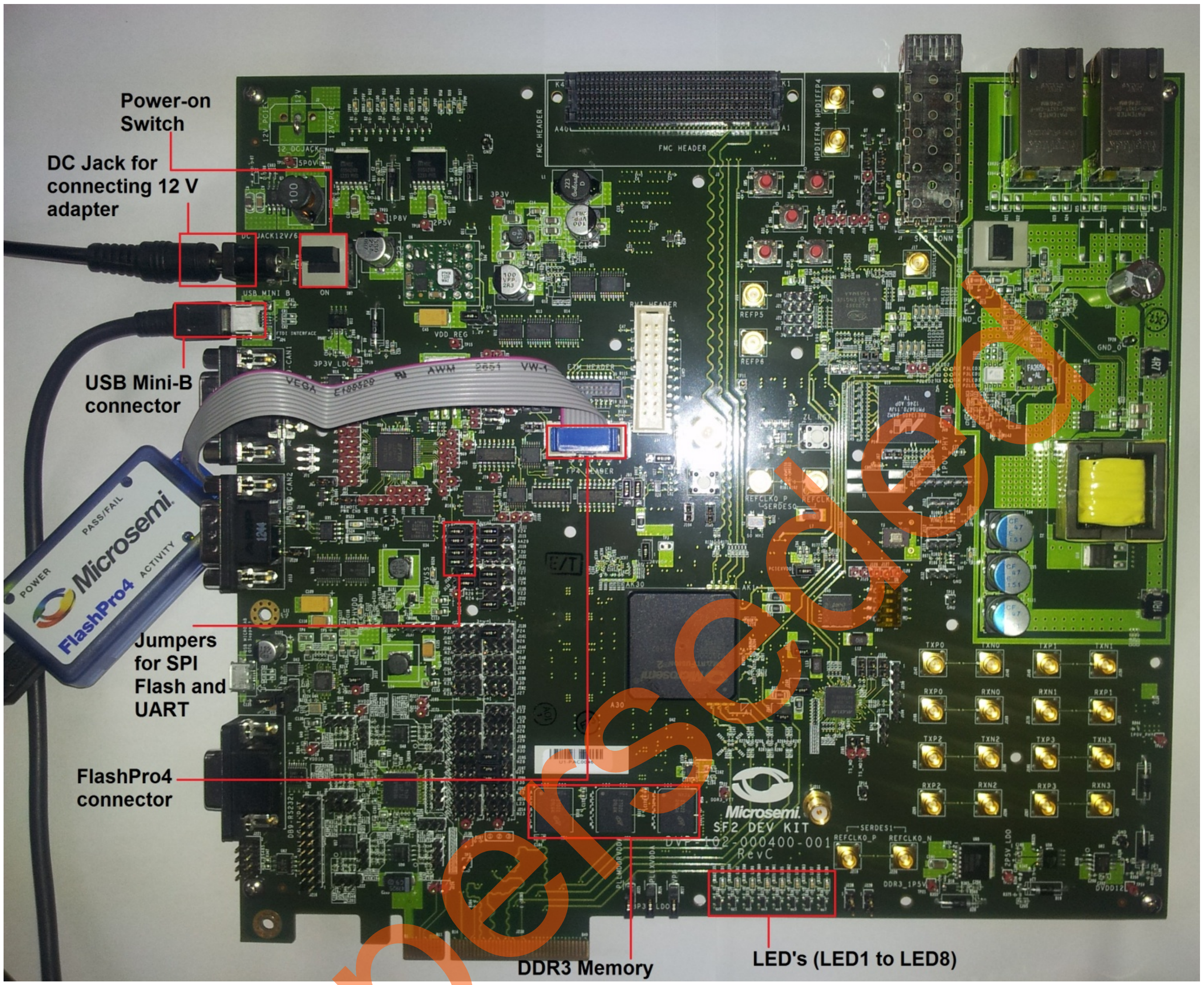
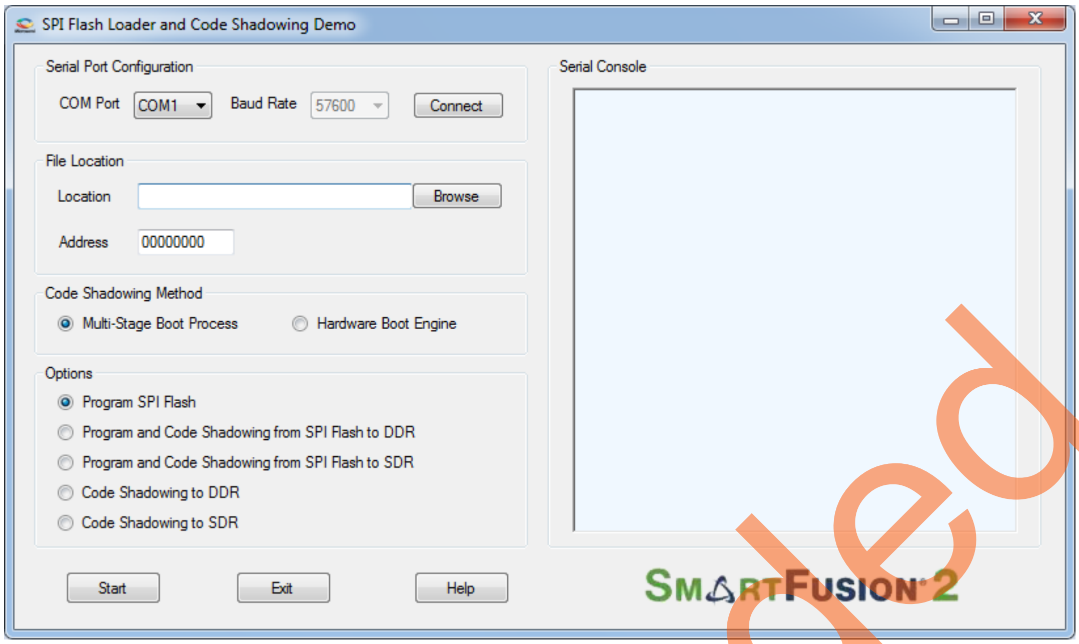


Figure 7. SmartFusion2 SoC FPGA Development Kit Setup

SPI Flash Loader and Code Shadowing Demo GUI

This is required to run the code shadowing demo. SPI Flash Loader and Code Shadowing Demo GUI is a simple graphic user interface that runs on the host PC to program the SPI flash and runs the code shadowing demo on the SmartFusion2 SoC FPGA Development Kit. UART is used as the underlining communication protocol between the host PC and SmartFusion2 SoC FPGA Development Kit. It also provides the serial console section to print the debug messages received from the application over the UART interface. Figure 8 shows the SPI Flash Loader and Code Shadowing Demo GUI.



Serial Port Configuration
COM Port: COM1
Baud Rate: 57600

File Location
Location:
Address: 00000000

Code Shadowing Method
[✓ Multi-Stage Boot Process] Hardware Boot Engine

Options
[✓ Program SPI Flash]
Program and Code Shadowing from SPI Flash to DDR
Program and Code Shadowing from SPI Flash to SDR
Code Shadowing to DDR
Code Shadowing to SDR

Figure 8. SPI Flash Loader and Code Shadowing Demo GUI

The GUI supports the following features:

- **Program SPI Flash**: Programs the image file into the SPI flash.
- **Program and Code Shadowing from SPI Flash to DDR**: Programs the image file into SPI flash, copies it to the DDR memory, and boots the image from the DDR memory.
- **Program and Code Shadowing from SPI Flash to SDR**: Programs the image file into SPI flash, copies it to the SDR memory, and boots the image from the SDR memory.
- **Code Shadowing to DDR**: Copies the existing image file from SPI flash to the DDR memory and boots the image from the DDR memory.
- **Code Shadowing to SDR**: Copies the existing image file from SPI flash to the SDR memory and boots the image from the SDR memory.

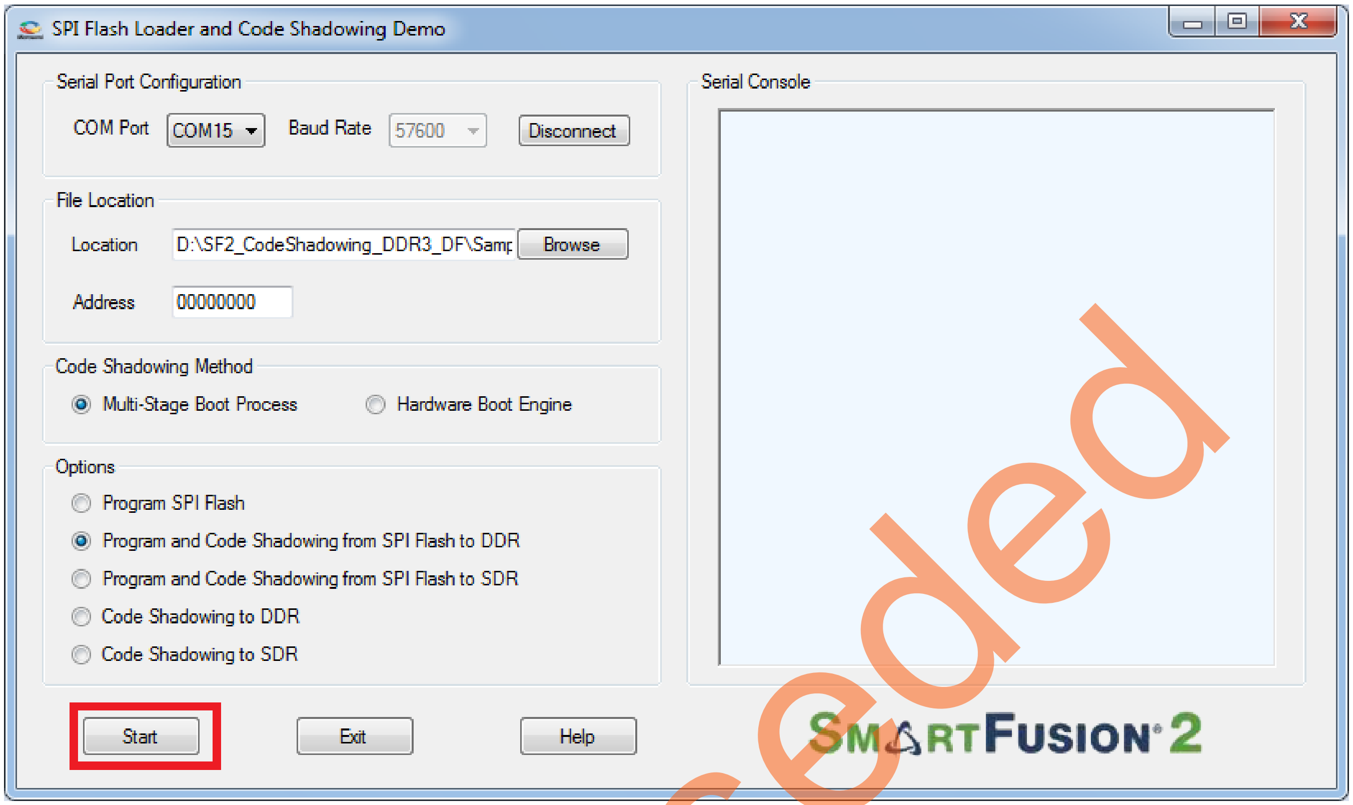
Click **Help** for more information on the GUI.

Running the Multi-Stage Boot Process Method Design

1. Change the power supply switch SW7 to **ON**.
2. Program the SmarFusion2 SoC FPGA device with the programming file provided in the design files (SF2_CodeShadowing_DDR3_DF\Programming Files\MultiStageBoot_meothod\CodeShadowing_top.stp using the FlashPro design software.
3. Launch the **SPI Flash Loader and Code Shadowing Demo** GUI executable file available in the design files (SF2_CodeShadowing_DDR3_DF\GUI Executable\SF2_FlashLoader.exe).

4. Select the appropriate COM port (to which the USB Serial drivers are pointed) from the **COM Port** drop-down list.
5. Click **Connect**. After establishing the connection, **Connect** changes to **Disconnect**.
6. Click **Browse** to select the example target executable image file provided with the design files (SF2_CodeShadowing_DDR3_DF/Sample Application Images/sample_image_DDR3.bin).

   Note: To generate the application image bin file, refer to Appendix-B – Generating Executable Bin File.
7. Keep the starting address of the SPI flash memory as default at 0x00000000.
8. Select the **Program and Code Shadowing from SPI Flash to DDR** option.
9. Click **Start** as shown in Figure 9 to load the executable image into SPI flash and code shadowing from DDR memory.



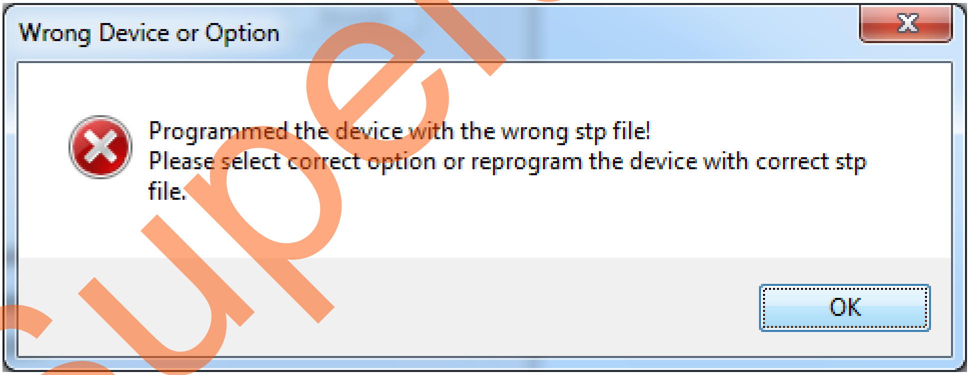
COM Port: COM15
Baud Rate: 57600
Location: D:\SF2_CodeShadowing_DDR3_DF\Samp
Address: 00000000

Code Shadowing Method: [✓ Multi-Stage Boot Process] Hardware Boot Engine

Options:
- Program SPI Flash
- [✓ Program and Code Shadowing from SPI Flash to DDR]
- Program and Code Shadowing from SPI Flash to SDR
- Code Shadowing to DDR
- Code Shadowing to SDR

Figure 9. Starting the Demo

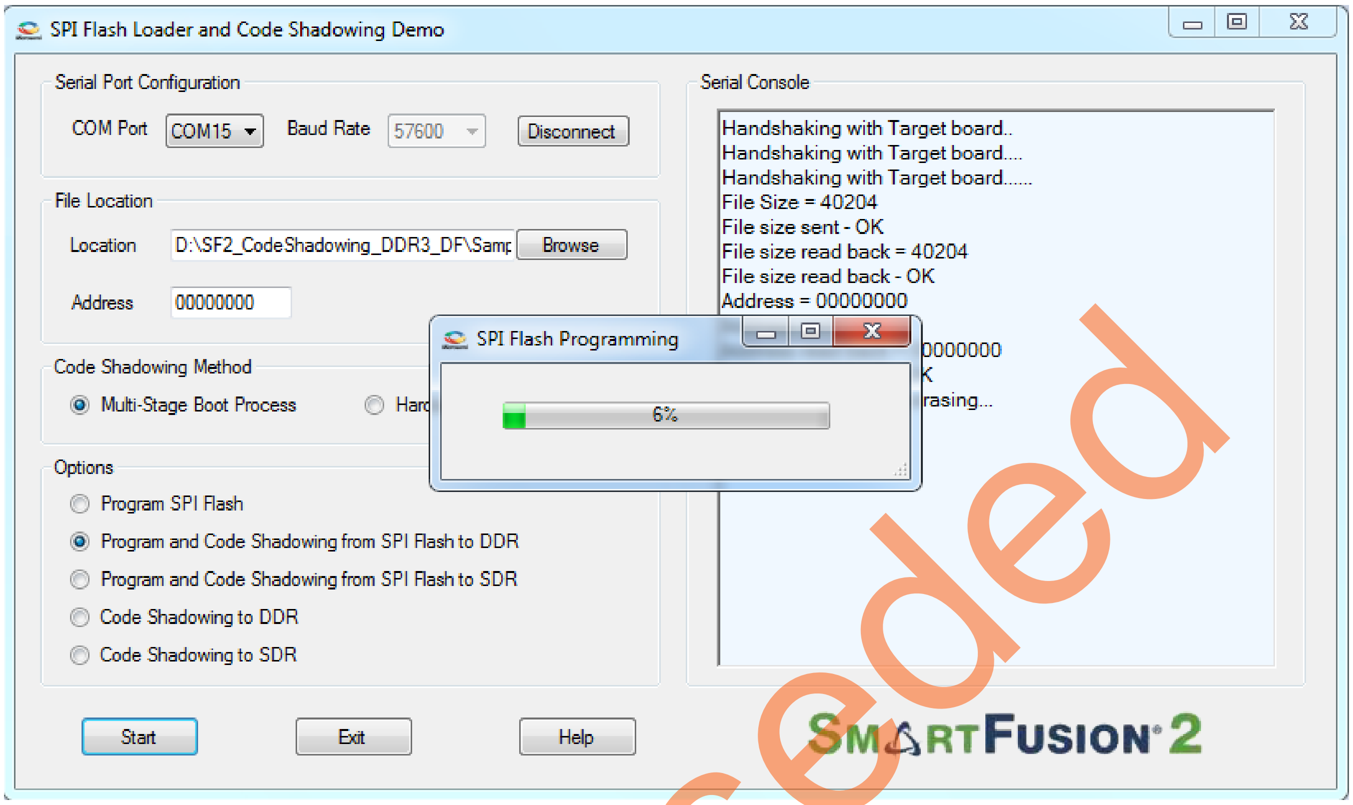
10. If the SmartFusion2 SoC FPGA device is programmed with a STAPL file in which MDDR is not configured for DDR memory then it shows an error message, as shown in Figure 10.



Programmed the device with the wrong stp file!
Please select correct option or reprogram the device with correct stp file.

Figure 10. Wrong Device or Option Message

11. The Serial Console section on the GUI shows the debug messages and starts programming SPI flash on successfully erasing the SPI flash. The GUI shows the status of SPI flash writing, as shown in Figure 11.



COM Port: COM15
Baud Rate: 57600
Location: D:\SF2_CodeShadowing_DDR3_DF\Samp
Address: 00000000
Code Shadowing Method: [✓ Multi-Stage Boot Process] Hard

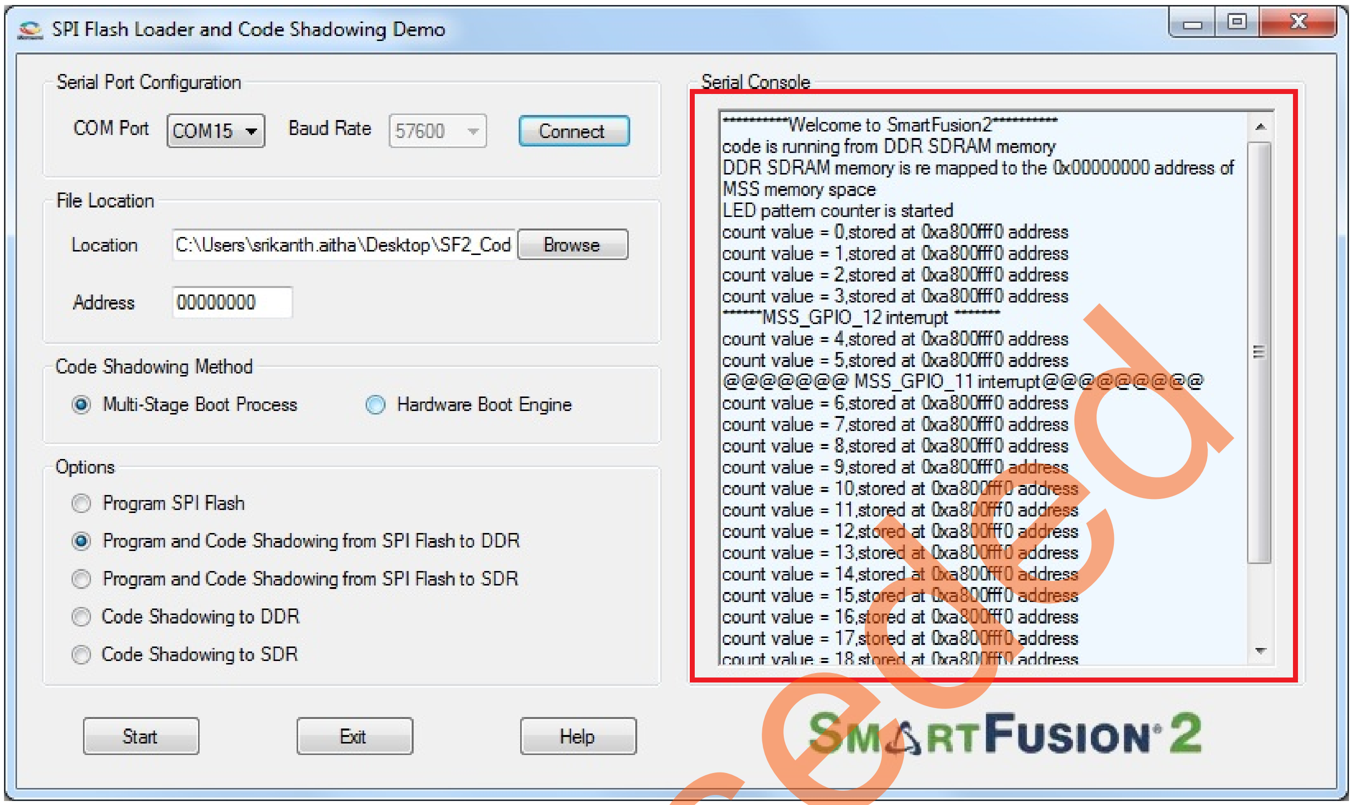
Options:
- Program SPI Flash
- [✓ Program and Code Shadowing from SPI Flash to DDR]
- Program and Code Shadowing from SPI Flash to SDR
- Code Shadowing to DDR
- Code Shadowing to SDR

Serial Console:

```
Handshaking with Target board..
Handshaking with Target board....
Handshaking with Target board......
File Size = 40204
File size sent - OK
File size read back = 40204
File size read back - OK
Address = 00000000
0000000
K
rasing...
```

Figure 11. Flash Loading

12. On programming the SPI flash successfully, the bootloader running on SmartFusion2 SoC FPGA copies the application image from SPI flash to the DDR memory and boots the application image. If the provided image sample_image_DDR3.bin is selected, the serial console shows the welcome messages, switch interrupt and timer interrupt messages as shown in Figure 12 and Figure 13, and a running LED pattern is displayed on LED1 to LED8 on the SmartFusion2 SoC FPGA Development Kit. Press **SW2** and **SW5** switch to see interrupt messages on serial console.


SPI Flash Loader and Code Shadowing Demo

Serial Port Configuration — COM Port: COM15; Baud Rate: 57600

File Location — Location: C:\Users\srikanth.aitha\Desktop\SF2_Cod; Address: 00000000

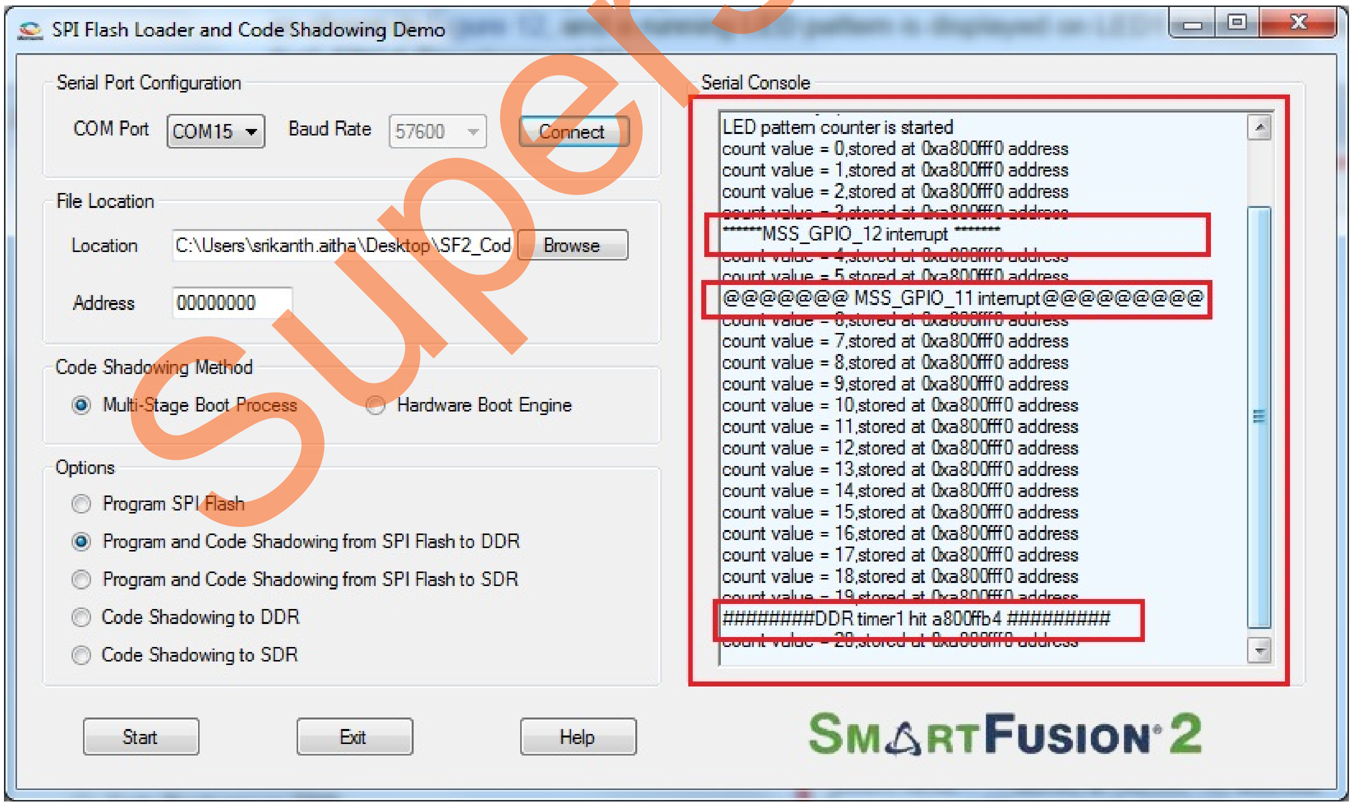
Code Shadowing Method: [✓ Multi-Stage Boot Process] Hardware Boot Engine

Options: Program SPI Flash; [✓ Program and Code Shadowing from SPI Flash to DDR]; Program and Code Shadowing from SPI Flash to SDR; Code Shadowing to DDR; Code Shadowing to SDR

Serial Console:

```
***********Welcome to SmartFusion2***********
code is running from DDR SDRAM memory
DDR SDRAM memory is re mapped to the 0x00000000 address of
MSS memory space
LED pattern counter is started
count value = 0,stored at 0xa800fff0 address
count value = 1,stored at 0xa800fff0 address
count value = 2,stored at 0xa800fff0 address
count value = 3,stored at 0xa800fff0 address
*******MSS_GPIO_12 interrupt *******
count value = 4,stored at 0xa800fff0 address
count value = 5,stored at 0xa800fff0 address
@@@@@@@ MSS_GPIO_11 interrupt@@@@@@@@@
count value = 6,stored at 0xa800fff0 address
count value = 7,stored at 0xa800fff0 address
count value = 8,stored at 0xa800fff0 address
count value = 9,stored at 0xa800fff0 address
count value = 10,stored at 0xa800fff0 address
count value = 11,stored at 0xa800fff0 address
count value = 12,stored at 0xa800fff0 address
count value = 13,stored at 0xa800fff0 address
count value = 14,stored at 0xa800fff0 address
count value = 15,stored at 0xa800fff0 address
count value = 16,stored at 0xa800fff0 address
count value = 17,stored at 0xa800fff0 address
count value = 18,stored at 0xa800fff0 address
```

Figure 12. Running the Target Application Image from DDR3 Memory


SPI Flash Loader and Code Shadowing Demo

Serial Port Configuration — COM Port: COM15; Baud Rate: 57600

File Location — Location: C:\Users\srikanth.aitha\Desktop\SF2_Cod; Address: 00000000

Code Shadowing Method: [✓ Multi-Stage Boot Process] Hardware Boot Engine

Options: Program SPI Flash; [✓ Program and Code Shadowing from SPI Flash to DDR]; Program and Code Shadowing from SPI Flash to SDR; Code Shadowing to DDR; Code Shadowing to SDR

Serial Console:

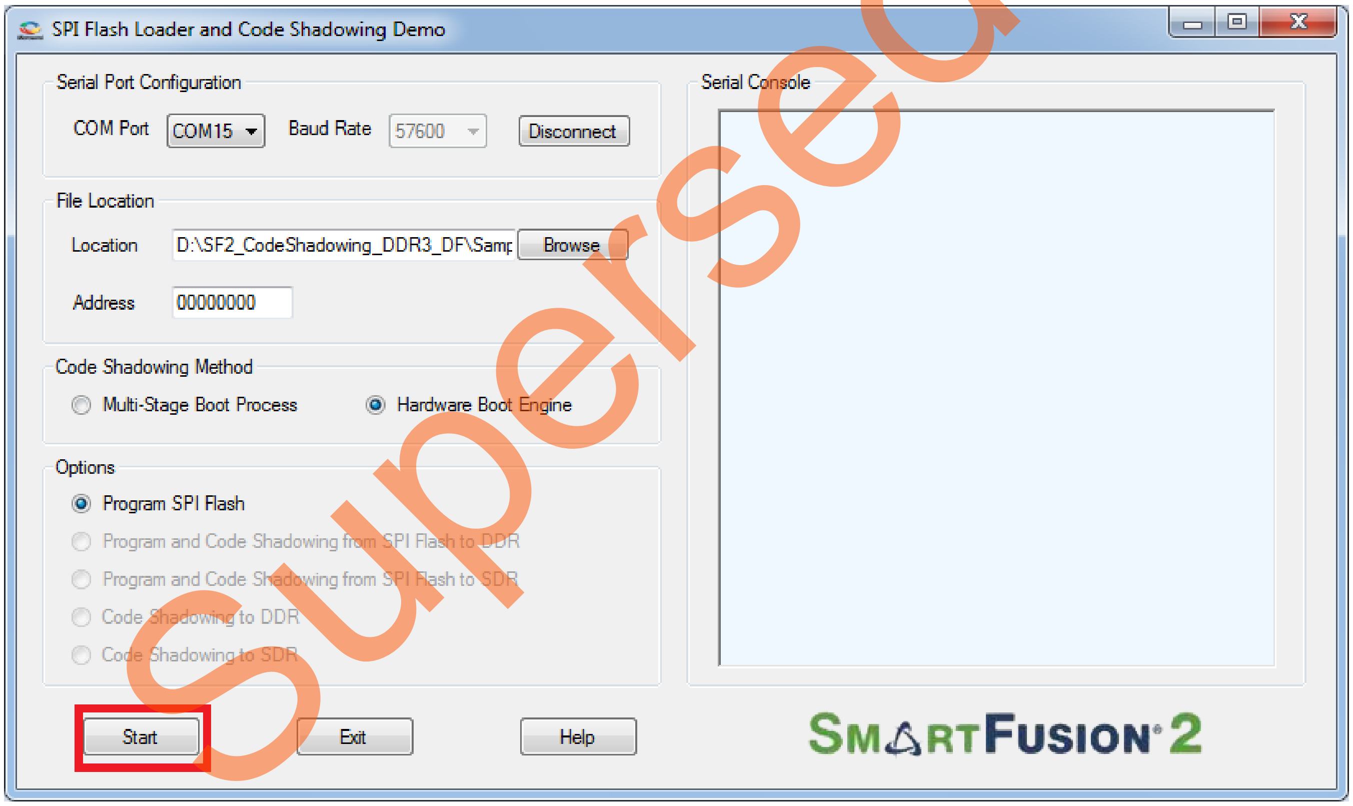
```
LED pattern counter is started
count value = 0,stored at 0xa800fff0 address
count value = 1,stored at 0xa800fff0 address
count value = 2,stored at 0xa800fff0 address
count value = 3,stored at 0xa800fff0 address
*******MSS_GPIO_12 interrupt *******
count value = 4,stored at 0xa800fff0 address
count value = 5,stored at 0xa800fff0 address
@@@@@@@ MSS_GPIO_11 interrupt@@@@@@@@@
count value = 6,stored at 0xa800fff0 address
count value = 7,stored at 0xa800fff0 address
count value = 8,stored at 0xa800fff0 address
count value = 9,stored at 0xa800fff0 address
count value = 10,stored at 0xa800fff0 address
count value = 11,stored at 0xa800fff0 address
count value = 12,stored at 0xa800fff0 address
count value = 13,stored at 0xa800fff0 address
count value = 14,stored at 0xa800fff0 address
count value = 15,stored at 0xa800fff0 address
count value = 16,stored at 0xa800fff0 address
count value = 17,stored at 0xa800fff0 address
count value = 18,stored at 0xa800fff0 address
count value = 19,stored at 0xa800fff0 address
#########DDR timer1 hit a800ffb4 ##########
count value = 20,stored at 0xa800fff0 address
```

Figure 13. Timer and Interrupt Messages in Serial Console

Running the Hardware Boot Engine Method Design

1. Change the power supply switch SW7 to **ON**.
2. Program the SmarFusion2 SoC FPGA device with the programming file provided in the design files (SF2_CodeShadowing_DDR3_DF\Programming Files\HWBootEngine_method\CodeShadowing_Fabric.stp using the FlashPro design software.
3. To program the SPI Flash make DIP switch 1 to **ON** position. This selection makes to boot Cortex-M3 from eNVM. Press **SW9** to reset the SmartFusion2 device.
4. Launch the **SPI Flash Loader and Code Shadowing Demo** GUI executable file available in the design files (SF2_CodeShadowing_DDR3_DF\GUI Executable\SF2_FlashLoader.exe).
5. Select the appropriate COM port (to which the USB Serial drivers are pointed) from the COM Port drop-down list.
6. Click **Connect**. After establishing the connection, **Connect** changes to **Disconnect**.
7. Click **Browse** to select the example target executable image file provided with the design files (SF2_CodeShadowing_DDR3_DF/Sample Application Images/sample_image_DDR3.bin).

   Note: To generate the application image bin file, refer to Appendix-B – Generating Executable Bin File.
8. Select **Hardware Boot Engine** option in **Code Shadowing Method**.
9. Select the **Program SPI Fla**sh option from **Options** menu.
10. Click **Start,** as shown in Figure 14 to load the executable image into SPI flash.



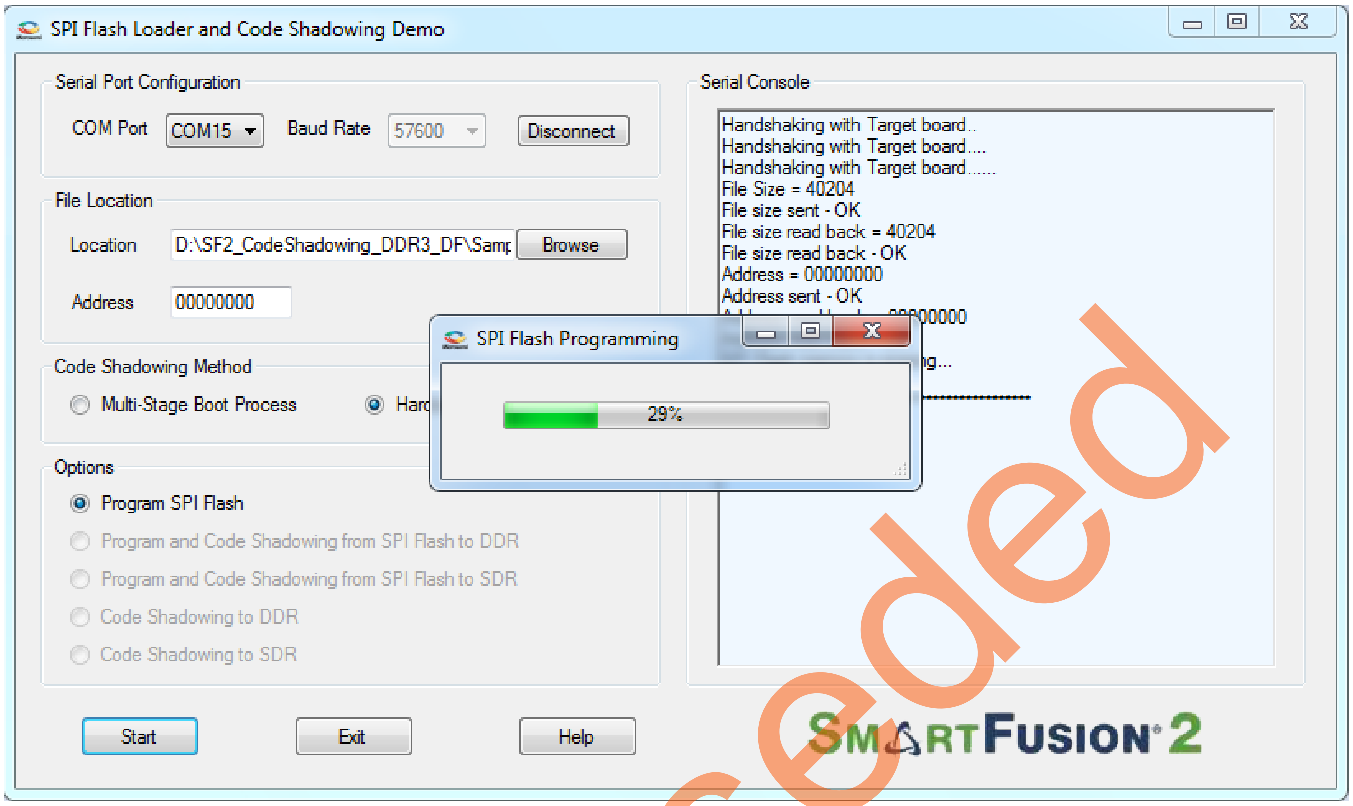
COM Port: COM15
Baud Rate: 57600
Disconnect
Location: D:\SF2_CodeShadowing_DDR3_DF\Samp
Address: 00000000

Multi-Stage Boot Process [✓ Hardware Boot Engine]

[✓ Program SPI Flash]
Program and Code Shadowing from SPI Flash to DDR
Program and Code Shadowing from SPI Flash to SDR
Code Shadowing to DDR
Code Shadowing to SDR

Figure 14. Starting the Demo

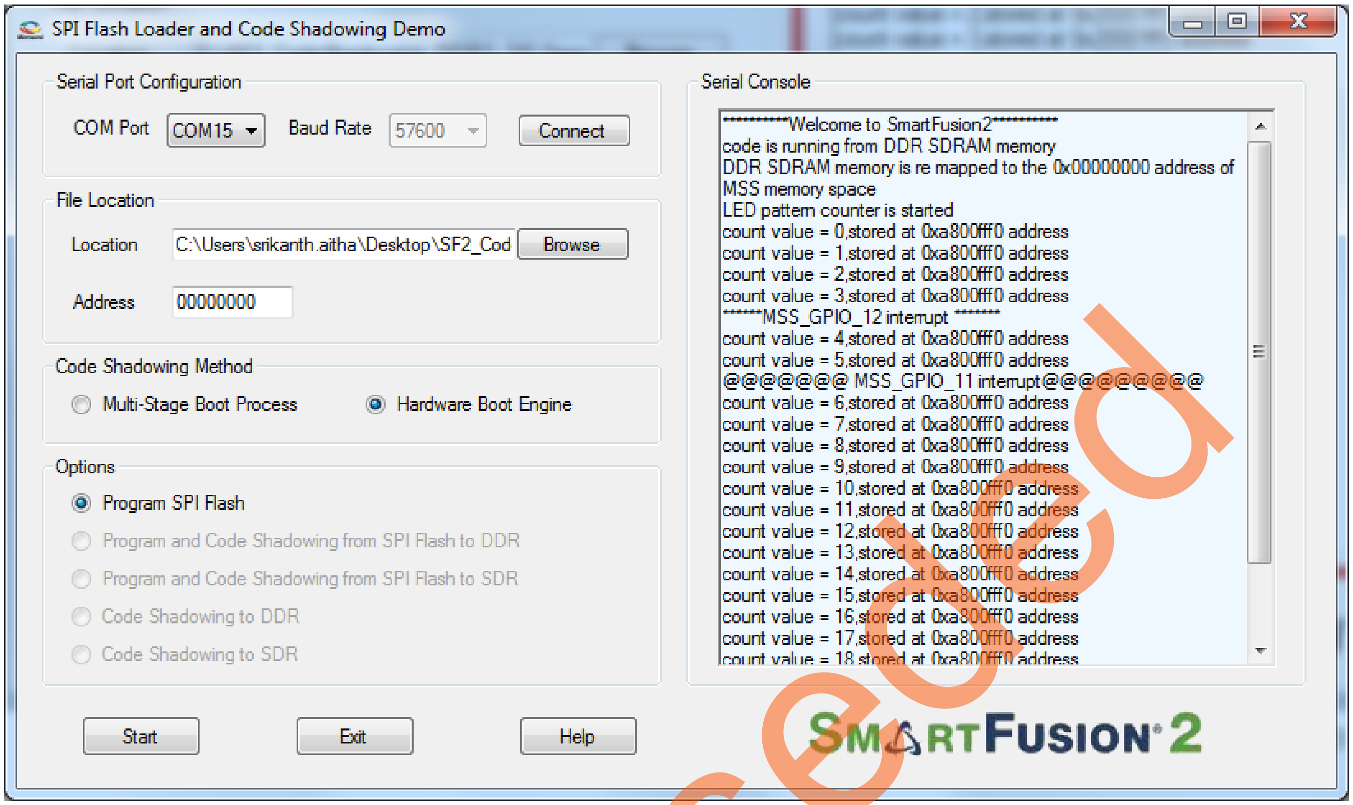
11. The Serial Console section on the GUI shows the debug messages and the status of SPI flash writing, as shown in Figure 15.



```
Handshaking with Target board..
Handshaking with Target board....
Handshaking with Target board......
File Size = 40204
File size sent - OK
File size read back = 40204
File size read back - OK
Address = 00000000
Address sent - OK
```

Figure 15. Flash Loading

12. After programming the SPI flash successfully, change DIP switch 1 to **OFF** position. This selection makes to boot Cortex-M3 processor from DDR memory.
13. Press **SW9** to reset the SmartFusion2 device. The boot engine copies the application image from SPI flash to the DDR memory and releases reset to Cortex-M3, which boots the application image from DDR memory. If the provided image “sample_image_DDR3.bin” is loaded to SPI flash, the serial console shows the welcome messages, switch interrupt (press **SW2** or **SW5**) and timer interrupt messages as shown in Figure 16 and a running LED pattern is displayed on LED1 to LED8 on the SmartFusion2 SoC FPGA Development Kit.



Code Shadowing Method: Multi-Stage Boot Process [✓ Hardware Boot Engine]

Options: [✓ Program SPI Flash]

```
***********Welcome to SmartFusion2**********
code is running from DDR SDRAM memory
DDR SDRAM memory is re mapped to the 0x00000000 address of
MSS memory space
LED pattern counter is started
count value = 0,stored at 0xa800fff0 address
count value = 1,stored at 0xa800fff0 address
count value = 2,stored at 0xa800fff0 address
count value = 3,stored at 0xa800fff0 address
*******MSS_GPIO_12 interrupt *******
count value = 4,stored at 0xa800fff0 address
count value = 5,stored at 0xa800fff0 address
@@@@@@@ MSS_GPIO_11 interrupt@@@@@@@@@
count value = 6,stored at 0xa800fff0 address
count value = 7,stored at 0xa800fff0 address
count value = 8,stored at 0xa800fff0 address
count value = 9,stored at 0xa800fff0 address
count value = 10,stored at 0xa800fff0 address
count value = 11,stored at 0xa800fff0 address
count value = 12,stored at 0xa800fff0 address
count value = 13,stored at 0xa800fff0 address
count value = 14,stored at 0xa800fff0 address
count value = 15,stored at 0xa800fff0 address
count value = 16,stored at 0xa800fff0 address
count value = 17,stored at 0xa800fff0 address
count value = 18,stored at 0xa800fff0 address
```

Figure 16. Running the Target Application Image from DDR3 Memory

Conclusion

This demonstration shows the capability of SmartFusion2 SoC FPGA device to interface with DDR memory and to run the executable image from the DDR memory by shadowing code from SPI flash memory. It also shows two methods of code shadowing implementation on SmartFusion2 device.

Appendix A – DDR Configurations

DDR3 Configurations

Table 3 shows the required MDDR register configurations for operating the DDR3 with clock 320 MHz.

Table 3. MDDR Configurations for Accessing DDR3 Memories at 320 MHz

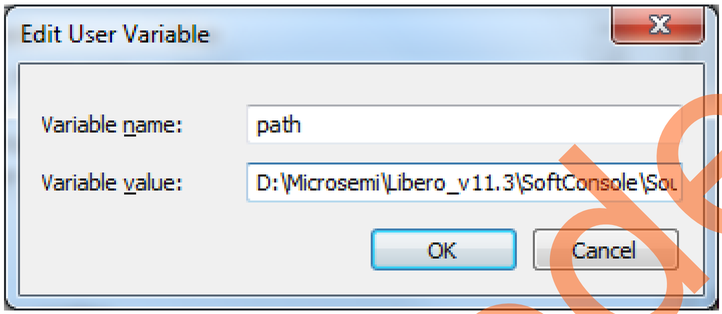
Register Name and Configured Value	Field	Value to be Loaded	Desired Value for MT41J512M8RA
DDRC_DYN_REFRESH_1_CR		**0x27 de**	
	tRFC(min)	0x4F	237 ns
	Speculative refresh	0x1E	90 ns
DDRC_DYN_REFRESH_2_CR		**0x30 f**	
	tRFEI	0x61	0x61(97)*32 clks = 9.3 us
DDRC_INIT_MR_CR		**0x520**	
	Write recovery		3
	DLL reset		Yes
	CAS latency		6
	Burst type		Sequential
	Burst length		8
DDRC_INIT_EMR_CR		**0x44**	
	Additive latency (AL)		CL-1
	Write levelization		Enable
DDRC_DRAM_BANK_TIMING_PARAM_CR			
	tRC	0x33	153 ns
	tFAW	0x20	96 ns
DDRC_DRAM_RD_WR_LATENCY_CR		**0x86**	
	WL		4
	RL		6
DDRC_DRAM_RD_WR_PRE_CR		**0x1E5**	
	Rd2pre	0x15	63 ns
	Wr2pre	0x11	51 ns
DDRC_DRAM_MR_TIMING_PARAM_CR		**0x58**	
	tMOD	0xB	11 clks
DDRC_DRAM_RAS_TIMING_CR		**0x10F**	
	tRAS(max)	0xF	15*1024= 46 us
	tRAS(min)	0x8	24 ns

Register Name and Configured Value	Field	Value to be Loaded	Desired Value for MT41J512M8RA
DDRC_DRAM_RD_WR_TRNARND_TIME_CR		**0x178**	
	Rd2wr	0xB	11 clks
	Wr2rd	0x18	24 clks
DDRC_DRAM_T_PD_CR		**0x33**	
	tXP	3	3 clks
	tCKE	3	3 clks
DDRC_DRAM_BANK_ACT_TIMING_CR		**0x1947**	
	tRP	7	21 ns
	tRRD	4	12 ns
	tCCD	2	2clks
	tRCD	6	18 ns
DDRC_PWR_SAVE_1_CR		**0x506**	
	Clks to power down	3	3*32=96clks
	Self refresh gap	0x14(20)	20*32=640clks
DDRC_ZQ_LONG_TIME_CR		0x200	512 clks
ZQ_SHORT_TIME_CR		0x40	64 clks
DDRC_PERF_PARAM_1_CR		**0x4000**	
	Burst length	0x2	Burst length is 8
HPR_QUEUE_PARAM_1_CR		**0x80F8**	
	XACT_RUN_LENGTH	0x8	8 transactions
	MIN_NON_CRITICAL	0xF	15 clks
	MAX_STARVE	0x1	15 clks
HPR_QUEUE_PARAM_2_CR	MAX_STARVE	0x7	
DDRC_PERF_PARAM_2_CR		**0x0**	
	Burst mode		Sequential

Appendix B – Generating Executable Bin File

The executable bin file is required to program the SPI flash for running the code shadowing demo. To generate the executable bin file from “sample_image_DDR3” SoftConsole, follow the below steps:

1. Build the SoftConsole project with the linkerscript “production-execute-in-place-externalDDR”.
2. Add the SoftConsole installation path, for example, C:\Microsemi\Libero_v11.3\SoftConsole\Sourcery-G++\bin, to the ‘Environment Variables’ as shown in Figure 17.



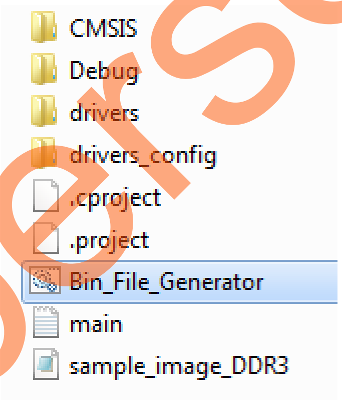
Edit User Variable

Variable name: path

Variable value: D:\Microsemi\Libero_v11.3\SoftConsole\Sou

Figure 17. Adding SoftConsole Installation Path

3. Double-click batch file **Bin-File-Generator** in the SoftConsole/CodeShadowing_MSS_CM3/Sample_image_DDR3 folder, as shown in Figure 18.



CMSIS
Debug
drivers
drivers_config
.cproject
.project
Bin_File_Generator
main
sample_image_DDR3

Figure 18. Bin File Generator

4. The **Bin-File-Generator** creates “sample_image_DDR3.bin” file

List of Changes

Revision	Changes	Page
Revision 4 (May 2014)	Updated the document for Libero SoC 11.3 software release (SAR 56851).	NA
Revision 3 (December 2013)	Updated the document for Libero SoC v11.2 software release (SAR 53019).	NA
Revision 2 (May 2013)	Updated the document for Libero SoC v11.0 software release (SAR 47552).	NA
Revision 1 (March 2013)	Updated the document for Libero SoC v11.0 beta SP1 software release (SAR 45068).	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.*		

Product Support

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

Customer Service

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

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

Customer Technical Support Center

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

Technical Support

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

Website

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

Contacting the Customer Technical Support Center

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

Email

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

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

My Cases

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

Outside the U.S.

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

ITAR Technical Support

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

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (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

50200386-4/05.14