

SmartFusion2 SoC FPGA – Remapping eNVM, eSRAM, and DDR/SDR SDRAM Memories - Libero SoC v11.5

Table of Contents

Purpose	1
Introduction	1
Design Requirements	4
Design Description	4
Hardware Implementation	4
Software Implementation	9
Setting up the Demo Design	18
Conclusion	24
Appendix A: Design Files	25
Appendix B: SmartFusion2 Advanced Development Kit Board	25
List of Changes	26

Purpose

This application note describes the remapping of the following memories to the ARM® Cortex®-M3 processor code region and explains how to execute the program code built with absolute addresses without remapping.

- Embedded nonvolatile memory (eNVM)
- Embedded random access memory (eSRAM)
- Double data rate (DDR)/single data rate (SDR) synchronous dynamic random access memory (SDRAM)

Introduction

SmartFusion®2 system-on-chip (SoC) field programmable gate array (FPGA) devices integrate Cortex-M3 processor, up to 512 KB of eNVM, 64 KB of eSRAM, and memory interfaces for DDR/SDR SDRAM for program code, and data.

The Cortex-M3 processor has a predefined memory map for code space, data space, and system space with dedicated bus interfaces. The desired memory regions of the SmartFusion2 SoC FPGA can be mapped to the Cortex-M3 processor code space for the application program execution. It also explains how to execute the program code built with absolute addresses without remapping.

SmartFusion2 SoC FPGA Booting and Address Space Overview

This application note describes the SmartFusion2 SoC FPGA boot sequence, and how to remap the various memory regions to the Cortex-M3 processor code region and to an optional softcore processor located in the FPGA fabric.

The Cortex-M3 processor is based on ARM architecture v-7M that includes a nested vectored interrupt controller (NVIC) for handling the interrupts, and includes a non-maskable interrupt. The NVIC contains the addresses of the initial stack pointer, exception handlers, and interrupt service routines (ISRs). The first entry in the NVIC must be the initial stack pointer and the second entry must be the address of the reset exception handler. The Cortex-M3 processor eliminates the need for setting up the initial C runtime environment using assembly code. Developers can code entirely in the C language.

The Cortex-M3 processor upon reset reads two words from memory:

- At the address location 0x00000000 for the initial stack pointer
- At the address location 0x00000004 for the address of the reset handler exception

The reset handler performs the basic initialization and execution control which is given to the main application code. This execution flow is shown in [Figure 1](#).

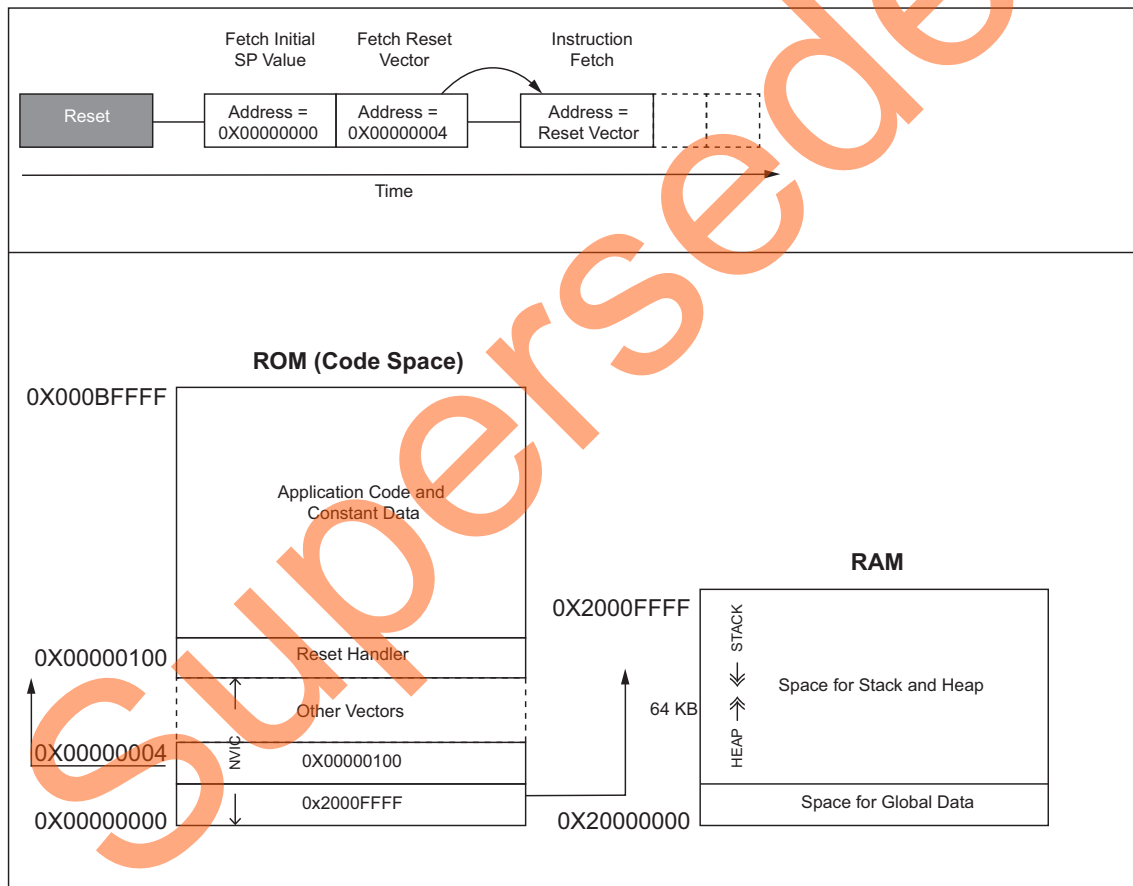


Figure 1 • Cortex-M3 Processor Execution Flow from Reset

SmartFusion2 SoC FPGA: Cortex-M3 Processor Code Space Details

The address range from the 0x00000000 to 0x1FFFFFFF (0.5GB space) is code space for the Cortex-M3 processor. Following are the SmartFusion2 SoC FPGA memory maps for the code/data space:

- On-chip eNVM (from 0x60000000 to 0x6007FFFF) of 512 KB for code and constant data regions
- On-chip eSRAM (from 0x20000000 to 0x2000FFFF) of 64 KB with SECDED for both code and data regions
- On-chip FPGA fabric RAM (FPGA Fabric FIC Region 0). This can be mapped via fabric interface controllers (FIC): FIC 0 or FIC 1. This region can be accessed by system bus for Instructions and data.
- External RAM memory interfaced through DDR or SDR interfaces (from 0xA0000000 to 0xDFFFFFFF) of 1 GB for both code and data regions

Any of the above memory regions with any offset from its base address, can be mapped to the Cortex-M3 processor code region space. On power-on, the eNVM region 0x60000000 is automatically remapped to the Cortex-M3 processor executable region start address (0x00000000). Therefore, for every power-on reset the Cortex-M3 processor fetches the initial stack pointer from 0x00000000 (eNVM address 0x60000000) and address of the reset handler from 0x00000004 (eNVM address 0x60000004). Once the execution control goes to the default reset handler, the boot up sequence executes and execution control jumps to the user boot code.

The user boot code can be at the following locations based on the execution environment:

- **In Release mode:** It must be in read-only memory (ROM) region. The SmartFusion2 SoC FPGA after reset is initialized and remaps the eNVM address 0x60000000 to 0x00000000 of the Cortex-M3 processor address space.
- **In Debug mode:** It can either be in ROM or RAM. Options are in the debugger command window to choose from where to debug (remap to 0x00000000) and in case of Debug mode, the SmartFusion2 SoC FPGA after reset is initialized through the flash bits and remaps the user boot code as follows:
 - eNVM address 0x60000000 to 0x00000000 of the Cortex-M3 processor address space, or
 - eSRAM address 0x20000000 to 0x00000000 of the Cortex-M3 processor address space

From the user boot code there can be multiple independent executable images in various parts of memories. The eNVM address locations can be remapped with any offset, eSRAM address locations with any offset, FPGA fabric RAM, or memory through DDR/SDRAM interface with any offset to the based address 0x00000000 of the Cortex-M3 processor code region.

Design Requirements

Table 1 lists the design requirements.

Table 1 • Design Requirements

Design Requirements	Description
Hardware Requirements	
SmartFusion2 Advanced Development Kit <ul style="list-style-type: none"> FlashPro5 programmer USB A to Mini-B cable 12 V Adapter 	Rev A
Host PC or Laptop	Any 64-bit Windows Operating System
Software Requirements	
Libero® System-on-Chip (SoC)	v11.5
SoftConsole	v3.4SP1
USB to UART drivers	-
One of the following serial terminal emulation programs: <ul style="list-style-type: none"> HyperTerminal TeraTerm PuTTY 	-

Design Description

The design examples in this application note use MMUART_0, GPIO, eSRAM, DDR and eNVM memory controllers. In the design examples, the System Builder Clock section is configured as shown in Figure 6 to run the M3_CLK at 111 MHz which drives the clock to the Cortex-M3 processor. The independent executable images are created with required memory map. These executable images can be remapped to the starting address of the Cortex-M3 processor code space, or can be made executable for the Cortex-M3 processor. The implementation details are explained in hardware and software implementation sections.

Hardware Implementation

The hardware implementation involves configuring MSS, Fabric, clocks and oscillator using System Builder. Figure 2 shows the top-level SmartDesign of the application.

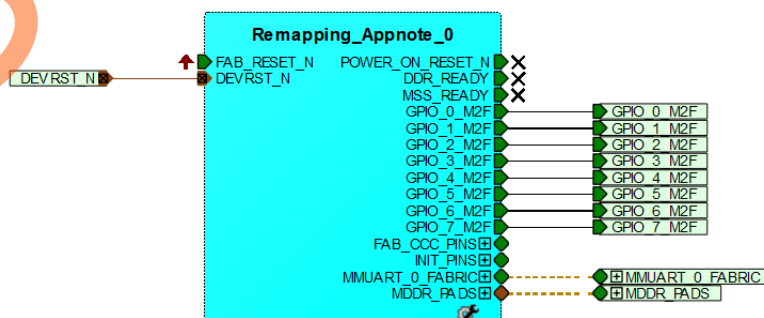


Figure 2 • Top-Level SmartDesign

The MDDR is configured for DDR3 at 333 MHz speed. Figure 3 and Figure 4 show the MSS MDDR configuration settings. Refer to "Appendix A: Design Files" on page 25 for DDR configuration file.

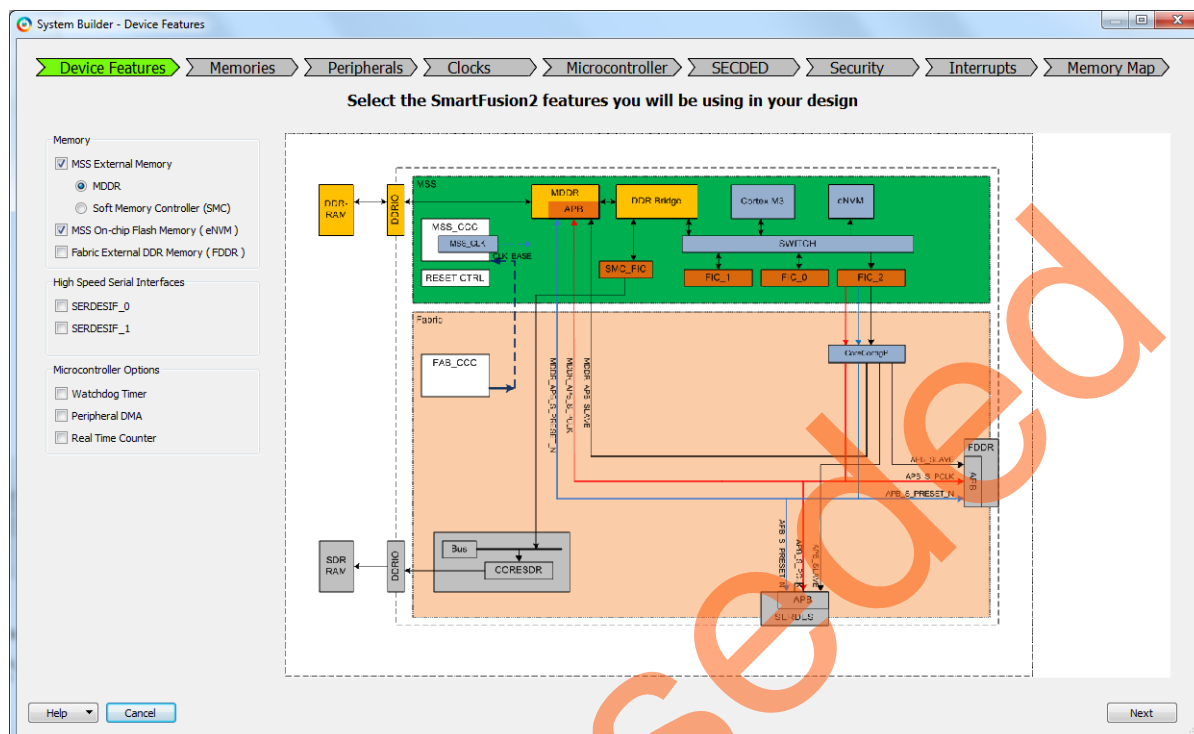


Figure 3 • Select MDDR



Add the eNVM user clients in ENVM configurator as shown in Figure 5.

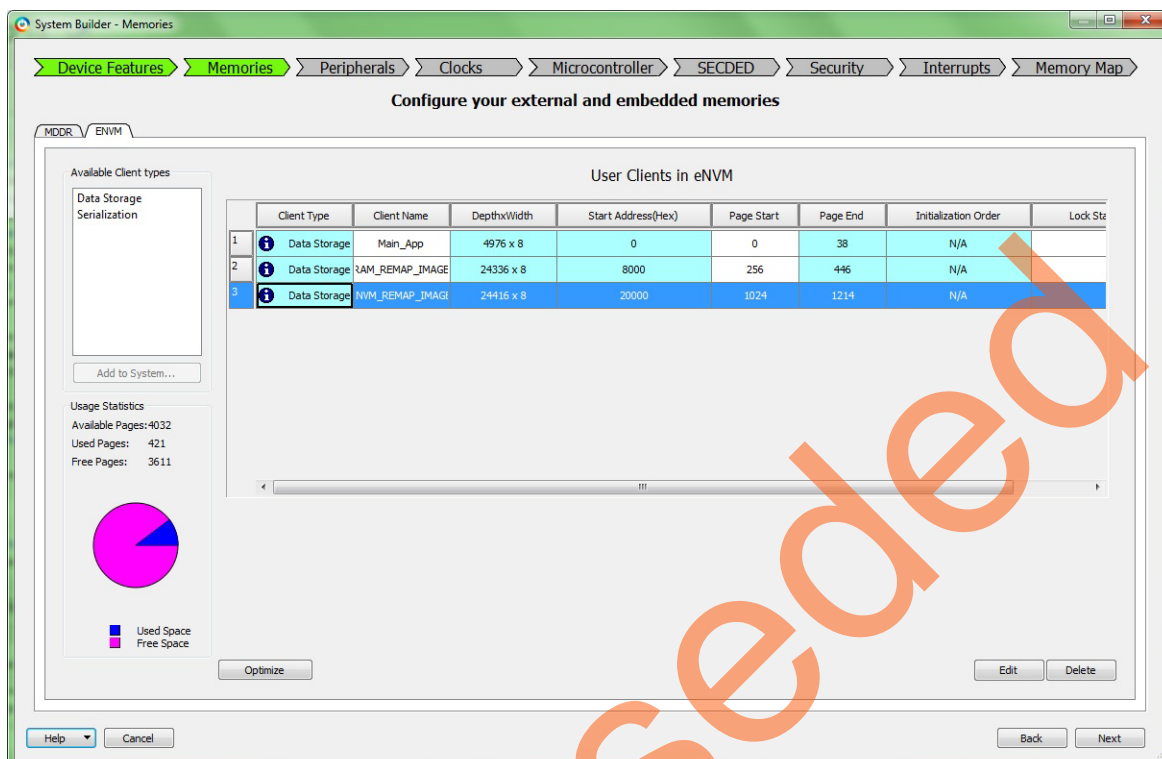
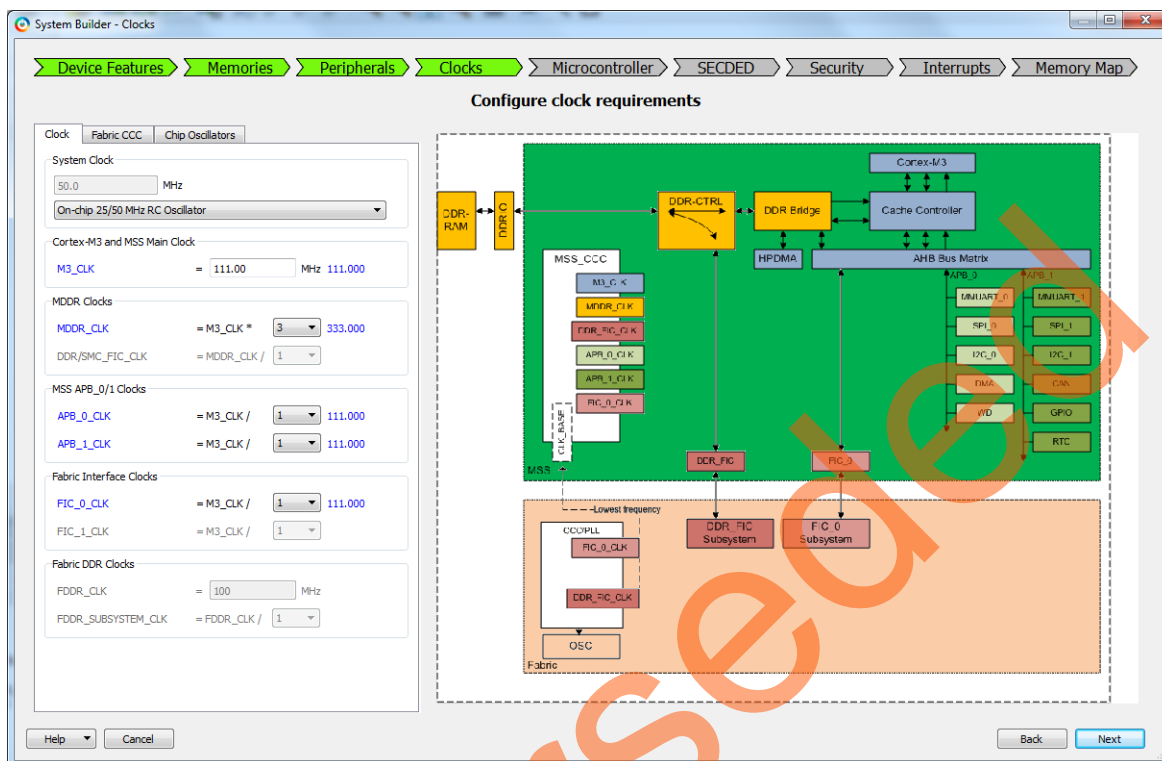


Figure 5 • Memory Device Configuration

The MMUART_0 is routed through FPGA fabric to communicate with the serial terminal program. The MSS_CCC clock is sourced from the FCCC via the CLK_BASE port. The FCCC is configured to provide the 100 MHz clock using GLO.

Figure 6 • Clock Configuration



Software Implementation

The following sections describe how to remap the various memory regions of the SmartFusion2 SoC FPGA to the Cortex-M3 processor code space. [Figure 7](#) describes the memory map for the Cortex-M3 processor.

	Memory Map of Cortex-M3 Processor	Memory Map of System Controller, FPGA Fabric Master, Ethernet MAC, Peripheral DMA	
	FPGA Fabric FIC Region5	FPGA Fabric FIC Region5	0xF0000000 - 0xFFFFFFFF
			0xE0000000 - 0xEFFFFFFF
	DDR_0 Space 3	DDR_0 Space 3	0xD0000000 - 0xDFFFFFFF
	DDR_0 Space 2	DDR_0 Space 2	0xC0000000 - 0xCFFFFFFF
	DDR_0 Space 1	DDR_0 Space 1	0xB0000000 - 0xBFFFFFFF
	DDR_0 Space 0	DDR_0 Space 0	0xA0000000 - 0xAFFFFFFF
	FPGA Fabric FIC Region4	FPGA Fabric FIC Region4	0x90000000 - 0x9FFFFFFF
	FPGA Fabric FIC Region3	FPGA Fabric FIC Region3	0x80000000 - 0x8FFFFFFF
	FPGA Fabric FIC Region2	FPGA Fabric FIC Region2	0x70000000 - 0x7FFFFFFF
			0x60100000 - 0x6FFFFFFF
	AHB-to-eNVM_1 Registers	AHB-to-eNVM_1 Registers	0x600C0000 - 0x600FFFFF
	AHB-to-eNVM_0 Registers	AHB-to-eNVM_0 Registers	0x60080000 - 0x600BFFFF
	eNVM_1	eNVM_1	0x60040000 - 0x6007FFFF
	eNVM_0	eNVM_0	0x60000000 - 0x6003FFFF
	FPGA Fabric FIC Region1	FPGA Fabric FIC Region1	0x50000000 - 0x5FFFFFFF
Peripheral Bit-band alias region of Cortex-M3 Processor	Peripherals(BB View)		0x44000000 - 0x4FFFFFFF
			0x42000000 - 0x43FFFFFF
	Cache Back door		0x40410000 - 0x41FFFFFF
			0x40400000 - 0x4040FFFF
	USB	USB	0x40044000 - 0x403FFFFF
			0x40043000 - 0x40043FFF
			0x40042000 - 0x40042FFF
	Ethernet MAC Control	Ethernet MAC Control	0x40041000 - 0x40041FFF
			0x40039000 - 0x40040FFF
	SYSREG	SYSREG	0x40038000 - 0x40038FFF
			0x40030000 - 0x40037FFF
	Config DDR_1, PCIe_0, PCIe_1 etc	Config DDR_1, PCIe_0, PCIe_1 etc	0x40020400 - 0x4002FFFF
	Config DDR_0	Config DDR_0	0x40020000 - 0x400203FF
			0x40018000 - 0x4001FFFF
	RTC	RTC	0x40017000 - 0x40017FFF
	COMBLK	COMBLK	0x40016000 - 0x40016FFF
	CAN	CAN	0x40015000 - 0x40015FFF
	High Performance DMA	High Performance DMA	0x40014000 - 0x40014FFF
	MSS GPIO	MSS GPIO	0x40013000 - 0x40013FFF
	I2C_1	I2C_1	0x40012000 - 0x40012FFF
	SPI_1	SPI_1	0x40011000 - 0x40011FFF
	UART_1	UART_1	0x40010000 - 0x40010FFF
			0x40007000 - 0x4000FFFF
	Fabric Interface Interrupt Controller	Fabric Interface Interrupt Controller	0x40006000 - 0x40006FFF
	Watchdog	Watchdog	0x40005000 - 0x40005FFF
	Timer	Timer	0x40004000 - 0x40004FFF
	Peripheral DMA Control	Peripheral DMA Control	0x40003000 - 0x40003FFF
	I2C_0	I2C_0	0x40002000 - 0x40002FFF
	SPI_0	SPI_0	0x40001000 - 0x40001FFF
	UART_0	UART_0	0x40000000 - 0x40000FFF
	FPGA Fabric FIC Region0	FPGA Fabric FIC Region0	0x30000000 - 0x3FFFFFFF
			0x24000000 - 0x2FFFFFFF
	eSRAM_0/eSRAM_1(BB View)		0x22000000 - 0x23FFFFFF
			0x20014000 - 0x21FFFFFF
	ECC eSRAM_1	ECC eSRAM_1	0x20012000 - 0x20013FFF
	ECC eSRAM_0	ECC eSRAM_0	0x20010000 - 0x20011FFF
	eSRAM_1	eSRAM_1	0x20008000 - 0x2000FFFF
	eSRAM_0	eSRAM_0	0x20000000 - 0x20007FFF
			0x00800000 - 0x1FFFFFFF
	eNVM (Cortex-M3) Virtual View	eNVM (Fabric) Virtual View	0x0007FFFF
			0x00000000

→ (63K space allocation for devices outside MSS)

Visible only to FPGA Fabric Master

Figure 7 • Cortex-M3 Processor Memory Map in SmartFusion2

Remapping eNVM Address Space to Cortex-M3 Processor Code Space

Figure 8 shows an example scenario with multiple executable images in the eNVM regions.

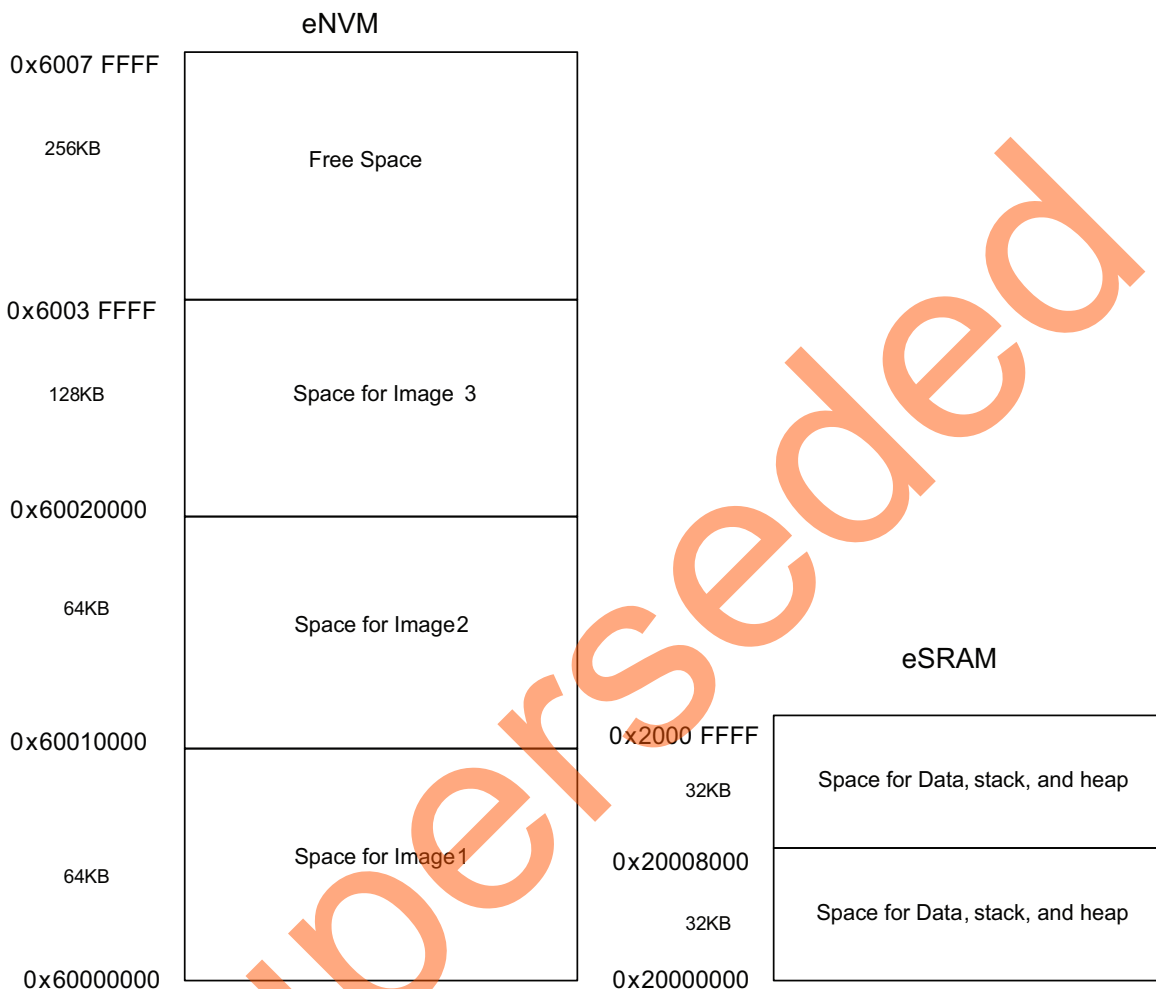


Figure 8 • Example Scenario of Multiple Executable Images in eNVM

In the example scenario (as shown in Figure 8), there are three images, which can be remapped to the starting address of the Cortex-M3 processor code space, or can be made executable for the Cortex-M3 processor. To create the independent executable images with the required memory map, it is required to create the linker scripts with the required memory map. The linker scripts are provided in the "Appendix A: Design Files" section. Once the executable images are created for the required memory map in Production mode, these images are added in the programming file using the eNVM clients in the Libero SoC hardware (HW) creation flow.

If the executable images are built with an absolute address, it is required to allow the execution control without using the remapping to the starting address of the code space (0x00000000). In such cases without remapping approach has to be used as explained below.

The execution control must be allowed to the desired image by using the following two approaches:

- **Without remapping:** By default, the eNVM base address 0x60000000 is remapped to the starting address of the code space of the Cortex-M3 processor. The vector table address of the desired image can be set by using the vector table offset register in the system registers, and pointing the stack pointer (SP) and program counter to the reset handler address of the desired image. This allows the Cortex-M3 processor to execute the new image. The eNVM offset address must be used in the linker script generation for the executable images in this approach. This approach is explained in the flow chart shown in [Figure 9](#).

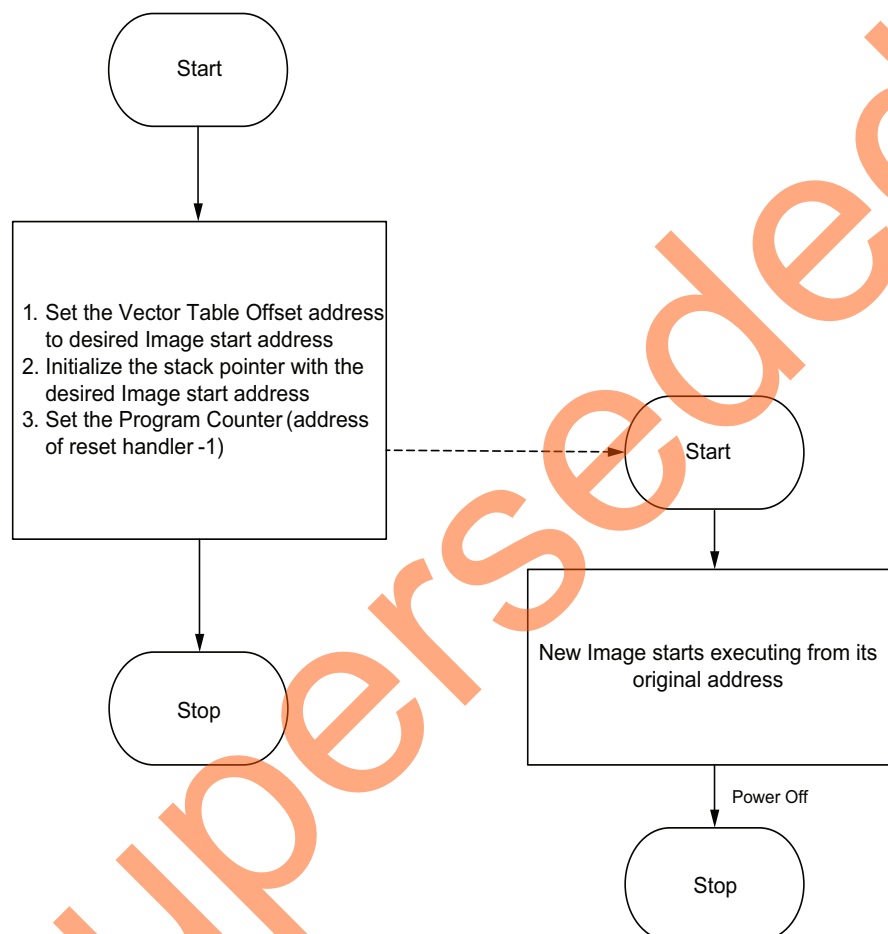


Figure 9 • Logic for Moving Execution Control to New Image in eNVM without Remapping

For example, for the memory map of the different images explained in the [Figure 8](#), the images are built with the base address as shown in [Figure 8](#). To run Image 2 while Image 1 is running, use the following steps, as shown in [Figure 9](#):

1. Set the vector table offset address register is set to 0x60010000
2. Initialize the stack pointer with the content of 0x60010000
3. Change the program counter to the reset handler of Image 2 that is, PC = (0x60010004 -1)

With the all above 3 steps Image 2 starts executing from 0x60010000.

- **With remapping:** In this approach, the new image address can be remapped to the starting address of the code region of the Cortex-M3 processor by using the ENV_M_CR, ENV_M_REMAPSIZE, and ENV_M_REMAP_BASE_CR registers. As the new image address is remapped to the bottom (0x0000_0000) of the Cortex-M3 processor code region, the linker scripts take care of building the images from the bottom (0x0000_0000) code region. The eNVM offset address must not be used in this approach. This approach is explained in the flow chart shown [Figure 10](#).

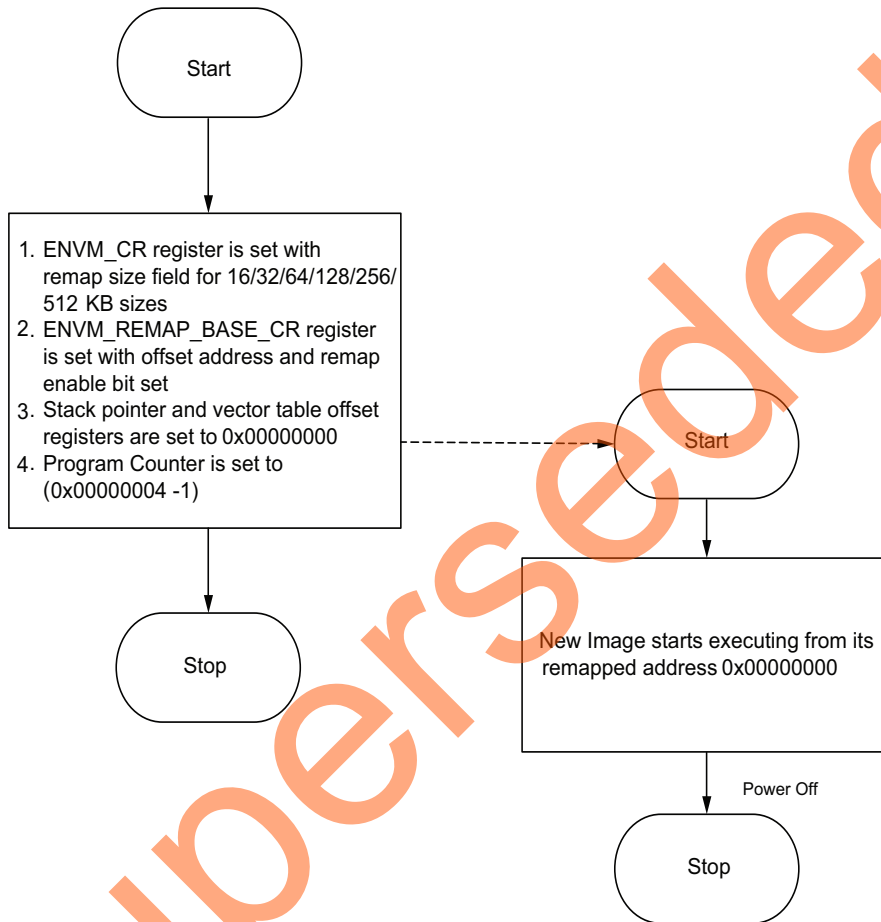


Figure 10 • Logic for Moving Execution Control to New Image in eNVM with Remapping

For example, for the memory map of the different images explained in the [Figure 8](#), the images are built with 0x00000000 as base address. To run the Image 2 while Image 1 is running, use the following steps (explained in [Figure 10](#)):

1. Set the ENV_M_CR register to 64KB as remap image size
2. Set the ENV_M_REMAP_BASE_CR register with 0x00010000
3. Set the Stack Pointer to 0x00000000
4. Set the PC to 0x00000004 - 1

Note: To set the ENV_M_CR register to 16/32/64/128/256/512 KB as remap image size, refer to [ENV_M controller section in SmartFusion2_CortexM3_UG.pdf](#).

With all the above steps the new Image 2 starts executing from 0x00000000 which is mapped to 0x60010000.

Reference design is provided with this application note with remapping and without remapping. Refer to "[Appendix A: Design Files](#)" section for design files and follow the steps explained in "[Setting up the Demo Design](#)" section for executing the reference design.

Table 2 describes the registers which are required to be set for the eNVM remapping to the bottom (0x0000_0000) of the Cortex-M3 processor. The SYSREG block is located at address 0x40038000 in the Cortex-M3 processor address space.

Table 2 • eNVM Remap Registers

Register Name	Address Offset	Register Type	Flash Write Protect	Reset Source	Description
ENVN_CR	0xC	RW-P	Register	sysreset_n	eNVM Configuration register
ENVN_REMAP_BASE_CR	0x10	RW-P	Register	sysreset_n	eNVM remap configuration register for the Cortex-M3 processor.

Remapping eNVM to Soft Core Processor Memory Map

Soft core processor implemented in SmartFusion2 SoC FPGA fabric can access the eNVM for the code execution purposes. For this use case the fabric interface controller (FIC_0 or FIC_1) and the eNVM AHB controller need to be set properly. The eNVM partitioning between the Cortex M3 and SoftCore processor needs to be taken care in such a way that these two partitions are mutually exclusive. The remapping of the eNVM offset address to the soft core processor bottom (0x0000_0000) address map is very similar to the remapping of the eNVM address to the Cortex-M3 processor. ENVN_REMAP_FAB_CR register has to be used instead of ENVN_REMAP_BASE_CR register. The SYSREG block is located at address 0x40038000 in the Cortex-M3 processor address space.

Table 3 describes the eNVM remap register to fabric SoftCore processor address space.

Table 3 • eNVM Remap Register to Fabric SoftCore Processor Address Space

Register Name	Address Offset	Register Type	Flash Write Protect	Reset Source	Description
ENVN_REMAP_FAB_CR	0x14	RW-P	Register	sysreset_n	NVM remap configuration register for the soft processor in the FPGA

Remapping eSRAM to Cortex-M3 Processor Code Space

Figure 11 shows the example scenario of the executable images in eSRAM regions.

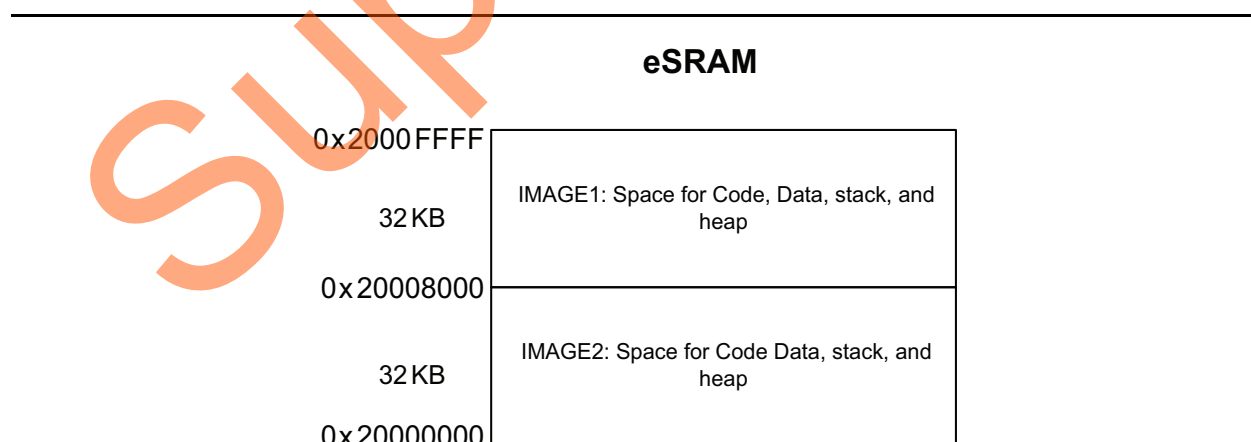


Figure 11 • Example Scenario of Multiple Executable Images in eSRAM

The above scenario, shown in [Figure 11](#), describes two images which can be remapped to the bottom (0x0000_0000) of the Cortex-M3 processor or can be made executable for the Cortex-M3 processor. To create the independent executable images with required memory map, the linker scripts need to be created with required memory map. The linker scripts are provided in the ["Appendix A: Design Files" section](#). Once the images are created for the required memory map in Production mode, these images are to be copied to an external memory like SPI Flash/eNVM, and are code shadowed by the bootloader to the eSRAM whenever it is required to execute the new images.

Once the images are copied to eSRAM by bootloader, the execution control can be allowed to the desired image by using any of the following two approaches:

If the executable images are built with an absolute address, the execution control needs to be allowed without using the remapping to starting address of the code space (0x00000000). In such cases, without remapping approach explained below (Point 1) has to be used.

If the executable images are built with the address 0x00000000, the execution control needs to be allowed by using remapping to starting address of the code space (0x00000000). In such cases, remapping approach explained below (Point 2) has to be used.

1. **Without remapping:** Using the vector table offset register in the system registers, the vector table address of the desired image can be set for execution, and point the stack pointer (SP) and the program counter to the reset handler of the desired image. This allows the Cortex-M3 processor to execute the new image. The eSRAM address must be used in the linker script generation for the executable images in this approach. This approach is explained in the flow chart shown in [Figure 12](#).

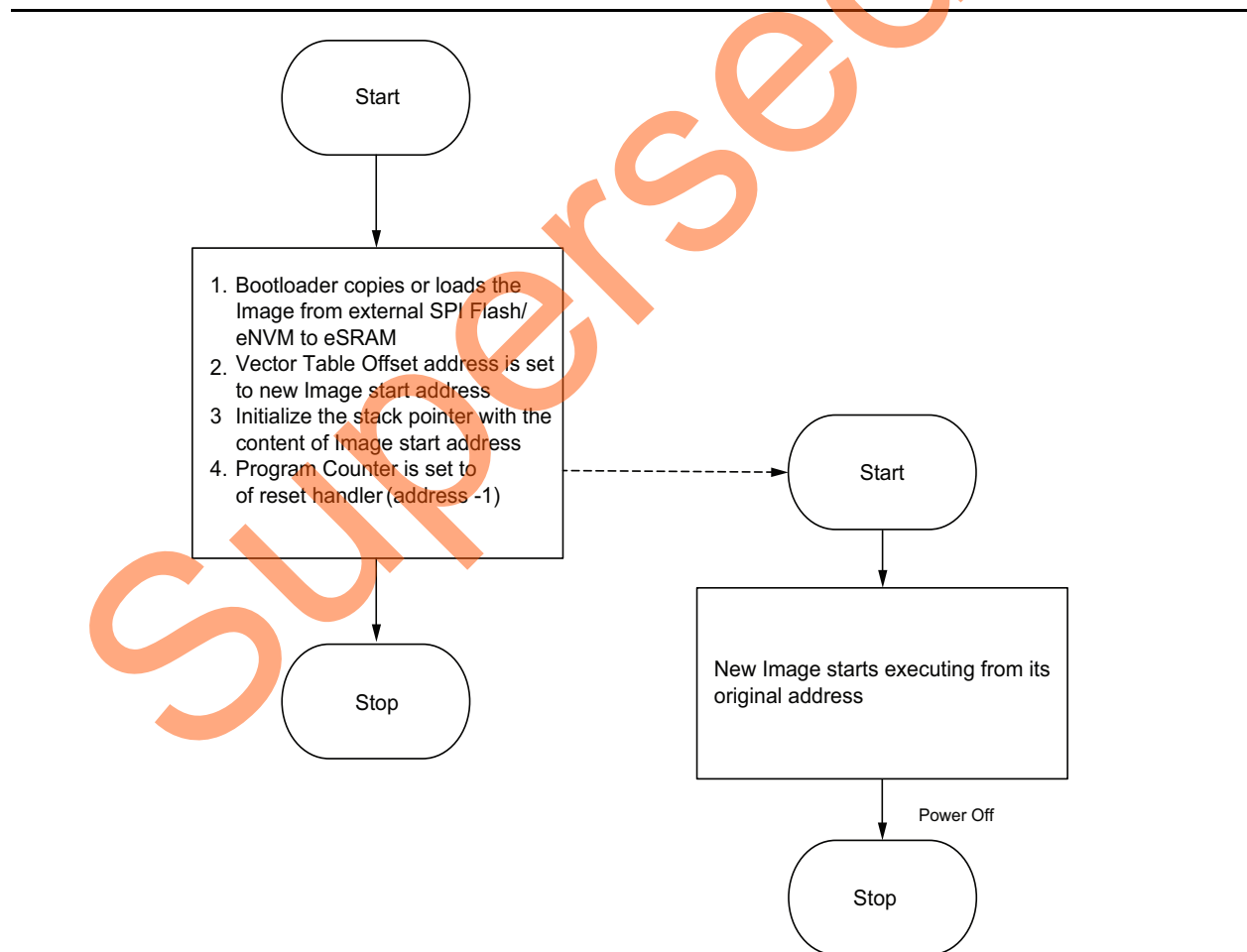


Figure 12 • Logic for Moving Execution Control to New Image in eSRAM without Remapping

- **With Remapping:** In this approach, the new image address can be remapped to the bottom (0x0000_0000) of the Cortex-M3 processor by using the ESRAM_CR registers. As the new image address is remapped to bottom (0x0000_0000) of the Cortex-M3 processor code region the linker scripts take care of building the images from the bottom (0x0000_0000) code region. The eSRAM address must not be used instead offset address from zero has to be used in the linker scripts for this approach. This approach is explained in the flow chart shown in [Figure 13](#).

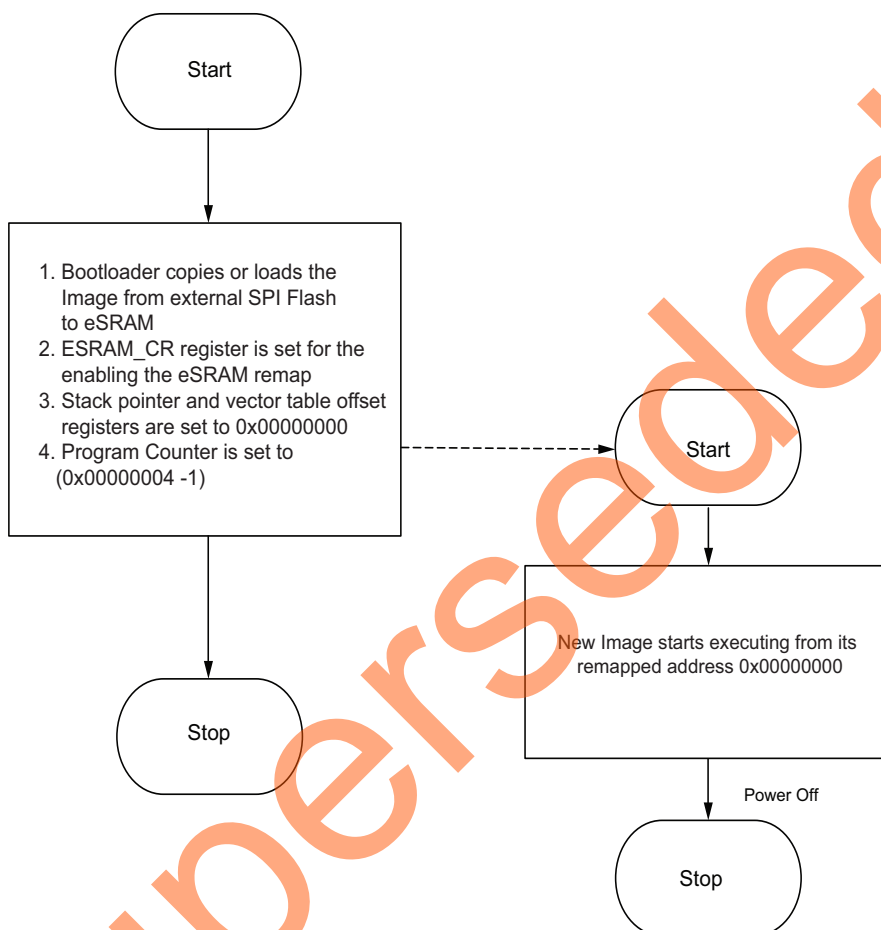


Figure 13 • Logic for Moving Execution Control to New Image in eSRAM with Remapping

For example, for the memory map of the different images explained in the [Figure 11](#), the images are built with 0x00000000 as base address. If it is required to jump from the Image 2 to Image 1 then use the following steps (as explained in the [Figure 13](#)).

1. Copy the image1 from Flash to eSRAM starting address 0x20008000
2. Set the ESRAM_CR register to enable the eSRAM remapping to 0x00000000
3. Set the Stack Pointer to 0x00008000 and Vector Table offset register to 0x00008000
4. Set the PC to 0x00008004 -1

With all the above steps the new Image1 starts executing from 0x00008000 which is the mapped to address 0x20008000.

Reference design is provided with this application note with remapping and without remapping. Refer to the ["Appendix A: Design Files"](#) section for the design files and follow the steps explained in ["Setting up the Demo Design"](#) section for executing the reference design.

Table 4 explains the registers that are required to be set for the eSRAM remapping. The SYSREG block is located at address 0x40038000 in the Cortex-M3 processor address space.

Table 4 • Registers Required to eSRAM Remapping

Register Name	Address Offset	Register Type	Flash Write Protect	Reset Source	Description
ESRAM_CR	0x0	RW-P	Register	sysreset_n	Controls address mapping of the eSRAMs

Remapping External RAM (DDR/SDR SDRAM Interface) to Cortex-M3 Processor Code Space

Figure 14 shows the scenario of the multiple executable images in DDR/SDRAM interface memory regions.

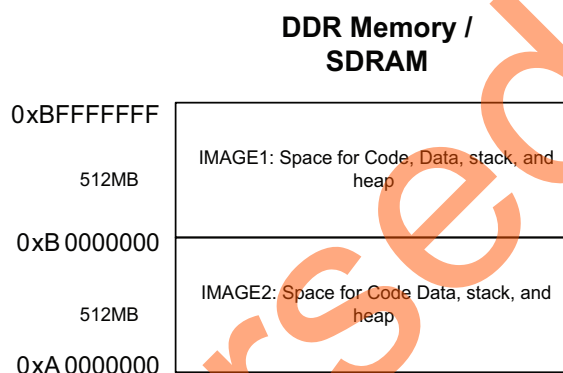


Figure 14 • Example Scenario of Multiple Executable Images in DDR/SDR SDRAM

In this scenario there are two images which can be remapped to the bottom (0x0000_0000) of the Cortex-M3 processor, or can be made executable for the Cortex-M3 processor.

To create the independent executable images with required memory map, the linker scripts need to be created with the required memory map. The linker scripts are provided in ["Appendix A: Design Files" section](#). Once the images are created for the required memory map in Production mode, these images are to be copied to an external memory like SPI Flash, and code shadowed by the bootloader to DDR memory or SDRAM whenever the execution of the new images is required.

Once the image is copied to the DDR memory and SDRAM by the bootloader, the execution control can be allowed to the desired image by using the following approach.

The new image address can be remapped by using the DDR_CR register to the bottom (0x0000_0000) of the Cortex-M3 processor code region. As the new image start address is re-mapped to the Cortex-M3 processor code region 0x0000_0000, the linker scripts take care of building the images from the code region 0x0000_0000. The DDR memory or SDRAM addresses (0xA000_0000) must not be used. Instead, the offset address from zero has to be used in the linker scripts for this approach.

As the DDR memory or SDRAM memory address range cannot be used in the Vector table offset register, so it is required to remap these memories to start address of the Cortex-M3 processor code space for the execution from these memories. This approach is explained in the flow chart shown in [Figure 15](#).

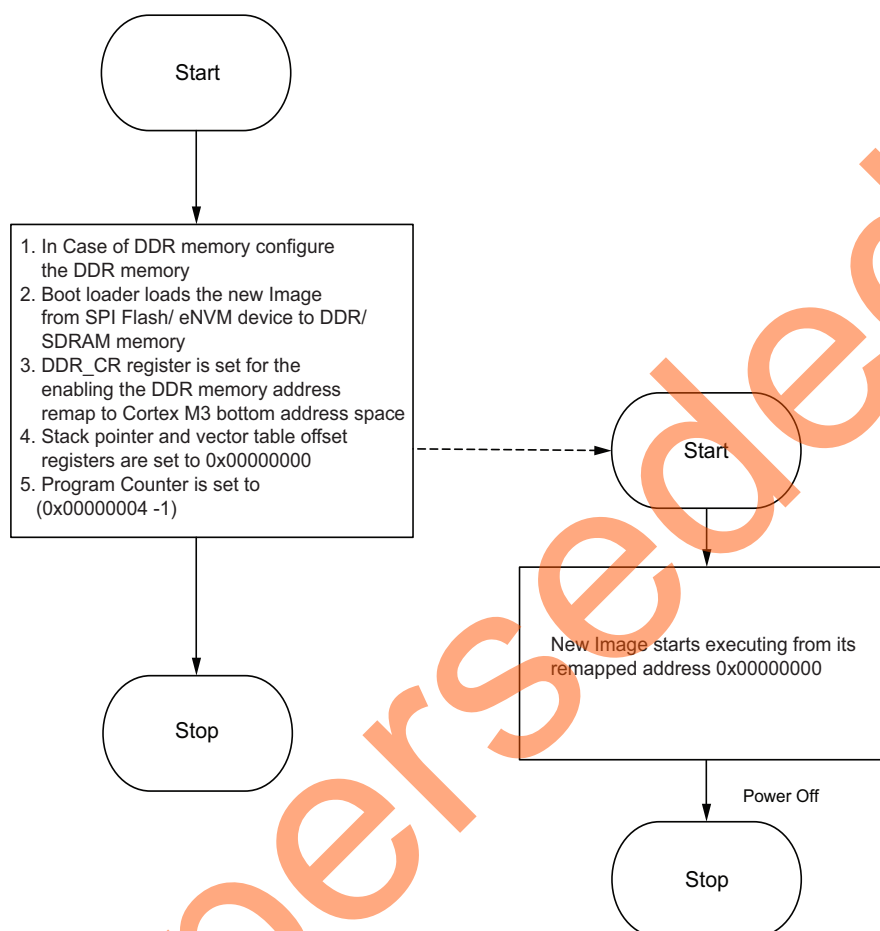


Figure 15 • Logic for Moving the Execution Control to New Image in DDR/SDR SDRAM with Remapping

Reference design is provided with this application note with remapping. Refer to the "[Appendix A: Design Files](#)" section for the design files and follow the steps explained in "[Setting up the Demo Design](#)" section for executing the reference design.

[Table 5](#) shows the registers required to be set for the DDR/SDR SDRAM remapping. The SYSREG block is located at address 0x40038000 in the Cortex-M3 processor address space.

Table 5 • Registers Required to DDR/SDR SDRAM Remapping

Register Name	Address Offset	Register Type	Flash Write Protect	Reset Source	Description
DDR_CR	0x8	RW-P	Register	sysreset_n	DDR control Register. Configures DDR Space.

Firmware Drivers

The following firmware drivers are used in this application:

- MSS MMUART driver
 - To communicate with serial terminal program on the Host PC
- MSS GPIO driver
 - To drive onboard LEDs.

Setting up the Demo Design

The following steps describe how to setup the demo for SmartFusion2 Advanced Development Kit Board:

1. Connect the Host PC to the J33 Connector using the USB A to mini-B cable. The USB to UART bridge drivers are automatically detected. Download and install the drivers from www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip if the drivers are not installed or detected automatically. Verify if the detection is made in the device manager as shown in Figure 16.

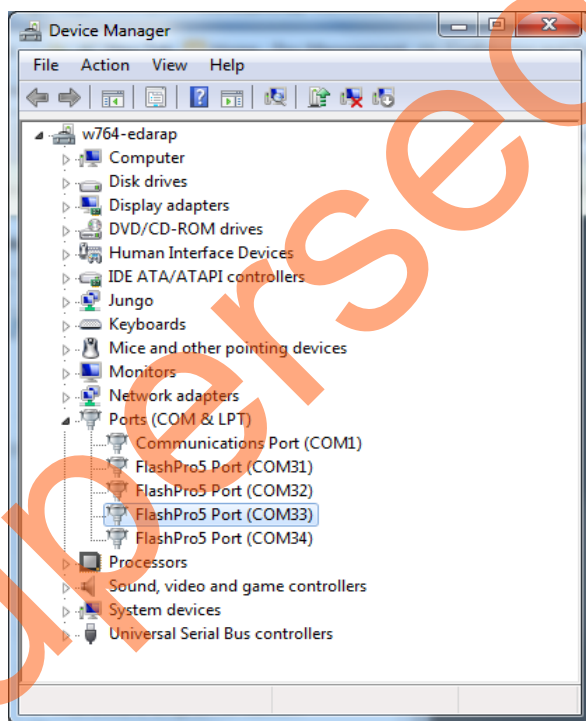


Figure 16 • Device Manager

2. Select one of the four COM ports with **Location** as on **USB FP5 Serial Converter C**. [Figure 17](#) shows the **Device Manager** window and its properties that display the **USB Serial Port** details. The COM port number is required to run the demo design.

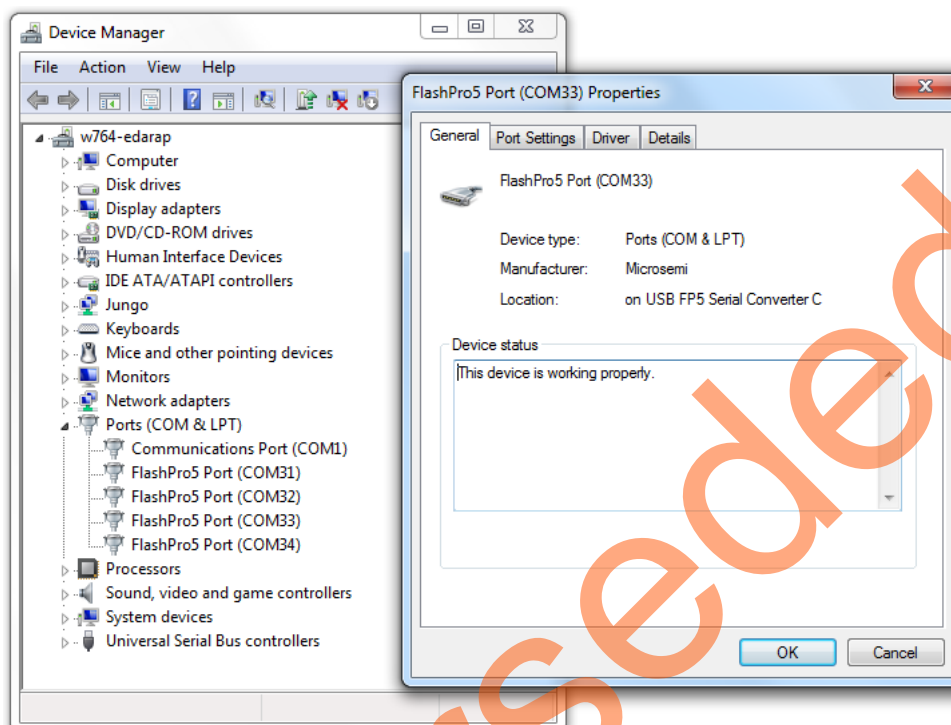


Figure 17 • Device Manager - FlashPro5 Properties

3. Connect the jumpers on the SmartFusion2 Advanced Dev Kit Board as shown in [Table 6](#).
CAUTION: The power supply switch **SW7** on the board should be in OFF position, while making the jumper connections.

Table 6 • SmartFusion2 FPGA Advanced Development Kit Jumper Settings

Jumper	Pin (from)	Pin (to)	Comments
J116, J353, J354, J54	1	2	These are the default jumper settings of the Advanced Dev Kit Board. Ensure these jumpers are set accordingly.
J123	2	3	
J124, J121, J32	1	2	JTAG programming via FTDI

4. Connect the power supply to the J42 Connector on the SmartFusion2 Advanced Development Kit Board.

Board Setup

Snapshots of the SmartFusion2 Advanced Development Kit Board with the complete set up is given in the ["Appendix B: SmartFusion2 Advanced Development Kit Board"](#) on page 25.

Programming SmartFusion2 Advanced Development Kit Board

The following steps describe how to program SmartFusion2 Advanced Development Kit Board.

1. Download the demo design from:
http://soc.microsemi.com/download/rsc/?f=m2s_ac390_liberov11p5_df
2. Switch **ON** the **SW7** power supply switch.
3. Launch the **FlashPro** software.
4. Click **New Project**.
5. In the **New Project** window, type the **Project Name** as Remapping_stp_Program.
6. Click **Browse** and navigate to the location where you want to save the project.
7. Click **Single device** as the **Programming mode**.

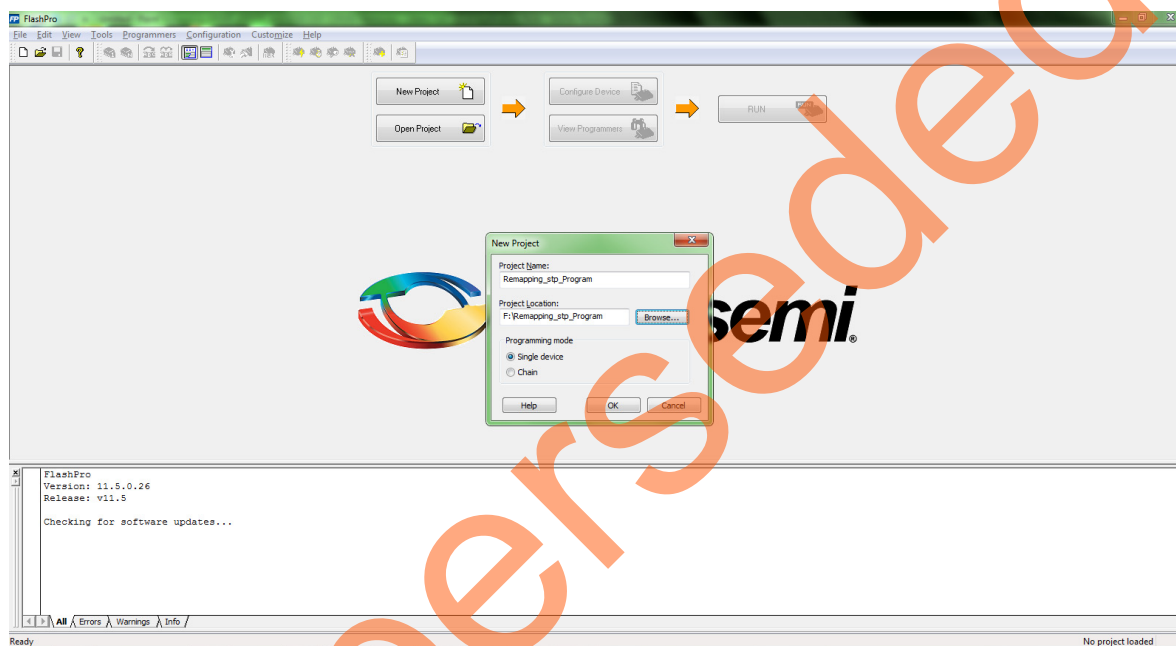


Figure 18 • FlashPro New Project

8. Click **OK** to save the project.

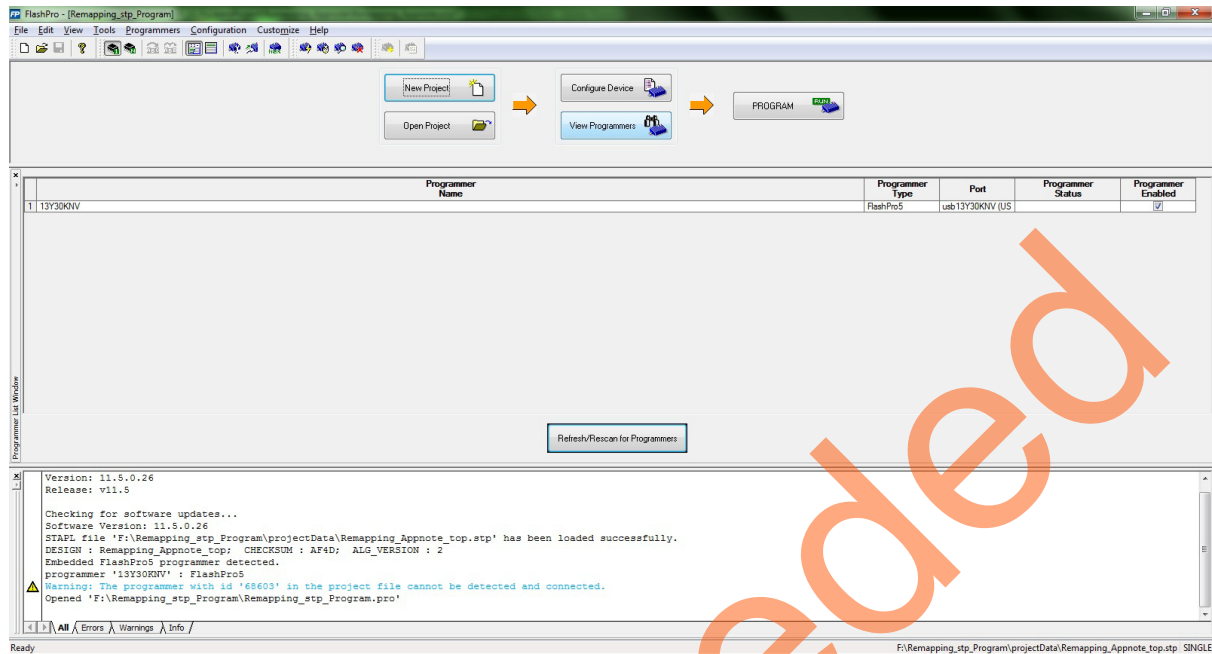


Figure 19 • FlashPro5 Programmer Type

9. Click **Configure Device** on the FlashPro GUI.
10. Click **Browse** and navigate to the location where the `Remapping_Appnote_top.stp` file is located and select the file. The location for SmartFusion2 Advanced Development Kit Board is:
`<download_folder>\m2s_ac390_liberov11p5_df\programming_file.`

11. Click **Open**. The required programming file is selected and is ready to be programmed in the device.

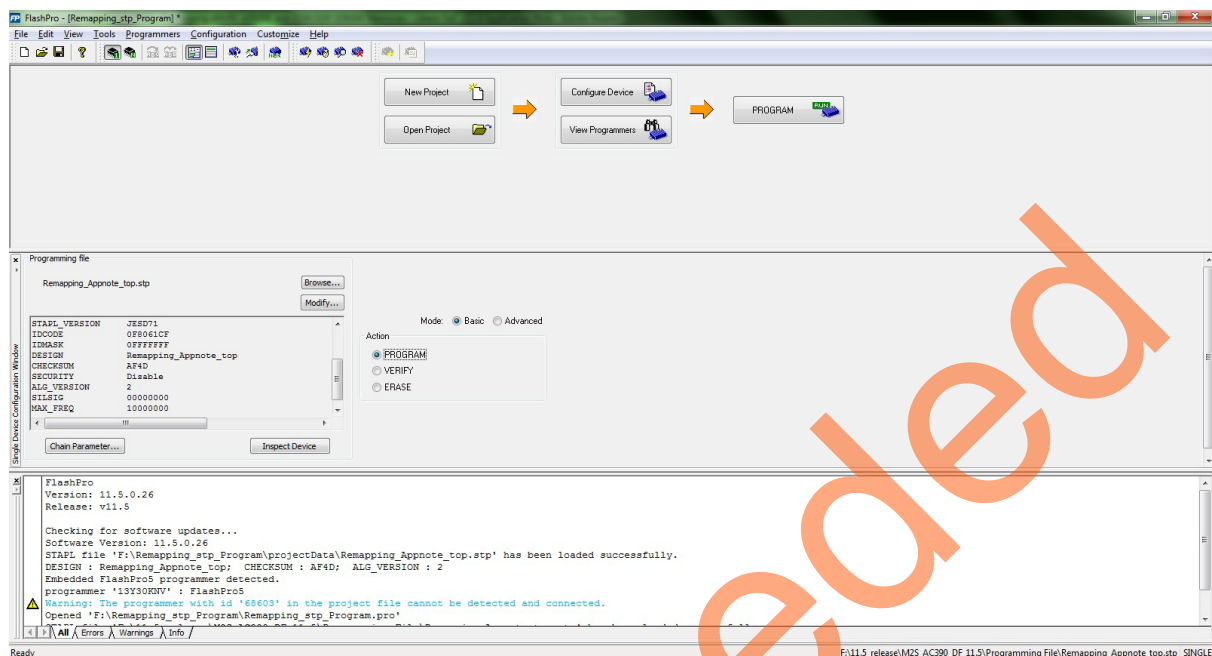


Figure 20 • FlashPro Project Configured

12. Click **PROGRAM** to start programming the device. Wait until a message is displayed indicating that the **PROGRAM PASSED**.

Running the Design

The following steps describe how to run the design.

1. Press **SW9** switch to reset the board after successful programming.

Figure 21 shows the Tera Term.

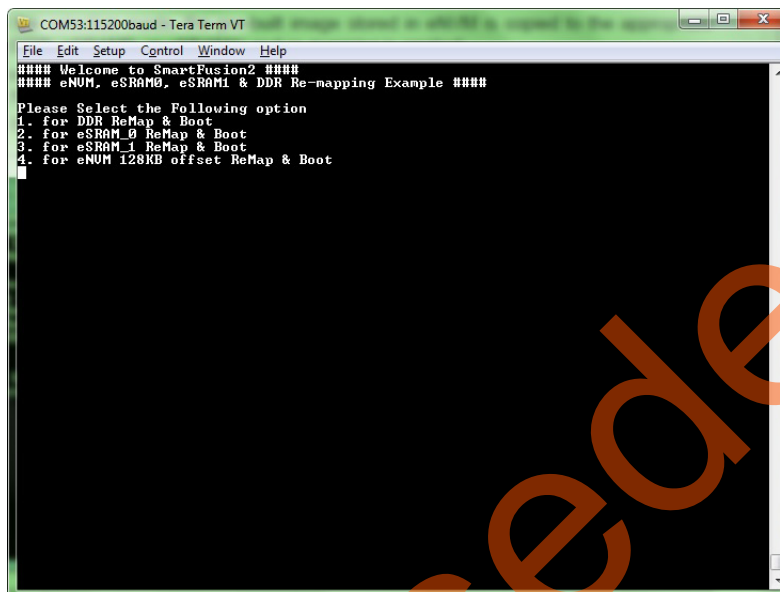


Figure 21 • Main Menu of Re-Mapping Application Note

2. Based on the selection made, the pre-built image stored in eNVM is copied to the appropriate locations (DDR, eSRAM0, or eSRAM1) and re-mapping is applied.
3. Once the re-mapping is completed, the new Image starts booting and the following messages are shown on the serial terminal and LED starts blinking on the SF2 Advanced Development Kit.

Note: Reset the SmartFusion2 Advanced Development Kit board to switch among the application images.

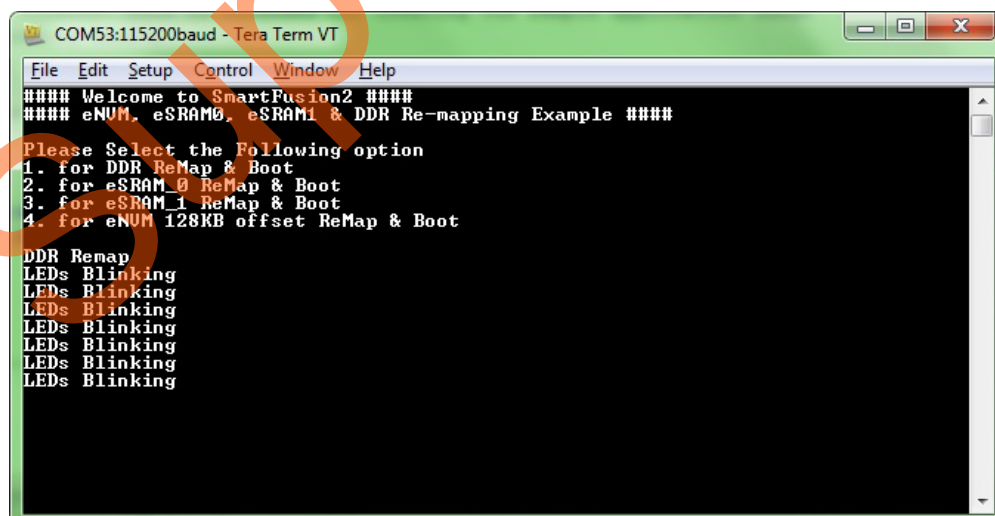


Figure 22 • Re-mapped Image is Running

For booting multiple images without remapping, refer to the [SmartFusion SoC: Basic Bootloader and Field Upgrade eNVM Through IAP Interface](#) application note.

Conclusion

This application note explains the remapping of the eNVM, eSRAM, and DDR/SDR SDRAM memories to the Cortex-M3 processor code region. It also explains how to execute the program code which is built with absolute addresses without remapping in case of eNVM and eSRAM.

Superseded

Appendix A: Design Files

The design files (DF), programming files (PF), and linker scripts (LD) can be downloaded from the Microsemi® SoC Products Group website:

http://soc.microsemi.com/download/rsc/?f=m2s_ac390_liberov11p5_df

http://soc.microsemi.com/download/rsc/?f=m2s_ac390_liberov11p5_pf

http://soc.microsemi.com/download/rsc/?f=m2s_ac390_liberov11p5_ld

The design file consists of Libero Verilog, SoftConsole software project, programming files (*.stp) for the SmartFusion2 SoC FPGA Advanced Development Kit. Refer to the Readme.txt file that is included in the design file for the directory structure and description.

Appendix B: SmartFusion2 Advanced Development Kit Board

Figure 23 shows the SmartFusion2 Advanced Development Kit Board.



Figure 23 • SmartFusion2 Advanced Development Kit Board

List of Changes

The following table lists critical changes that were made in the user guide.

Date	Changes	Page
Revision 7 (February 2015)	Updated the document for Libero SoC v11.5 software release (SAR 64108).	NA
Revision 6 (September 2014)	Updated the document for Libero SoC v11.4 software release (SAR 60315).	NA
	Updated the document for Advanced Development Kit Board details (SAR 60315).	NA
Revision 5 (May 2014)	Figure 2 is changed (SAR 57912).	4
	Added Figure 3 (SAR 57912).	5
	Added Figure 4 (SAR 57912).	6
	Added Figure 5 (SAR 57912).	7
	Updated the document for Libero SoC v11.3 software release (SAR 57912).	NA
Revision 4 (January 2014)	Figure 6 is changed.	8
	Figure 3 is changed.	5
Revision 3 (May 2013)	Updated the document for Libero SoC v11.0 software release (SAR 47617).	NA
Revision 2 (March 2013)	Updated the document for Libero SoC v11.0 beta SP1 software release (SAR 45398)	NA
Revision 1 (November 2012)	Updated "Remapping eNVM Address Space to Cortex-M3 Processor Code Space" section. (SAR 42911).	10
	Updated "Remapping eSRAM to Cortex-M3 Processor Code Space" section (SAR 42911).	13
	Updated "Remapping External RAM (DDR/SDR SDRAM Interface) to Cortex-M3 Processor Code Space" section (SAR 42911).	16
	Updated "Setting up the Demo Design" section (SAR 42911).	18
	Updated "Appendix A: Design Files" section (SAR 42911).	25



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

© 2015 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.

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; 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.

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.