

# SmartFusion cSoC: Programming FPGA Fabric and eNVM Using In-Application Programming Interface

## Table of Contents

Introduction . . . . .	1
IAP and DirectC Overview . . . . .	2
IAP With Overall System Design . . . . .	4
Known Issues . . . . .	5
Design Overview . . . . .	6
Design Example . . . . .	6
Solution 1: Programming FPGA Fabric/eNVM Directly from the Host PC Using UART . . . . .	7
Solution 2: Programming FPGA Fabric Directly From the Host PC Using Ethernet . . . . .	12
Solution 3: Programming FPGA Fabric/eNVM From the On-Board SPI Flash . . . . .	17
Conclusion . . . . .	27
Appendix A – Design Files . . . . .	28
List of Changes . . . . .	29

## Introduction

The SmartFusion® customizable system-on-chip (cSoC) device contains a hard embedded microcontroller subsystem (MSS), programmable analog circuitry, and FPGA fabric consisting of logic tiles, static random access memory (SRAM), and phase-locked loops (PLLs). The MSS consists of a 100 MHz ARM® Cortex™-M3 processor, advanced high-performance bus (AHB) matrix, system registers, Ethernet MAC, peripheral DMA (PDMA), real-time counter (RTC), embedded nonvolatile memory (eNVM), embedded SRAM (eSRAM), fabric interface controller (FIC), the Philips Inter-Integrated Circuit (I²C), serial peripheral interface (SPI), external memory controller (EMC), embedded flashROM (eFROM), Watchdog Timer, 10/100 Ethernet Controller, GPIO block, in-application programming (IAP) and system registers. The programmable analog block contains the analog compute engine (ACE) and the analog front-end (AFE), consisting of ADCs, DACs, active bipolar prescalers (ABPS), comparators, current monitors, and temperature monitor circuitry.

The IAP block is a feature of the MSS that interfaces with the Cortex-M3 processor through the APB bus. The IAP block provides hardware capability to program the flash components of SmartFusion cSoC devices, by invoking the IAP block in the programmed end user's application. These flash components are the FPGA array, the MSS eNVM, and the FlashROM. The hard 100 MHz 32-bit ARM Cortex-M3 processor in the SmartFusion cSoC device is used to execute the DirectC code in conjunction with the IAP block to update the FPGA fabric and eNVM.

The processor utilizes the DirectC application to execute the programming algorithms (erase, program, and verify), that controls the JTAG state machine in the IAP block. The programming operation is done one row at a time. When programming for a row is done, IAP sends an interrupt to the processor for the next row until the flash component is completely programmed. The processor, upon interrupt, initiates the next instruction/command based on the programming algorithm. It is the responsibility of the processor application to clear the interrupt source upon servicing an interrupt request. This is done by writing to any register in the IAP block.

The IAP programming can be initiated by the system, within the Cortex-M3 processor application, or from the FPGA fabric/eNVM through an interrupt to the Cortex-M3 processor as defined at design time; the flexibility is dependent on the user application.

Figure 1 shows the various blocks of the SmartFusion cSoC architecture that are used for performing IAP to program the FPGA fabric/eNVM.

When IAP is initiated, a host application on the host PC transfers the programming file (\*.dat) from the host PC to the target (SmartFusion cSoC-based system) through UART/Ethernet for performing IAP to program the FPGA fabric/eNVM.

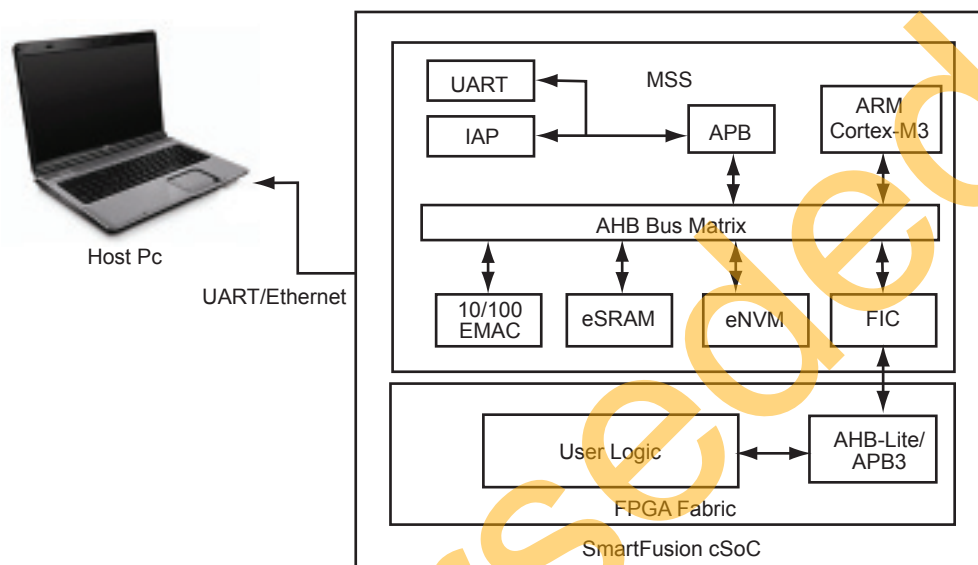


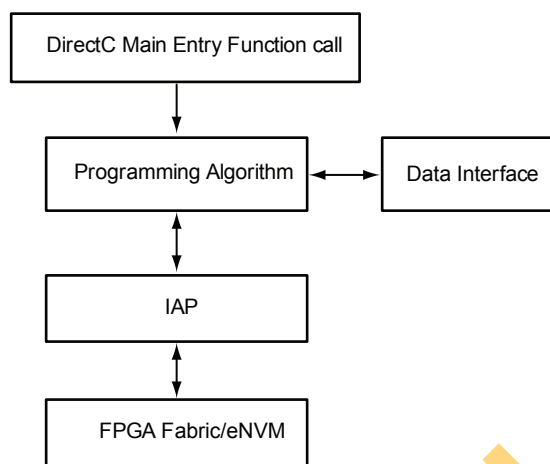
Figure 1 • Typical SmartFusion cSoC Configuration for IAP

## IAP and DirectC Overview

DirectC is a set of source code files, designed to support IAP for updating FPGA fabric/eNVM. The DirectC code files are grouped together to form the following blocks:

- DirectC main entry function call
- Programming algorithm
- Data interface
- IAP

Figure 2 illustrates the flow of DirectC and IAP usage in programming flash components.



**Figure 2 • Flow of DirectC and IAP Usage**

## DirectC Main Entry Function Call

The DirectC source code starts with the main entry function call. This block performs the following steps:

1. Initializes all the needed variables for programming.
2. Checks data CRC to ensure there is no corruption.
3. Erases the target flash device/component before programming.
4. Makes a call to the programming algorithm block to perform IAP.

## Programming Algorithm

The programming algorithm block does the following:

- Initializes the flash component with the help of IAP block.
- Calls the appropriate function to perform the required action.
- Interfaces with the data interface (UART, Ethernet etc) to read the programming file.

## Data Interface

The data interface provides the programming data \*.dat file when requested by the IAP block through the programming algorithm to perform the requested action. This programming data is provided either directly from host PC or from the on-board storage that was loaded earlier.

## IAP

The IAP block is part of a digital subsystem which acts as an APB peripheral for the Cortex-M3 processor. This block is a combination of both hardware and software. It provides a low-level interface to program the flash component through the control and status registry.

This block primarily does the following:

- IAP software processes the data and applies the appropriate algorithm to program the device.
- IAP hardware block handles the JTAG communication to program the controller block to program the flash component.

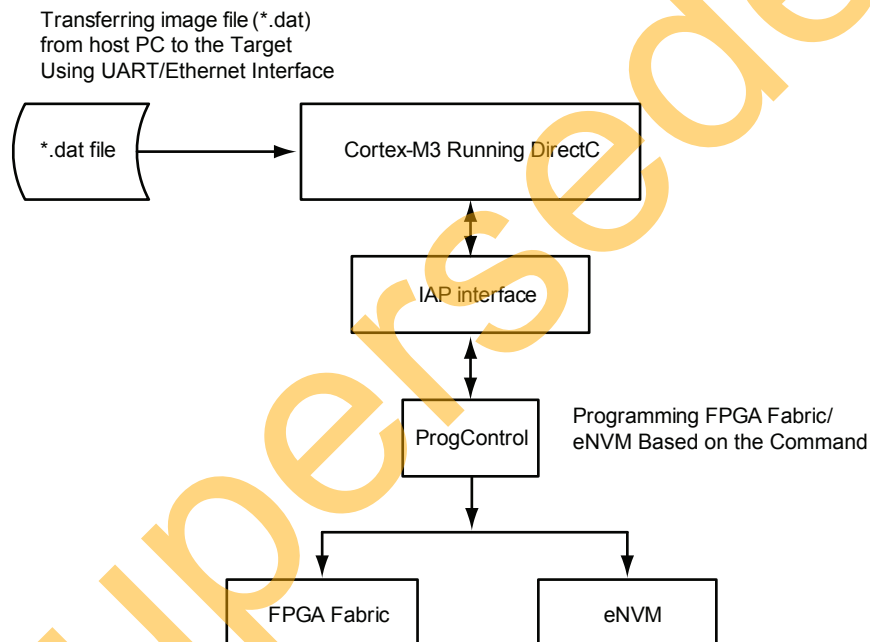
## IAP With Overall System Design

Figure 3 shows the high-level architecture of IAP with overall system design for programming the flash component with an input programming file.

The DirectC code that is running on the Cortex-M3 processor uses the IAP interface in order to perform IAP to update the FPGA fabric/eNVM with the input programming file. The input programming file (\*.dat) is transferred from the host PC to the target through UART/Ethernet.

The following parts are involved while performing IAP:

1. An interface application software running on the Cortex-M3 processor with the DirectC code initiates IAP and handles external communications.
2. A host application software provides the programming file data when the IAP block requests data to program the FPGA fabric/eNVM.
3. The DirectC processes the data and applies the appropriate algorithm to initialize the FPGA fabric/eNVM for programming.
4. The IAP hardware block handles the JTAG communication to program the controller (ProgCtrl) hardware block, for programming FPA fabric/eNVM with the incoming programming file.



**Figure 3 • High-Level Architecture of IAP**

The following are needed for the designer to implement and perform IAP:

- Integrate the IAP driver and the DirectC files into the application code. You can download the IAP driver using the Firmware Catalog from the Microsemi website or provided within the hardware project firmware directory.
- Ensure definitions for `ENABLE_IAP_SUPPORT` and `enable_mss_support` (in `dpuser.h`) are enabled in the DirectC code. The DirectC code enables the IAP support by setting the `hardware_interface` variable to `IAP_SEL` to perform IAP. These variables are enabled by default in the code.
- Implement the data interface block to provide the programming file from the host PC to the target when IAP requests for data. There are several physical communication medium interfaces (UART/Ethernet) available on SmartFusion cSoCs to get the programming file \*.dat on to the target.

The data interface block mainly reads the programming file \*.dat data from the host through UART/Ethernet or from on-board SPI flash storage. In the case of on-board SPI flash storage, the

complete programming file \*.dat is loaded in SPI flash prior to performing the IAP block. When UART is used as a physical medium, a handshaking mechanism is required between the host and the target to load the programming file directly from the host into internal memory (eSRAM) or to a memory storage area on the target board. Typically this handshake mechanism has to be implemented before IAP actually requests programming data to program the flash component. When Ethernet is used as a physical medium, it uses Ethernet routines and a TCP/IP stack to load the programming file directly from the host PC over the network into internal memory (eSRAM) or to a storage file system on the target application.

The design example provided in this application note demonstrates the programming of IAP blocks to update the FPGA fabric or eNVM with the programming file that is loaded onto the target using various communication medium interfaces.

All the solutions that are explained in this application note perform IAP programming, but the difference between them is the underlying physical interface medium which is used to load the programming file \*.dat from the host to the target.

The design files provided with this document contain an IAP driver and DirectC programming files to enable programming of the FPGA fabric and eNVM. For more information about DirectC, refer to the [DirectC User's Guide](#).

The SoftConsole projects that are provided can be used for any of the methods explained in this document.

## Requirements and Recommendations

- Stable system power is needed during IAP block programming. When you are designing a system, make sure that you have taken system-level design considerations into account to avoid IAP program failures due to power interruptions. If an IAP program fails due to power interruptions, you need to restart the IAP block programming.
- Disable the Watchdog timer in the software or handle the Watchdog interrupt.
- Implement the error detection and correction mechanism to ensure programming data integrity.
- The application has to take care of the failsafe mechanism for IAP block programming directly from the host PC using UART or Ethernet.
- Ensure the settings in Serial Terminal Emulation Program. Refer to the [Configuring Serial Terminal Emulation Programs](#) tutorial for configuring HyperTerminal, Tera Term, or PuTTY.
- When you are running the design in A2F200 evaluation kit board, use files from the path C:\A2F\_AC362\_DFA2F\_200.
- When you are running the design in A2F500 development kit board, use files from the path C:\A2F\_AC362\_DFA2F\_500.
- When you regenerate the MSS component from Libero SoC project, ensure the following lines commented out in softconsole project workspace.

hw\_platform library module:

File name : DirectC/dpcom.c

Line number 125: `//#error Please add code here to get the programming data. Please refer to the Required Source Code Modifications section of the DirectC user's guide.`

File name : DirectC/dpuser.c

Line number 15 : `// #include "main.h"`

Line number 48 : `//#error Please modify this function to time delays. Please refer to the Required Source Code Modifications section of the DirectC user's guide.`

## Known Issues

The IAP block does not have control over the BSR register and, as a result, fabric I/Os cannot be set to any logic level or maintain their last known state. When IAP programming is enabled, fabric I/Os are tristated. Do not attempt any BSR instruction since the result is not predictable.

## Design Overview

The intent of this application note is to describe IAP and DirectC and demonstrate the capability of the SmartFusion cSoC device to program the FPGA fabric and eNVM using the IAP block. Three different solutions for programming the FPGA fabric and eNVM are presented:

"Solution 1: Programming FPGA Fabric/eNVM Directly from the Host PC Using UART"

"Solution 2: Programming FPGA Fabric Directly From the Host PC Using Ethernet"

"Solution 3: Programming FPGA Fabric/eNVM From the On-Board SPI Flash"

The design examples attached to this application note can be used as a reference when you need to program the FPGA fabric or eNVM using the IAP interface.

A basic understanding of the SmartFusion design flow is assumed.

Refer to the [SmartFusion Microcontroller Subsystem User's Guide](#) and [Using UART with SmartFusion](#) along with the [Libero SoC v10.0 User's Guide](#) to understand the SmartFusion design flow.

In the design examples, the programming file for the FPGA fabric or eNVM can only be from one of the following MSS communication peripherals:

- MSS\_UART
- MSS\_SPI
- MSS\_EMAC

The programming data bitstream can be transferred through any of the communication peripherals (UART, SPI, I<sup>2</sup>C, or EMAC) by the Cortex-M3 processor in packets required to program a single row. DirectC running on the Cortex-M3 processor shifts the programming data bit stream in sets of 128 bits into the IAP block and commands the IAP block through the DirectC and STAPL programming algorithm.

Programming error conditions are generated from the programming controller registers and interpreted by the programming algorithm in the firmware application.

## Design Example

The SmartFusion FPGA fabric or eNVM can be programmed using the IAP interface on the SmartFusion Development Kit Board and the SmartFusion Evaluation Kit Board. This design example explains three different solutions:

"Solution 1: Programming FPGA Fabric/eNVM Directly from the Host PC Using UART"

"Solution 2: Programming FPGA Fabric Directly From the Host PC Using Ethernet"

"Solution 3: Programming FPGA Fabric/eNVM From the On-Board SPI Flash"

### FPGA Fabric

The FPGA fabric can be programmed using any of the three solutions listed above.

- "Solution 1: Programming FPGA Fabric/eNVM Directly from the Host PC Using UART" uses the UART interface as a medium to transfer the programming file \*.dat data when requested by IAP to program the FPGA fabric.
- "Solution 2: Programming FPGA Fabric Directly From the Host PC Using Ethernet" uses the EMAC interface as a medium to transfer the programming file \*.dat data requested by IAP to program the FPGA fabric.
- "Solution 3: Programming FPGA Fabric/eNVM From the On-Board SPI Flash" uses the Ethernet/UART interface as a medium to transfer the programming file \*.dat to SPI flash. Later the programming file is read from SPI flash when requested by DirectC and the IAP block to program the FPGA fabric.

## eNVM

The eNVM can be programmed using Solution 1 and Solution 3 only.

- "Solution 1: Programming FPGA Fabric/eNVM Directly from the Host PC Using UART" uses the UART interface as a medium to transfer the programming file \*.dat data when requested by IAP to update eNVM.
- "Solution 3: Programming FPGA Fabric/eNVM From the On-Board SPI Flash" uses the Ethernet/UART interface as a medium to transfer the programming file \*.dat to SPI flash. Later the programming file is read from SPI flash when requested by DirectC and the IAP block to program eNVM.

Prior to running these design examples, generate the appropriate hardware and programming file \*.dat based on the solution.

## Hardware Implementation

To run the IAP application on the SmartFusion cSoC device, configure the hardware with the required MSS peripherals (UART, SPI, I<sup>2</sup>C, or EMAC) that need to be enabled. The provided hardware can be used for any of these interfaces except for I<sup>2</sup>C explained in this document.

The default hardware development flow should be followed for programming file generation. For generating the files for DirectC and IAP block programming, follow the appropriate steps given.

## DirectC (\*.dat) File Generation

### For FPGA Fabric Programming

- Create a Libero<sup>®</sup> System-on-Chip (SoC) project with the desired logic in the fabric.
- Using FlashPro, export the programming file as a DirectC file (\*.dat), using **File > Export > Export Single Programming File**.

An example DirectC file (\*.dat) that causes LEDs to blink in different fashions is provided with the design files.

### For eNVM Programming

- Create a Libero SoC project with the eNVM data client
- Using FlashPro, export the programming file as a DirectC (\*.dat) file

Refer to the [SmartFusion: Building the Executable Image in Release mode and Loading into eNVM Tutorial](#) for more information on how to create the programming file with the eNVM client.

# Solution 1: Programming FPGA Fabric/eNVM Directly from the Host PC Using UART

**Disclaimer:** You must implement a failsafe mechanism while performing IAP using this solution. This solution uses the UART interface to load the programming file from the host PC into internal memory (eSRAM) on the target board when requested by the target. This solution allows you to program the FPGA fabric or eNVM from eSRAM using the IAP interface.

The following steps are required for programming the FPGA fabric or eNVM directly from the host PC using UART.

## On the Host PC

1. Initialize the UART interface on the host PC with a baud rate that matches the UART baud rate for the image running on the target.
2. Perform the handshake with the target UART (the SmartFusion Development Kit Board or the SmartFusion Evaluation Kit Board).



3. Send the programming Action code command to initialize IAP programming. The Action code command for programming the FPGA fabric Array is 9; for programming eNVM the code is 16.
4. Read the address and number of bytes to be sent from the target UART port.
5. Send the requested amount of data from the requested address to the target.
6. Step 4 and Step 5 continue till IAP on the target completes programming FPGA fabric or eNVM with the programming file (\*.dat). Once IAP completes programming FPGA fabric or eNVM, Step 4 and Step 5 continue till IAP completes verification.

## On the Target

1. Initialize the UART on the target (the SmartFusion Development Kit Board or the SmartFusion Evaluation Kit Board) with a baud rate that matches the UART baud rate that the host tool (Memory Loader) is running on host PC.
2. Perform the handshake with the host UART.
3. Read the programming Action code command from host PC UART port into eSRAM to initialize the IAP.
4. Send the address and number of bytes to be read from the host PC UART. The host PC responds with the requested data from the requested address.
5. Receive all the data from the host PC that has been requested by the target.
6. Step 4 and Step 5 continue till DirectC completes programming of the FPGA fabric or eNVM with the programming file (\*.dat) from the host PC. Once DirectC completes programming of the FPGA fabric or eNVM, Step 4 and Step 5 continue till DirectC completes verification.



Figure 4 illustrates programming the FPGA fabric or eNVM directly from the host PC.

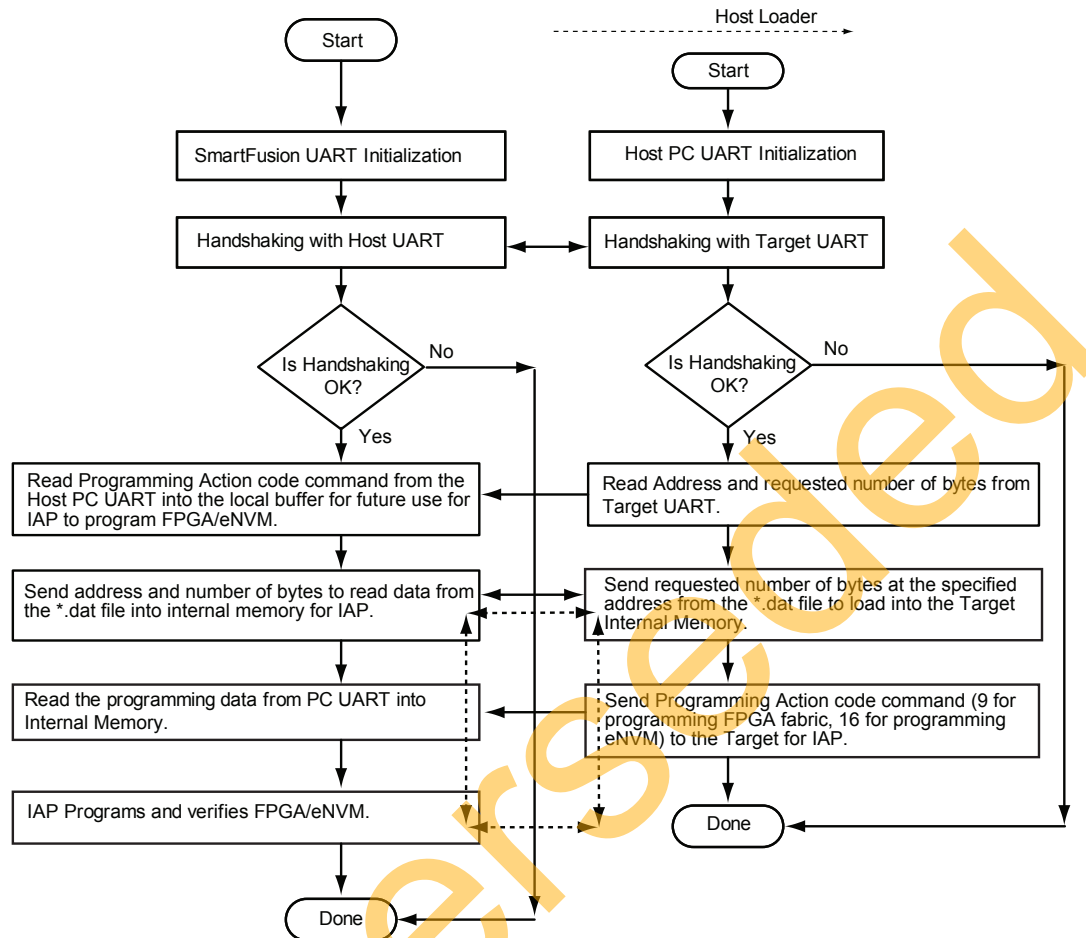


Figure 4 • Programming FPGA Directly From the Host PC Using UART

The Libero SoC project and SoftConsole project are provided in the design files attached associated with this design example (refer to "Appendix A – Design Files" on page 28). The provided Softconsole project design files are for running only in debug mode. Refer to the [SmartFusion cSoC: Building Executable Image in Release Mode and Loading into eNVM Tutorial](#) to understand on how to create the design files for running in Release mode to program FPGA fabric.

Refer to the [SmartFusion cSoC: Basic Bootloader and Field Upgrade eNVM Through IAP Interface](#) application note for running in Release mode to update eNVM.

For Hardware implementation, refer to the "Hardware Implementation" section on page 7.

## Macro Settings

In this solution, designer should not output debug messages in the code because the same UART port is used for transferring the programming file from the host. There will be a conflict with the incoming and output debug messages on the same UART port which leads to interruption in communication and stalls the IAP. To avoid conflict with the UART port, the following must be commented in the DirectC/dpuser.h file to receive the programming file from the host when requested by IAP.

```
#define ENABLE_DEBUG
```

## Board Settings

The design example works on the SmartFusion Development Kit Board and the SmartFusion Evaluation Kit Board with default board settings. Refer to the following user's guides for default board settings.

- [SmartFusion Development Kit User's Guide](#)
- [SmartFusion Evaluation Kit User's Guide](#)

## Running the Design

The design files provided for "[Solution 1: Programming FPGA Fabric/eNVM Directly from the Host PC Using UART](#)" section on page 7 are compatible with the SmartFusion Development Kit Board and the SmartFusion Evaluation Kit Board. You can refer to these files to evaluate the IAP feature in the SmartFusion cSoC device.

To program the FPGA fabric, program the SmartFusion Development Kit Board or the SmartFusion Evaluation Kit Board with the appropriate hardware design files available at:

\\Programming\_FPGAFabric\\SmartFusion\_IAP ("[Appendix A – Design Files](#)" on page 28) using FlashPro and then power cycle the board.

To program the eNVM, program the SmartFusion Development Kit Board or the SmartFusion Evaluation Kit Board with the appropriate hardware design files available at:

\\Programming\_eNVM\\SmartFusion\_IAP\_HW ("[Appendix A – Design Files](#)" on page 28) using FlashPro and then power cycle the board.

### On the Target

To program the FPGA fabric, invoke the SoftConsole IDE, double-click **Write Application Code** under **Develop Firmware** in the Libero SoC design flow window. Refer to "[Appendix A – Design Files](#)" on page 28 for more information. Select the **IAP\_UART\_From\_HostPC** project and launch the debugger.

To program eNVM, invoke the SoftConsole IDE, double-click **Write Application Code** under **Develop Firmware** in the Libero SoC design flow window. Refer to "[Appendix A – Design Files](#)" on page 28 for more information. Select the **IAP\_UART\_From\_HostPC** project and launch the debugger.

### On the Host PC

After running the debugger in the Soft Console, launch UART Host Memory Loader (**A2F\_Mem\_UARTHost\_Loader.exe**) at the command prompt.

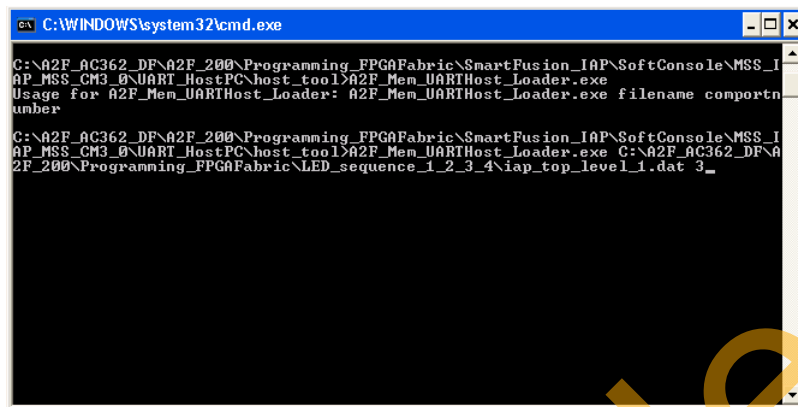
UART Host Memory Loader is an executable program (\*.exe) provided in the host tool folder. UART Host Memory Loader program runs on the host and is used to transfer the FPGA fabric or eNVM programming file (\*.dat) from the host to the board.

Verify that the UART COM port is not used by any other application, such as HyperTerminal or PuTTY.

The UART Host Memory Loader tool needs to be initiated at the command prompt. Open a command prompt window in the host PC and change to the directory where the host tool is located: \\Programming\_FPGAFabric\\SmartFusion\_IAP\\SoftConsole\\MSS\_IAP\_MSS\_CM3\_0\\UART\_HostPC\\host\_tool.

Refer to "[Appendix A – Design Files](#)" on page 28 for Solution 1.

Type **A2F\_Mem\_UARTHost\_Loader.exe** at the command prompt with the filename.dat and uart com port #. **Figure 5** shows the UART Host Memory Loader (A2F\_Mem\_UARTHost\_Loader) help for programming the FPGA fabric.



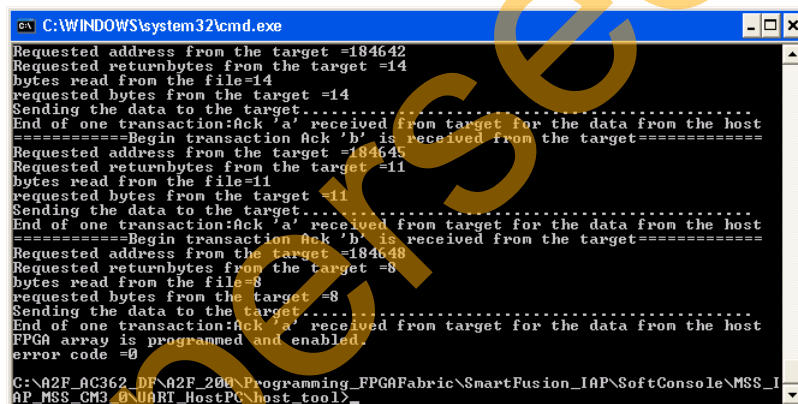
```

C:\WINDOWS\system32\cmd.exe
C:\A2F_AC362_DF\A2F_200\Programming_FPGAFabric\SmartFusion_IAP\SoftConsole\MSS_I
AP_MSS_CM3_0\UART_HostPC\host_tool>A2F_Mem_UARTHost_Loader.exe
Usage for A2F_Mem_UARTHost_Loader: A2F_Mem_UARTHost_Loader.exe filename comportsn
number
C:\A2F_AC362_DF\A2F_200\Programming_FPGAFabric\SmartFusion_IAP\SoftConsole\MSS_I
AP_MSS_CM3_0\UART_HostPC\host_tool>A2F_Mem_UARTHost_Loader.exe C:\A2F_AC362_DF\A
2F_200\Programming_FPGAFabric\LED_sequence_1_2_3_4\iap_top_level1.dat 3_

```

**Figure 5 • Launching UART Host Memory Loader on Host PC**

**Figure 6** shows the debug messages while programming the FPGA fabric.



```

C:\WINDOWS\system32\cmd.exe
Requested address from the target =184642
Requested returnbytes from the target =14
bytes read from the file=14
requested bytes from the target =14
Sending the data to the target.....
End of one transaction:Ack 'a' received from target for the data from the host
=====Begin transaction:Ack 'b' is received from the target=====
Requested address from the target =184645
Requested returnbytes from the target =11
bytes read from the file=11
requested bytes from the target =11
Sending the data to the target.....
End of one transaction:Ack 'a' received from target for the data from the host
=====Begin transaction:Ack 'b' is received from the target=====
Requested address from the target =184648
Requested returnbytes from the target =8
bytes read from the file=8
requested bytes from the target =8
Sending the data to the target.....
End of one transaction:Ack 'a' received from target for the data from the host
FPGA array is programmed and enabled.
error code =0
C:\A2F_AC362_DF\A2F_200\Programming_FPGAFabric\SmartFusion_IAP\SoftConsole\MSS_I
AP_MSS_CM3_0\UART_HostPC\host_tool>

```

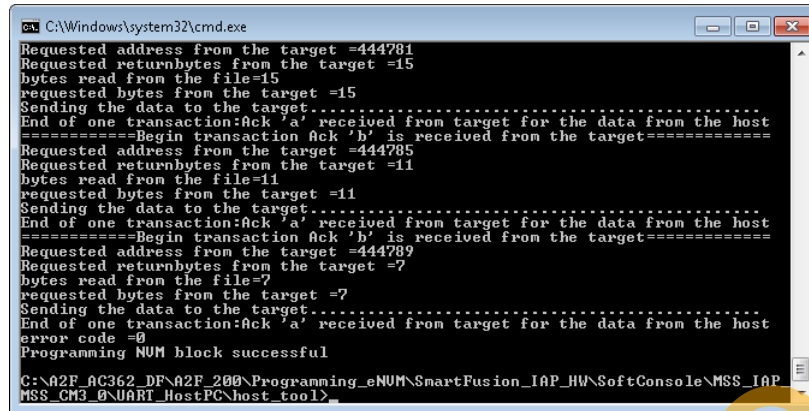
**Figure 6 • Debug Messages After Running the UART Host Memory Loader on Host PC**

You can observe the blinking LED sequence on the SmartFusion Development Kit Board or the SmartFusion Evaluation Kit Board for the \*.dat file that was launched on the command prompt. There will be a change in the running LED sequence providing a visual verification that the SmartFusion cSoC device has been reprogrammed.

For the Solution 1 of programming eNVM, the host tool is located at:  
 \Programming\_eNVM\SmartFusion\_IAP\_HWS\SoftConsole\MSS\_IAP\_MSS\_CM3\_0\UART\_HostPC.  
 Browse to this directory to launch the UART Host memory loader (A2F\_Mem\_UARTHost\_Loader.exe) at the command prompt.

**Note:** The host tool, UART Host memory loader, is different for Solution 1 of programming eNVM and Solution 1 of programming FPGA fabric.

Figure 7 shows the debug messages while programming the eNVM.



```

C:\Windows\system32\cmd.exe
Requested address from the target =444781
Requested returnbytes from the target =15
bytes read from the file=15
requested bytes from the target =15
Sending the data to the target.....
End of one transaction:Ack 'a' received from target for the data from the host
=====Begin transaction Ack 'b' is received from the target=====
Requested address from the target =444785
Requested returnbytes from the target =11
bytes read from the file=11
requested bytes from the target =11
Sending the data to the target.....
End of one transaction:Ack 'a' received from target for the data from the host
=====Begin transaction Ack 'b' is received from the target=====
Requested address from the target =444789
Requested returnbytes from the target =7
bytes read from the file=7
requested bytes from the target =7
Sending the data to the target.....
End of one transaction:Ack 'a' received from target for the data from the host
error code =0
Programming NUM block successful
C:\A2F_AC362_DF\A2F_200\Programming_eNVM\SmartFusion_IAP_HW\SoftConsole\MSS_IAP
MSS_CM3_0\UART_HostPC\host_tool>

```

**Figure 7 • Debug Messages While Programming the eNVM**

In programming the eNVM, you must restart the board to see the message on the OLED for the \*.dat file that was launched at the command prompt. You can observe the "ENVM PROGRAMMING SUCCESSFUL" message on the OLED on the SmartFusion cSoC board. This is visual evidence that the eNVM has been programmed.

## Solution 2: Programming FPGA Fabric Directly From the Host PC Using Ethernet

Disclaimer: You must implement a failsafe mechanism while performing IAP using this solution.

This solution uses Ethernet to load the programming file from the host PC into internal memory (eSRAM) on the target board when requested by the target. The design files provided for this solution support only programming of the FPGA fabric. Programming eNVM is not supported because the image used to program eNVM does not fit into internal memory (eSRAM). The image includes code for the TCP/IP stack, IAP, and DirectC.

The following steps are required for programming the FPGA fabric directly from the host PC.

### On the Host PC

1. On the host PC, initialize the socket connection with the dynamic IP address produced by the target to establish the connection for communication between the target board and host PC.
2. Read the address and number of bytes from the network that have been sent by the target and transfer the requested data over the network.
3. Step 2 continues till IAP on the target completes programming FPGA fabric with the programming file (\*.dat). Once IAP completes programming the FPGA fabric, Step 2 continues till IAP completes verification.

### On the Target

1. On the target, initialize the EMAC and TCP/IP stack. This initialization produces dynamic IP for communication with the host PC.
2. Send the address and number of bytes over the network to read the \*.dat file from the host PC.
3. Read the programming data from the EMAC interface on the target board that has been sent by the host PC over the network in the form of packets.

4. Step 2 and Step 3 continue till IAP completes programming of the FPGA fabric with the programming, (\*.dat) file from the host. Once IAP completes programming of the FPGA fabric, Step 2 and Step 3 continue till IAP completes verification.

Figure 8 illustrates programming the FPGA fabric directly from the host PC using Ethernet.

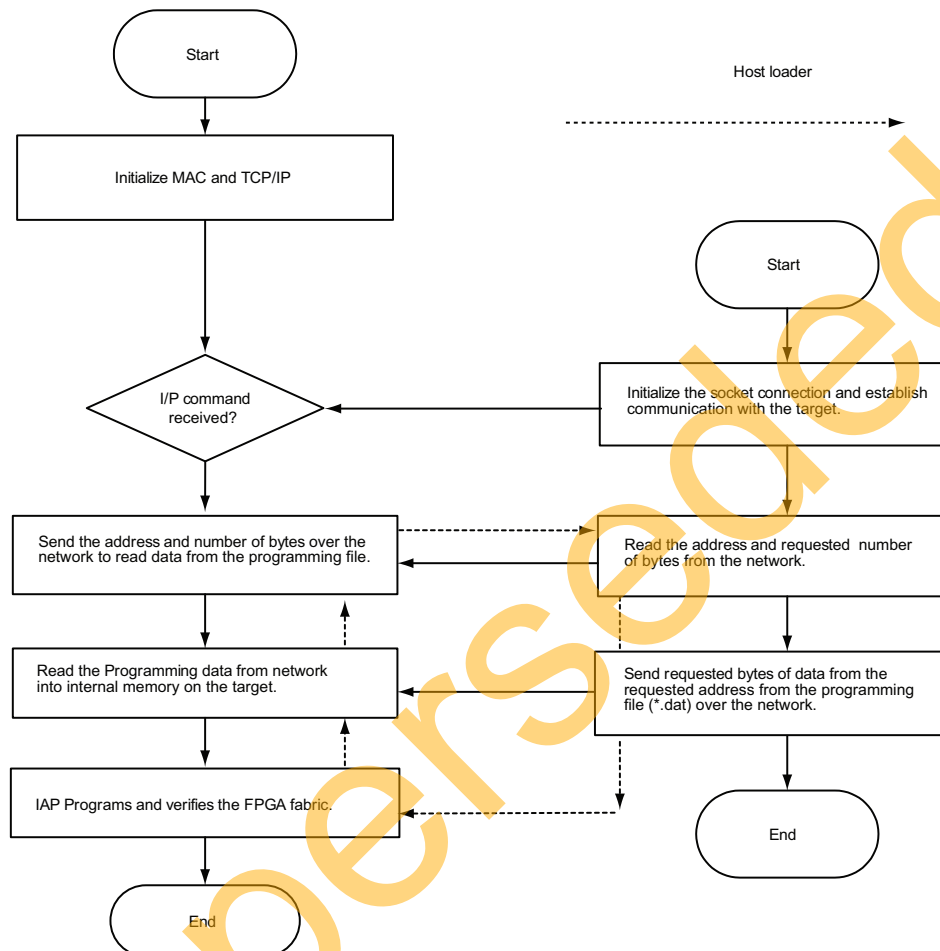


Figure 8 • Programming FPGA Directly From the Host PC Using Ethernet

The Libero SoC project is provided in the design files attached to this design example. Refer to "Appendix A – Design Files" on page 28.

The Softconsole project design files are for running only in debug mode. Refer to the [SmartFusion cSoC: Building Executable Image in Release Mode and Loading into eNVM Tutorial](#) to understand how to create the design files for running in Release mode to program FPGA fabric.

For hardware implementation, refer to the "Hardware Implementation" section on page 7.

## Running the Design

The design file provided for "Solution 2: Programming FPGA Fabric Directly From the Host PC Using Ethernet" section on page 12 is compatible only with the SmartFusion Development Kit Board because the 50 MHz clock source for the Ethernet PHY is sourced by an FPGA I/O on the SmartFusion Development Kit Board. You can use these files to evaluate the IAP feature in SmartFusion cSoC devices.

Program the SmartFusion Development Kit Board with the appropriate hardware design files available at Programming\_FPGAFabric\Ethernet\_Host\_PC\_IAP ("Appendix A – Design Files" on page 28) using



FlashPro, and then power cycle the board. This solution uses the hardware EMAC and supplied software for the uIP TCP/IP stack to transfer the FPGA fabric programming file (\*.dat) through the Ethernet network into internal memory (eSRAM) on the target. Socket application programming is being used on both the target and host PC. TCP/IP manages the requests in the form of packets between the SmartFusion cSoC target and host.

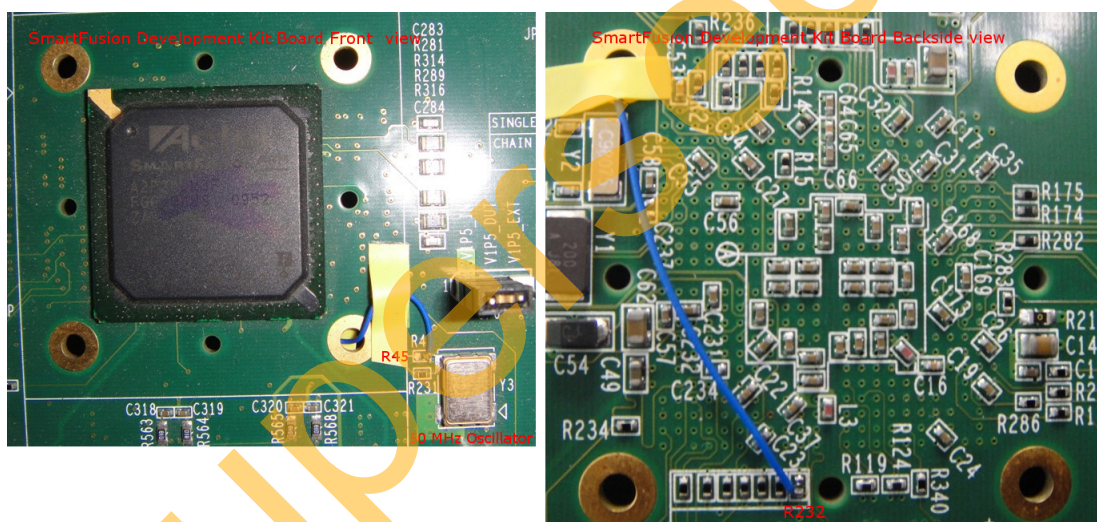
### Board Setup

To demonstrate this solution, rework is required on the SmartFusion Development Kit Board. When DirectC and the IAP block begin to program the FPGA fabric, the 50 MHz clock provided to the Ethernet PHY module through pin F2 is disconnected, since the FPGA I/Os are tristated or held at predefined or last known state during FPGA programming. Due to this clock loss for Ethernet communications, the IAP program is stalled as the programming file has not been fully downloaded to the SmartFusion cSoC device.

The workaround is an external 50 MHz clock source that needs to be fed to MAC\_CLK and RMII PHY clock to avoid the communication interruption. The provided solution works only on the SmartFusion Development Kit Board as there is an external 50 MHz clock source available.

The following method is a proven solution for routing the external clock to RMII PHY clock (pin number 34) with the SmartFusion Development Kit Board.

Route the 50 MHz clock from the 50 MHz oscillator to the RMII PHY. This can be done with a small amount of board rework. Solder a wire from the R45, which is next to the 50 MHz oscillator (Y3), to the resistor R232.



**Figure 9 • Modified Board**

Figure 9 shows routing of the 50 MHz clock from the external clock to the RMII PHY clock.

For this method, use the following hardware files supplied with the design files in this location:

\\Programming\_FPGAFabric\Ethernet\_Host\_PC\_IAP.

**Note:** When you are designing a system, make sure that you are feeding an external 50 MHz clock to MAC\_CLK and RMII PHY.

Figure 10 illustrates the ethernet clocking scheme for a system with IAP.

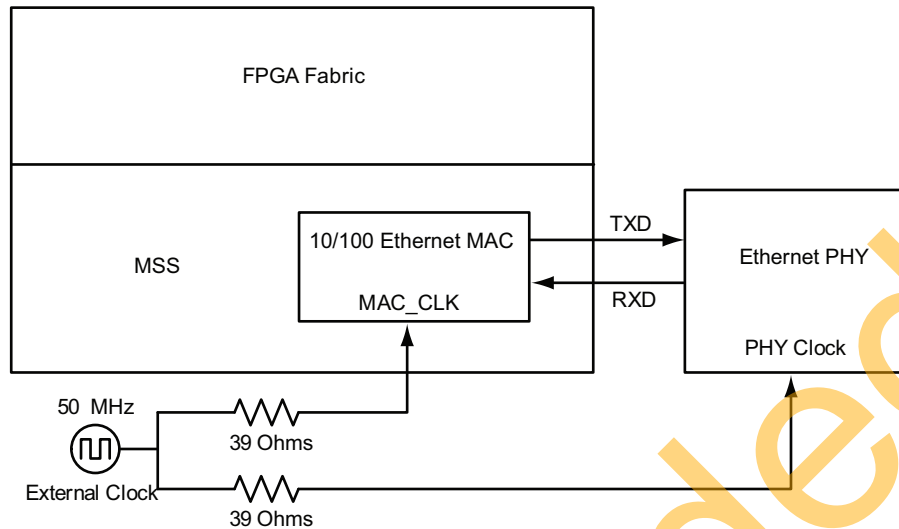


Figure 10 • Ethernet Clocking Scheme for a System With IAP

### On the Target

To program the FPGA fabric, invoke the SoftConsole IDE, double-click **Write Application Code** under **Develop Firmware** in the Libero SoC design flow window. See "Appendix A – Design Files" on page 28 for more information. Select the project Ethernet\_From\_HostPC\_IAP and launch the debugger.

### On the Host PC

Start a HyperTerminal session with a 57600 baud rate, 8 data bits, 1 stop bit, no parity, and no flow control. If your computer does not have the HyperTerminal program, use any free serial terminal emulation program, such as PuTTY or Tera Term. Refer to the [Configuring Serial Terminal Emulation Programs](#) tutorial for configuring HyperTerminal, Tera Term, or PuTTY. When you run the debugger in SoftConsole, the application starts printing the target IP address on HyperTerminal. Figure 11 shows HyperTerminal with the dynamic IP address and IAP options.

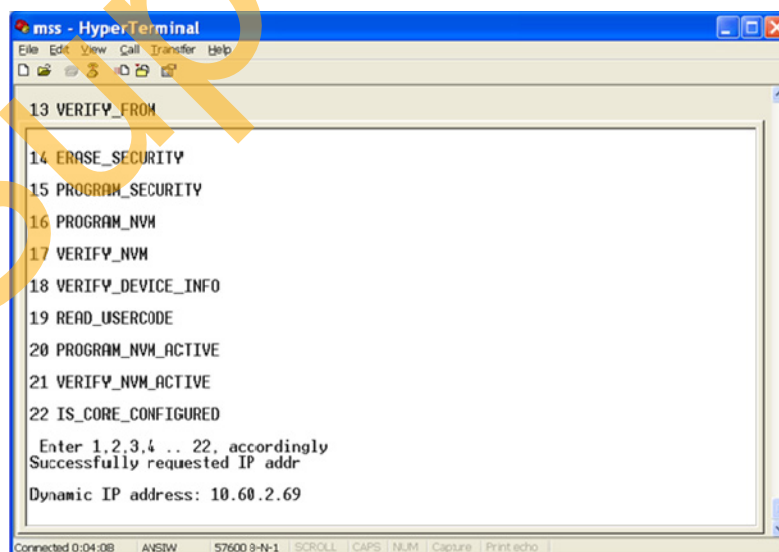


Figure 11 • HyperTerminal with the Dynamic IP Address and IAP Options



After running the debugger in SoftConsole, launch the Ethernet Host Memory Loader (A2F\_Mem\_EthernetHost\_Loader.exe) with the required input parameters on the command prompt, as shown in [Figure 12](#).

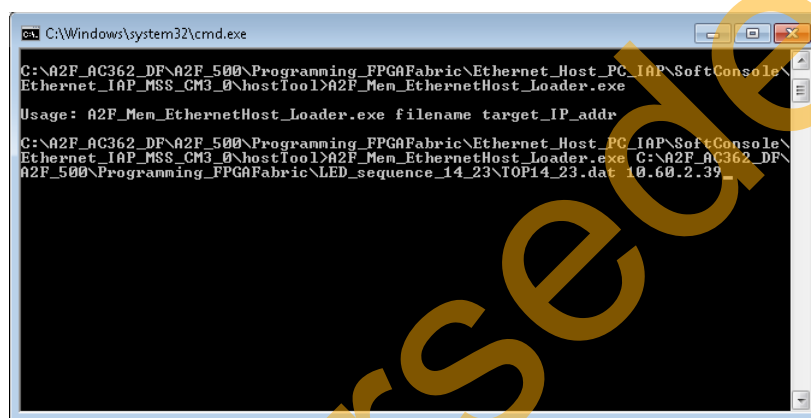
Ethernet Host Memory loader is an executable program (\*.exe) provided in the host tool folder. It runs on the host to establish socket communication between the target and host PC to transfer the FPGA fabric programming file (\*.dat) over network to the target.

Open a command prompt window in the host PC and change to the directory where the host tool is located:

\\Programming\_FPGAFabric\\Ethernet\_Host\_PC\_IAP\\SoftConsole\\Ethernet\_IAP\_MSS\_CM3\_0\\hostTool.  
Refer to "[Appendix A – Design Files](#)" on page 28.

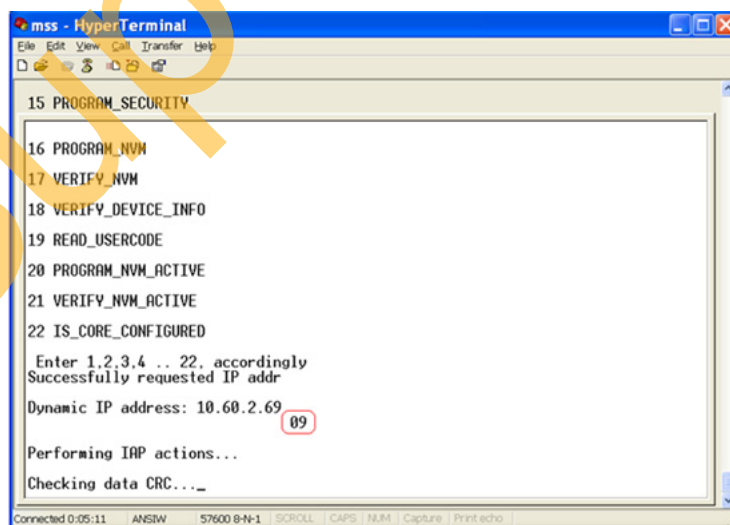
Type **A2F\_Mem\_EthernetHost\_Loader.exe**. Help on how to run the Ethernet Host Memory loader is printed.

[Figure 12](#) shows the Ethernet Host Memory Loader (A2F\_Mem\_EthernetHost\_Loader) help for programming the FPGA fabric.



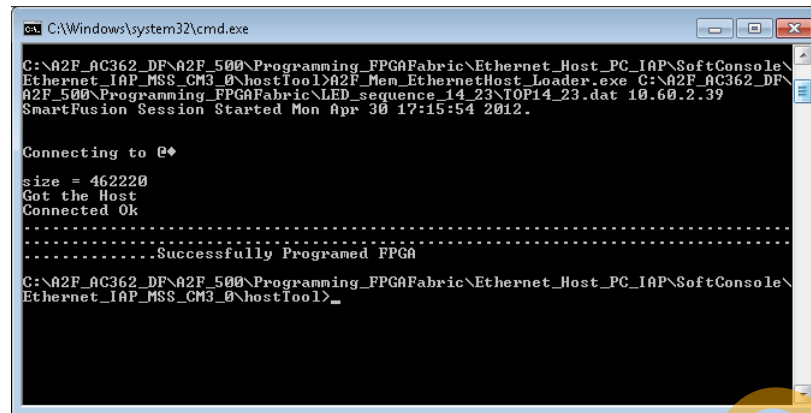
**Figure 12 • Launching the Ethernet Host Memory Loader**

After launching the Ethernet Host Memory Loader, choose option 09 in HyperTerminal to program the FPGA fabric, as shown in [Figure 13](#).



**Figure 13 • HyperTerminal after Choosing Option 09 for Programming FPGA Fabric**

The debug messages are printed during the FPGA fabric programming, as shown in Figure 14.



```

C:\Windows\system32\cmd.exe
C:\A2F_AC362_DF\A2F_500\Programming_FPGAFabric\Ethernet_Host_PC_IAP\SoftConsole\Ethernet_IAP_MSS_CM3_0\hostTool>A2F_Mem_EthernetHost_Loader.exe C:\A2F_AC362_DF\A2F_500\Programming_FPGAFabric\LED_sequence_14_23\IOP14_23.dat 10.60.2.39
SmartFusion Session Started Mon Apr 30 17:15:54 2012.

Connecting to E+
size = 462220
Got the Host
Connected Ok
.....Successfully Programed FPGA
C:\A2F_AC362_DF\A2F_500\Programming_FPGAFabric\Ethernet_Host_PC_IAP\SoftConsole\Ethernet_IAP_MSS_CM3_0\hostTool>_
  
```

**Figure 14 • Debug Messages While Programming the FPGA Fabric**

You can observe the blinking LED sequence on the SmartFusion cSoC board for the \*.dat file that was launched on the command prompt. There is a change in the running LED sequence. This is visual evidence of the new design that has been programmed into the FPGA fabric.

## Solution 3: Programming FPGA Fabric/eNVM From the On-Board SPI Flash

Disclaimer: You must implement a failsafe mechanism while performing IAP using this solution.

This solution uses SPI flash to load the complete programming file (\*.dat) into the SPI flash. The software application reads the programming file (\*.dat) from the on-board SPI flash into internal memory (eSRAM) to program the FPGA fabric or eNVM.

This solution consists of two steps.

Step 1 of the solution is to download the programming file into the external memory.

There are 2 options for this step:

1. Using the UART interface, load the FPGA fabric or the eNVM programming file into an on-board SPI flash with no file system.
2. Using the Ethernet interface, load the FPGA fabric or the eNVM programming file into an on-board SPI flash file system.

Step 2 programs the FPGA fabric or eNVM using the IAP interface from the on-board SPI utilizing option 1 or 2 from Step 1.

### Step 1: Downloading the Programming File

#### Option 1: Using UART Interface

1. Initialize the UART on the host PC with a baud rate that matches the UART baud rate with the image that is running on the target.
2. Perform the handshake with the target UART (SmartFusion Development Kit Board or the SmartFusion Evaluation Kit Board).
3. Transfer the programming file to the on-board SPI flash through the SmartFusion cSoC device on the target until the entire file has been transferred to the SPI flash.

#### On the Target

1. Initialize the UART on the target (SmartFusion Development Kit Board or the SmartFusion Evaluation Kit Board) with a baud rate that matches the UART baud rate with the host tool (Memory Loader) that is running on the host PC and also initialize the SPI.

2. Perform the handshake with the host PC UART.
3. Receive the programming file data from the host PC UART into SPI flash until the entire file has been received.

Figure 15 shows loading the FPGA fabric or eNVM programming file into SPI flash through UART.

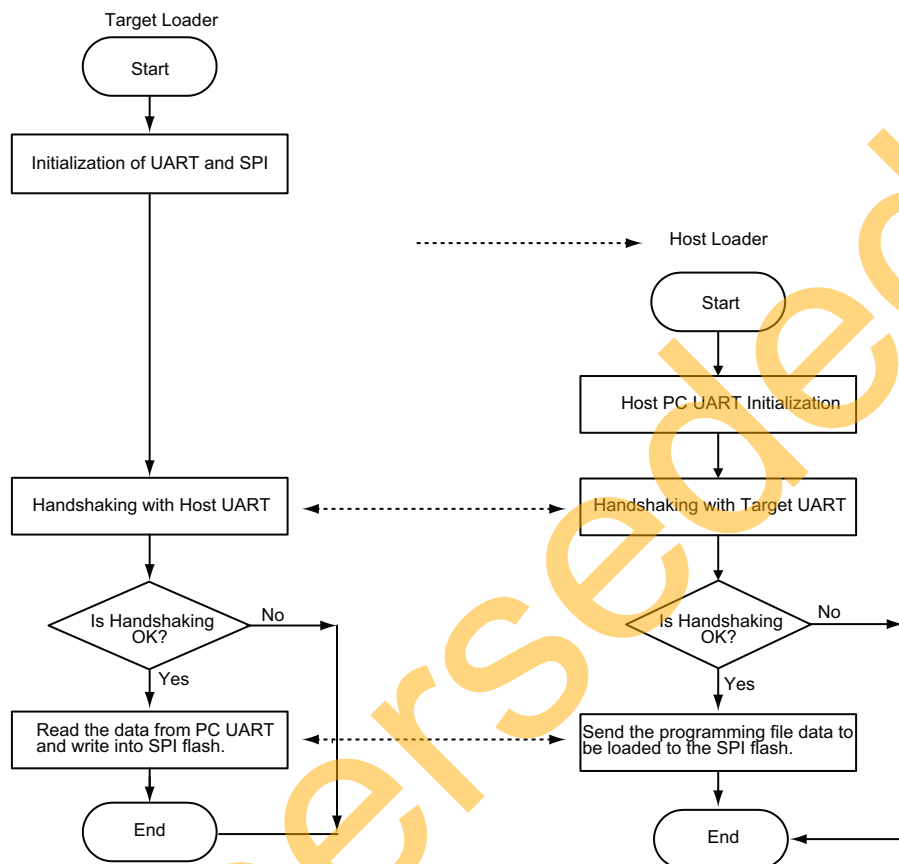


Figure 15 • SPI Flash Programming through UART

### Option 2: Using Ethernet TFTP

This method allows you to program the SPI flash file system with an FPGA fabric file. Programming eNVM is not supported because the image used to program eNVM does not fit into internal memory (eSRAM). The eNVM programming image includes DirectC, IAP, SPI flash drivers, TFTP, lwIP, and the file system code.

The following steps are required for loading the programming file from the host PC into the on-board SPI flash file system using TFTP.

#### On the Host PC

1. Run the TFTP client on the host PC to initialize the socket connection with the given IP address that is generated by the TFTP server (target).
2. The programming data file from the host PC is transferred over the network to the SPI flash on the target.

#### On the Target

1. Initialize the EMAC, SPI, lwIP, and the file system on the target board (the SmartFusion Development Kit Board or the SmartFusion Evaluation Kit Board).
2. Upon receiving the TFTP IP command, receive the whole programming data file from the host PC into the target board SPI flash on which the file system is residing.

Figure 16 shows loading the FPGA fabric programming file into SPI flash through the TFTP protocol.

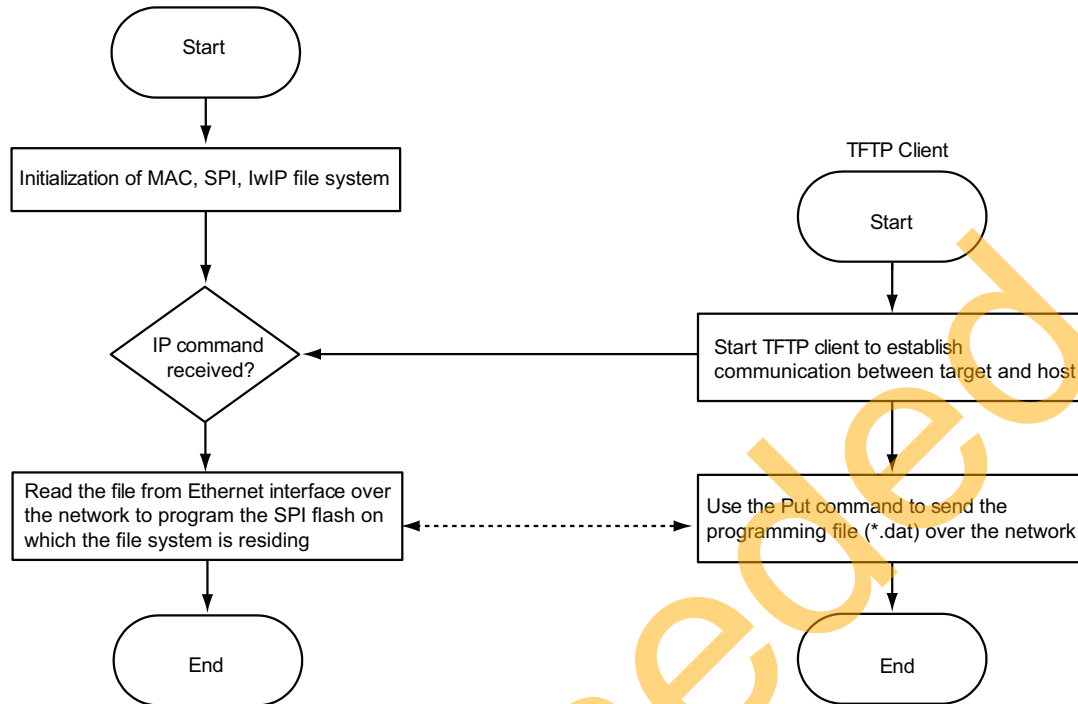


Figure 16 • Programming SPI Flash through TFTP Protocol

## Step 2: Programming the FPGA Fabric or eNVM

For programming the FPGA fabric, the IAP interface is used to program the FPGA fabric with the programming file (\*.dat) that is residing in the on-board SPI flash file system on the target board.

Once IAP completes programming FPGA fabric, IAP does verification of the programmed data with the source. For programming the eNVM, the IAP interface is used to program the eNVM with the programming file (\*.dat) that is residing in on-board SPI flash with no file system present.

Once IAP completes programming the eNVM, IAP does verification of the programmed data with the source.

The Libero SoC project is provided in the design files attached to this design example ("Appendix A – Design Files" on page 28).

The Softconsole project design files are for running only in debug mode. Refer to the [SmartFusion cSoC: Building Executable Image in Release Mode and Loading into eNVM Tutorial](#) to understand how to create the design files for running in releasing mode.

For Hardware implementation, refer to the "Hardware Implementation" section on page 7.

## Macro Settings

The following Macros are valid only for "Solution 3: Programming FPGA Fabric/eNVM From the On-Board SPI Flash". They must be enabled/disabled in the SPI flash API file (BSP\bsp\_config.h) to enable the appropriate board. The design example works with both the SmartFusion Development Kit Board and the SmartFusion Evaluation Kit Board.

1. `#define SPI_FLASH_ON_SF_DEV_KIT 1`: This Macro enables the SPI flash driver software for the SmartFusion Development Kit Board.
2. `#define SPI_FLASH_ON_SF_EVAL_KIT 1`: This Macro enables the SPI flash driver software for the SmartFusion Evaluation Kit Board.

Comment the Macro based on the board used to run this example.

## Board Settings

The design example is made to be working on SmartFusion development kit board and SmartFusion evaluation kit board with default board settings. Refer the following user's guides for default board settings.

- [SmartFusion Development Kit User's Guide](#)
- [SmartFusion Evaluation Kit User's Guide](#)

## Running the Design

The design files provided for Solution 3 are compatible with the SmartFusion Development Kit Board and the SmartFusion Evaluation Kit Board. You can use these files to evaluate the IAP feature in the SmartFusion cSoC.

To program FPGA fabric, program the SmartFusion Development Kit Board or the SmartFusion Evaluation Kit Board with the appropriate hardware design files available at: \Programming\_FPGAFabric\SmartFusion\_IAP ("[Appendix A – Design Files](#)" on page 28) using FlashPro, and then power cycle the board.

To program the eNVM, program the SmartFusion Development Kit Board or the SmartFusion Evaluation Kit Board with the appropriate hardware design files available at: \Programming\_eNVM\SmartFusion\_IAP\_HW ("[Appendix A – Design Files](#)" on page 28) using FlashPro, and then power cycle the board.

This solution consists of two steps:

- The first step is loading the SPI flash with the FPGA fabric or eNVM programming file from the host PC.
- The second step is to program the FPGA fabric or eNVM with the programming file (\*.dat) that is residing in on-board SPI flash.

## Step 1

Load the SPI flash with the FPGA fabric/eNVM programming file from the host PC. This can be done by two methods: using the UART interface or using the Ethernet TFTP protocol.

### Using the UART Interface

This method allows programming of both FPGA fabric and eNVM files into SPI flash.

To program the FPGA fabric, invoke the SoftConsole IDE, double-click **Write Application Code** under **Develop Firmware** in the Libero SoC design flow window.

Refer to the "[Appendix A – Design Files](#)" on page 28 for more information. Select the project UART\_SPIFlashLoad and launch the debugger. To program the eNVM, invoke the SoftConsole IDE, double-click **Write Application Code** under **Develop Firmware** in the Libero SoC design flow window.

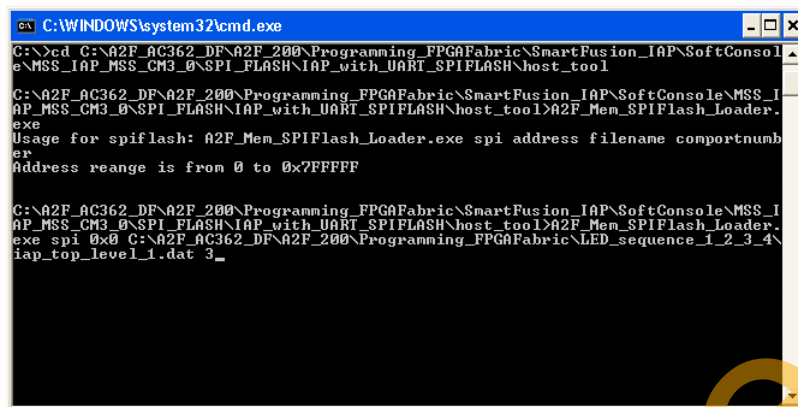
Refer to the "[Appendix A – Design Files](#)" on page 28 for more information. Select the project SPIFlashLoad\_UART and launch the debugger. After running the debugger in SoftConsole, launch the SPI Flash Memory Loader (A2F\_Mem\_SPIFlash\_Loader.exe) at the command prompt.

Open a command prompt window in the host PC and change to the directory where the host tool is located:

```
\Programming_FPGAFabric\SmartFusion_IAP\SoftConsole\MSS_IAP_MSS_CM3_0\SPI_FLASH\IAP_with_UART_SPIFLASH\host_tool.
```

Type **A2F\_Mem\_SPIFlash\_Loader.exe**; help on how to run the SPI Flash Memory Loader is printed.

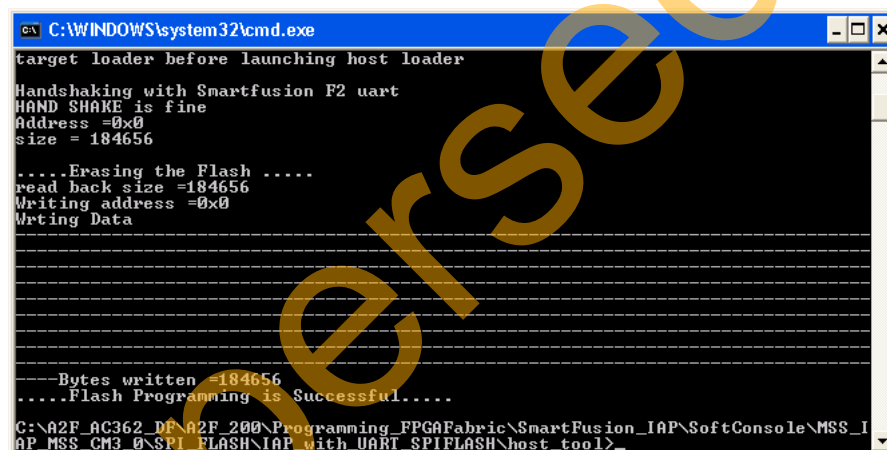
Figure 17 shows the SPI Flash Memory Loader (A2F\_Mem\_SPIFlash\_Loader) help for programming the FPGA fabric.



```
C:\WINDOWS\system32\cmd.exe
C:\>cd C:\A2F_AC362_DF\A2F_200\Programming_FPGAFabric\SmartFusion_IAP\SoftConsole\MSS_IAP_MSS_CM3_0\SPI_FLASH\IAP_with_UART_SPIFLASH\host_tool
C:\A2F_AC362_DF\A2F_200\Programming_FPGAFabric\SmartFusion_IAP\SoftConsole\MSS_IAP_MSS_CM3_0\SPI_FLASH\IAP_with_UART_SPIFLASH\host_tool>A2F_Mem_SPIFlash_Loader.exe
Usage for spiflash: A2F_Mem_SPIFlash_Loader.exe spi address filename comportnumber
Address range is from 0 to 0x7FFFFF
C:\A2F_AC362_DF\A2F_200\Programming_FPGAFabric\SmartFusion_IAP\SoftConsole\MSS_IAP_MSS_CM3_0\SPI_FLASH\IAP_with_UART_SPIFLASH\host_tool>A2F_Mem_SPIFlash_Loader.exe spi 0x0 C:\A2F_AC362_DF\A2F_200\Programming_FPGAFabric\LED_sequence_1_2_3_4\iap_top_level_1.dat 3_
```

Figure 17 • Launching SPI Flash Memory Loader

Figure 18 shows the debug messages while programming SPI flash.



```
C:\WINDOWS\system32\cmd.exe
target loader before launching host loader
Handshaking with Smartfusion F2 uart
HAND SHAKE is fine
Address =0x0
size = 184656
.....Erasing the Flash .....
read back size =184656
Writing address =0x0
Writing Data
.....
.....Bytes written =184656
.....Flash Programming is Successful.....
C:\A2F_AC362_DF\A2F_200\Programming_FPGAFabric\SmartFusion_IAP\SoftConsole\MSS_IAP_MSS_CM3_0\SPI_FLASH\IAP_with_UART_SPIFLASH\host_tool>
```

Figure 18 • Debug Messages While Programming SPI Flash

When programming eNVM, open a command prompt window in the host PC and change to the directory where the host tool is located:

\Programming\_eNVM\SmartFusion\_IAP\_HW\SoftConsole\MSS\_IAP\_MSS\_CM3\_0\IAP\_with\_UART\_SPIFLASH\host\_tool.

Launch the SPI Flash Memory Loader (A2F\_Mem\_SPIFlash\_Loader.exe) at the command prompt.

**Note:** The host tool, SPI Flash Memory Loader, is different for Solution 3 of programming eNVM and for Solution 3 of programming FPGA fabric.

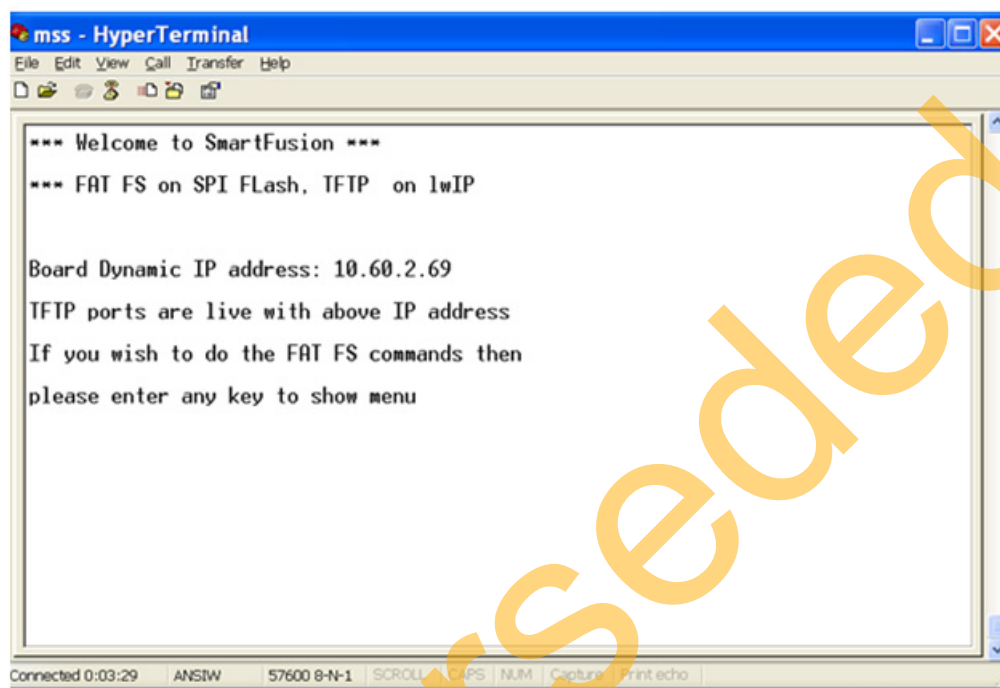
### Using Ethernet TFTP

This section describes using the Ethernet TFTP protocol to get the FPGA fabric programming file from the remote PC or local PC and programming the SPI flash.

The TFTP server on the target, which includes a hardware EMAC, lwIP, and the file system on the storage area (SPI flash) allows file transfer capability to-and-from the storage area file system resident on the target board to the host PC.

The TFTP server generates a dynamic IP address which can be used to establish communication with the TFTP client running on the host PC. TFTP clients can transfer data remotely over the network from anywhere to the target.

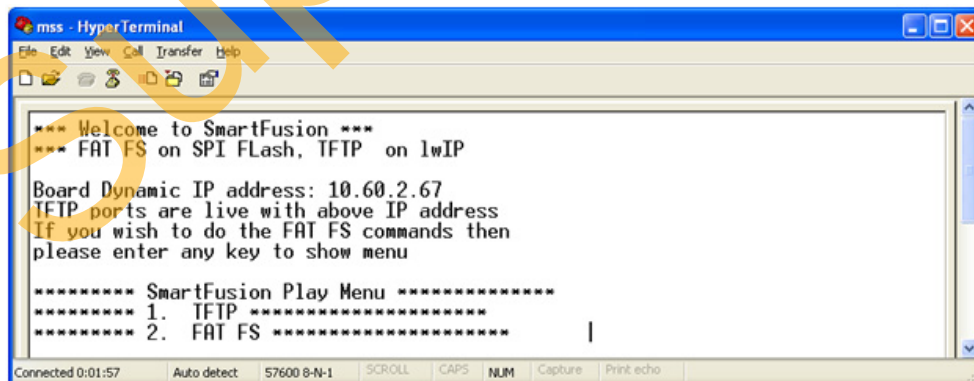
To program the FPGA fabric, invoke the SoftConsole IDE, double-click **Write Application Code** under **Develop Firmware** in the Libero SoC design flow window. Refer to "Appendix A – Design Files" on page 28 for more information. Select the project Ethernet\_SPIFlashLoad and launch the debugger. When you run the debugger in SoftConsole, the application prints the IP address on HyperTerminal, as shown in Figure 19.



**Figure 19 • HyperTerminal with Dynamic IP**

After entering any key, the play menu of the SmartFusion cSoC device is displayed. Figure 20 shows the menu with two options.

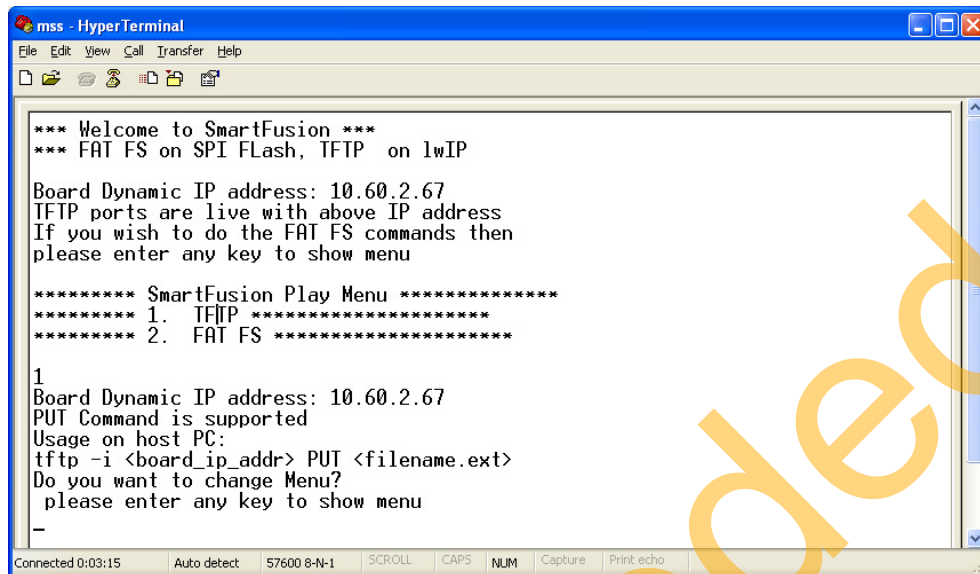
1. TFTP
2. FatFs



**Figure 20 • HyperTerminal with TFTP and FatFs Options**



Choose option 1 to display the usage command on the host PC for TFTP. Figure 21 shows the usage command for TFTP after choosing option 1.



```

*** Welcome to SmartFusion ***
*** FAT FS on SPI Flash, TFTP on lwIP

Board Dynamic IP address: 10.60.2.67
TFTP ports are live with above IP address
If you wish to do the FAT FS commands then
please enter any key to show menu

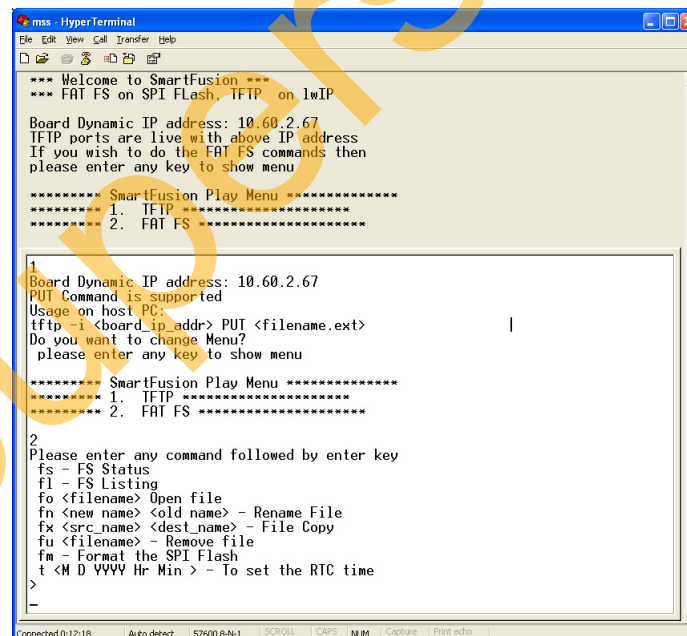
***** SmartFusion Play Menu *****
***** 1. TFTP *****
***** 2. FAT FS *****

1
Board Dynamic IP address: 10.60.2.67
PUT Command is supported
Usage on host PC:
tftp -i <board_ip_addr> PUT <filename.ext>
Do you want to change Menu?
please enter any key to show menu
-

```

Figure 21 • HyperTerminal with TFTP Usage Command

Enter any command to show the main menu. Choose option 2 to see the file system commands. Figure 22 shows the supported file system commands after choosing option 2.



```

*** Welcome to SmartFusion ***
*** FAT FS on SPI Flash, TFTP on lwIP

Board Dynamic IP address: 10.60.2.67
TFTP ports are live with above IP address
If you wish to do the FAT FS commands then
please enter any key to show menu

***** SmartFusion Play Menu *****
***** 1. TFTP *****
***** 2. FAT FS *****

1
Board Dynamic IP address: 10.60.2.67
PUT Command is supported
Usage on host PC:
tftp -i <board_ip_addr> PUT <filename.ext>
Do you want to change Menu?
please enter any key to show menu

***** SmartFusion Play Menu *****
***** 1. TFTP *****
***** 2. FAT FS *****

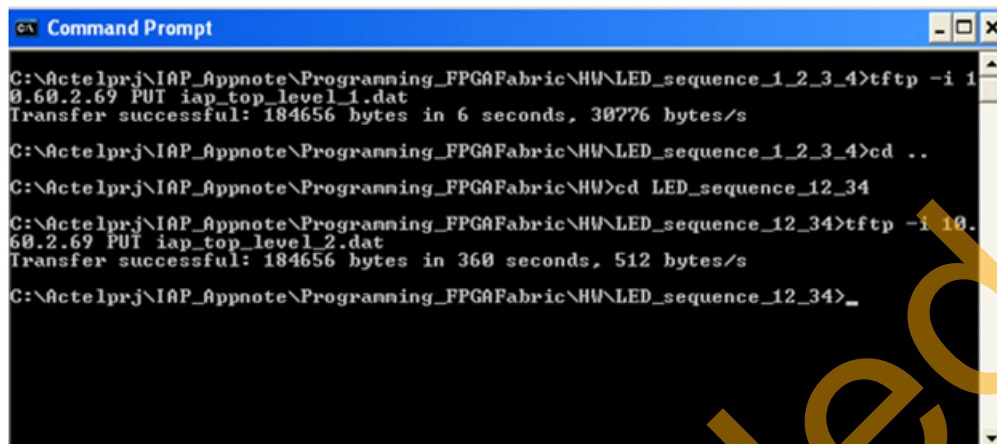
2
Please enter any command followed by enter key
fs - FS Status
fl - FS Listing
fo <filename> Open file
fn <new name> <old name> - Rename File
fx <src name> <dest name> - File Copy
fu <filename> - Remove file
fm - Format the SPI Flash
t <M D VVVV Hr Min> - To set the RTC time
>
-

```

Figure 22 • HyperTerminal with FatFs Commands

After running the debugger in SoftConsole, launch the TFTP client at the command prompt. TFTP client should be enabled if it is disabled in your host PC.

The file system on the storage area resident on the target board allows the transfer of multiple files to-and-from the storage area. Figure 23 shows the TFTP client on the host PC.



```

C:\Actelprj\IAP_Appnote\Programming_FPGA\Fabric\HW\LED_sequence_1_2_3_4>tftp -i 10.60.2.69 PUT iap_top_level_1.dat
Transfer successful: 184656 bytes in 6 seconds, 30776 bytes/s

C:\Actelprj\IAP_Appnote\Programming_FPGA\Fabric\HW\LED_sequence_1_2_3_4>cd ..

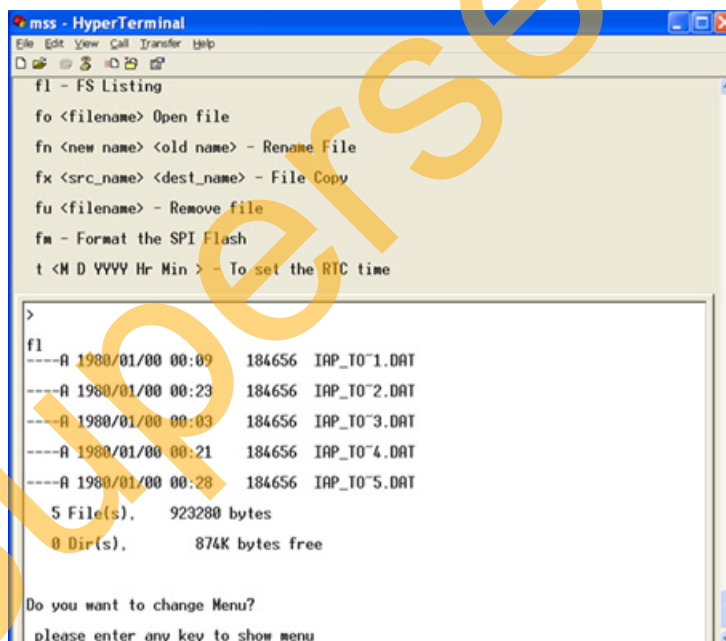
C:\Actelprj\IAP_Appnote\Programming_FPGA\Fabric\HW>cd LED_sequence_12_34

C:\Actelprj\IAP_Appnote\Programming_FPGA\Fabric\HW\LED_sequence_12_34>tftp -i 10.60.2.69 PUT iap_top_level_2.dat
Transfer successful: 184656 bytes in 360 seconds, 512 bytes/s

C:\Actelprj\IAP_Appnote\Programming_FPGA\Fabric\HW\LED_sequence_12_34>_
  
```

**Figure 23 • Launching TFTP Client on Host PC**

Type the command fl in HyperTerminal to see the list of stored files on the SPI flash present on the target board. Figure 24 shows the list of stored files on the SPI flash.



```

mss - HyperTerminal
File Edit View Call Transfer Help
[Icons]

fl - FS Listing

fo <filename> Open file
fn <new name> <old name> - Rename File
fx <src_name> <dest_name> - File Copy
fu <filename> - Remove file
fm - Format the SPI Flash
t <M D YYYY Hr Min > - To set the RTC time

>
fl
----A 1980/01/00 00:09 184656 IAP_TO~1.DAT
----A 1980/01/00 00:23 184656 IAP_TO~2.DAT
----A 1980/01/00 00:03 184656 IAP_TO~3.DAT
----A 1980/01/00 00:21 184656 IAP_TO~4.DAT
----A 1980/01/00 00:28 184656 IAP_TO~5.DAT

 5 File(s), 923280 bytes
 0 Dir(s), 874K bytes free

Do you want to change Menu?
please enter any key to show menu
  
```

**Figure 24 • List of Files Stored on the File System in SPI Flash**

This method allows programming of the SPI flash file system only with the FPGA fabric file. The eNVM file image, which includes DirectC, IAP, SPI flash drivers, TFTP, lwIP, and file system code, does not fit into the internal memory (eSRAM) to program the eNVM.

## Step 2

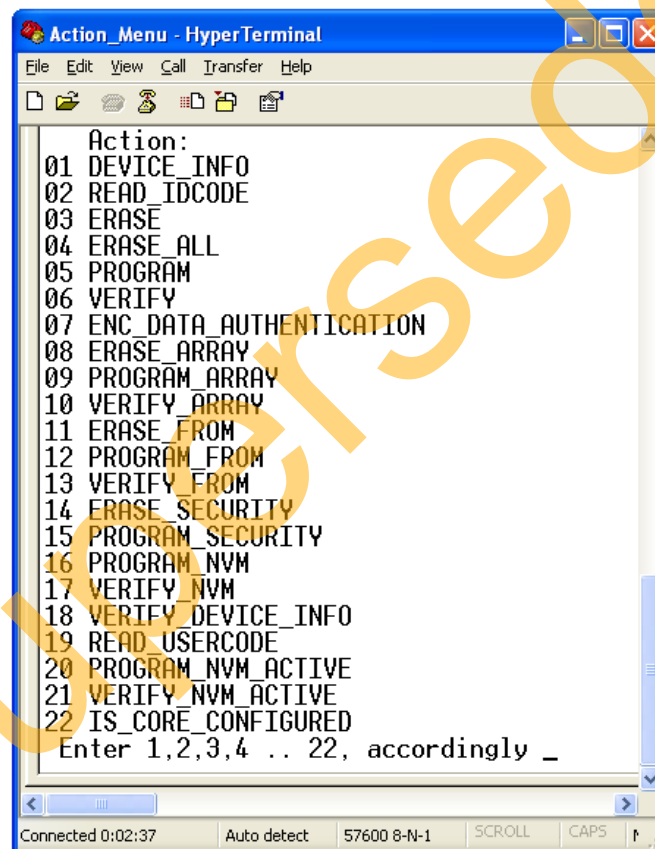
This section demonstrates the IAP interface for programming the FPGA fabric or eNVM with the on-board SPI flash contents and performing various control interactions with the FPGA fabric or eNVM using the IAP interface.

For programming the FPGA fabric, stop the debugger in the softConsole project and close the current project, UART\_SPIFlashLoad/Ethernet\_SPIFlashLoad. Select the IAP\_UART\_prj/IAP\_Ethernet\_prj project and launch the debugger.

For programming the eNVM, stop the debugger in the softConsole project and close the current project, SPIFlashLoad\_UART. Select the IAP\_Program\_prj project and launch the debugger.

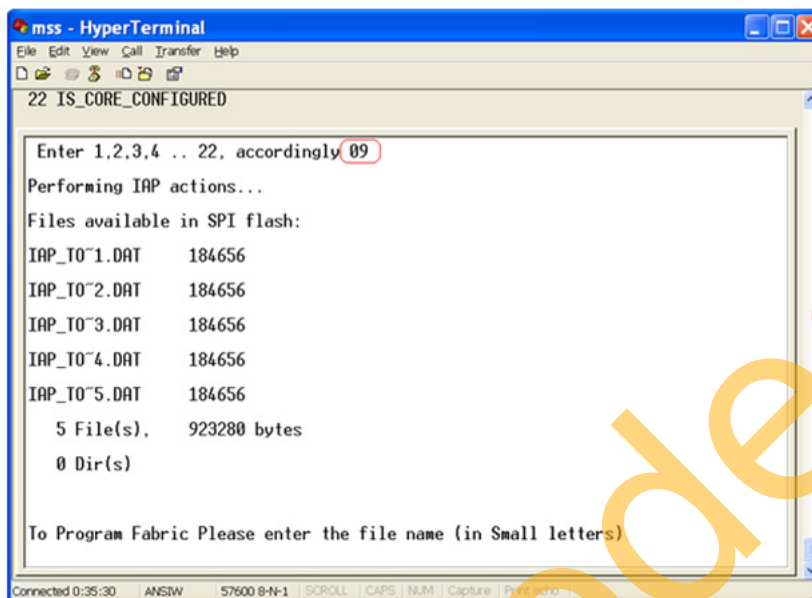
Start a HyperTerminal session with a 57600 baud rate, 8 data bits, 1 stop bit, no parity, and no flow control. If your computer does not have the HyperTerminal program, use any free serial terminal emulation program such as PuTTY or Tera Term. Refer to the [Configuring Serial Terminal Emulation Programs tutorial](#) for configuring HyperTerminal, Tera Term, or PuTTY.

When you run the debugger in SoftConsole, the application starts printing the IAP programming options on HyperTerminal. [Figure 25](#) shows HyperTerminal with IAP programming options.



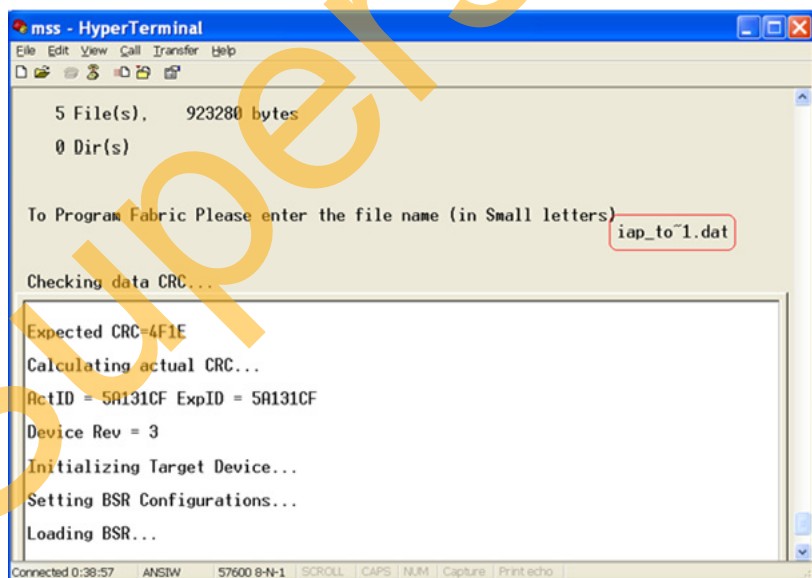
**Figure 25 • Action Menu in HyperTerminal**

You must select option 09 (program array) to program the FPGA fabric array. Figure 26 shows the debug messages when the file system is resident on the on-board SPI flash target board.



**Figure 26 • Menu After Entering 09 (Program Array)**

Figure 27 shows the debug messages after choosing one of the stored files for programming the FPGA fabric using the IAP interface.



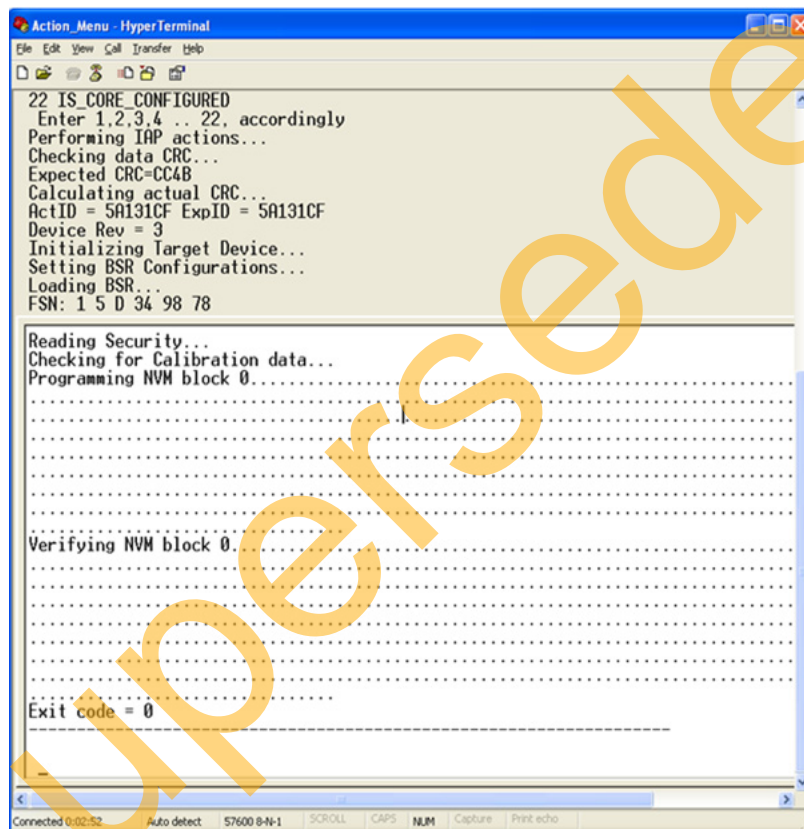
**Figure 27 • Debug Messages During the FPGA Fabric Programming**

For the UART interface method, there is no file system mounted on the SPI flash. The IAP reads the programming file content from the SPI flash into eSRAM when requested by DirectC and the IAP interface. For the TFTP method, the file system is mounted on the SPI flash. IAP reads the programming file content from the SPI flash file system into eSRAM using file I/O operations when requested by the IAP interface.

After programming the FPGA fabric, you can observe the blinking LED sequence on the SmartFusion Development Kit Board or the SmartFusion Evaluation Kit Board. There is a change in the running LED sequence. This is visual evidence that the new design has been programmed into the FPGA fabric.

As mentioned above, the eNVM programming is done with no file system mounted on SPI flash and the programming file for the eNVM is loaded into the SPI flash using the UART interface. The design files provided with this application note are used to program the eNVM using the UART interface with no file system because there is insufficient internal memory on the target board. You must choose option 16 to program the eNVM. Figure 28 shows the debug messages during the programming when no file system is mounted on the SPI flash.

After eNVM has been programmed, restart the board to see the message on the OLED for the \*.dat file that was launched on the command prompt. The message "ENVM PROGRAMMING SUCCESSFUL" on OLED is displayed on the SmartFusion Development Kit Board or the SmartFusion Evaluation Kit Board. This is visual evidence that the eNVM has been programmed.



```
22 IS CORE CONFIGURED
Enter 1,2,3,4 .. 22, accordingly
Performing IAP actions...
Checking data CRC...
Expected CRC=CC4B
Calculating actual CRC...
ActID = 5A131CF ExpID = 5A131CF
Device Rev = 3
Initializing Target Device...
Setting BSR Configurations...
Loading BSR...
FSN: 1 5 D 34 98 78

Reading Security...
Checking for Calibration data...
Programming NVM block 0...
Verifying NVM block 0...
Exit code = 0
```

**Figure 28 • Menu After Entering 16 (Program\_NVM)**

## Conclusion

This application note described IAP and DirectC and demonstrated the capability of a SmartFusion IAP interface in programming the FPGA fabric and eNVM. Three different solutions were presented for programming the FPGA fabric and eNVM on the SmartFusion Evaluation Kit Board and the SmartFusion Development Kit Board:

- "Solution 1: Programming FPGA Fabric/eNVM Directly from the Host PC Using UART"
- "Solution 2: Programming FPGA Fabric Directly From the Host PC Using Ethernet"
- "Solution 3: Programming FPGA Fabric/eNVM From the On-Board SPI Flash"

## Appendix A – Design Files

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

[www.microsemi.com/soc/download/rsc/?f=A2F\\_AC362\\_DF](http://www.microsemi.com/soc/download/rsc/?f=A2F_AC362_DF)

The design file consists of Libero SoC projects and SoftConsole software projects.

Refer to the ReadMe.docx file for each solution included in the design file for directory structure and description.

Superseded

## List of Changes

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

Revision*	Changes	Page
Revision 8 (January 2013)	Added "Board Settings" section under "Solution 1: Programming FPGA Fabric/eNVM Directly from the Host PC Using UART" section (SAR 43469).	10
	Added "Board Settings" section under "Solution 3: Programming FPGA Fabric/eNVM From the On-Board SPI Flash" section (SAR 43469).	20
Revision 7 (May 2012)	Modified 'On the Host PC' under "Solution 1: Programming FPGA Fabric/eNVM Directly from the Host PC Using UART" section (SAR 38378).	7
	Replaced Figure 5 and Figure 6 (SAR 38378).	11
	Replaced Figure 7 (SAR 38378).	12
	Modified 'On the Host PC' under "Solution 2: Programming FPGA Fabric Directly From the Host PC Using Ethernet" section (SAR 38378).	12
	Replaced Figure 12 (SAR 38378).	16
	Replaced Figure 14 (SAR 38378).	17
	Modified "Running the Design" under "Solution 3: Programming FPGA Fabric/eNVM From the On-Board SPI Flash" section (SAR 38378).	20
	Replaced Figure 17 and Figure 18	21
Revision 6 (April 2012)	Added "Known Issues" section (SAR 37493).	5
Revision 5 (February 2012)	Modified the "Introduction" section (SAR 36681).	1
	Modified "DirectC Main Entry Function Call" section (SAR 36681).	3
	Modified "Requirements and Recommendations" section (SAR 36681).	5
	Removed "zip" extension in the Design files link (SAR 36763).	28
Revision 4 (January 2012)	Modified the "Solution 1: Programming FPGA Fabric/eNVM Directly from the Host PC Using UART" section (SAR 36009).	10
	Modified the "Solution 2: Programming FPGA Fabric Directly From the Host PC Using Ethernet" section (SAR 36009).	12
	Modified the "Solution 3: Programming FPGA Fabric/eNVM From the On-Board SPI Flash" section (SAR 36009).	17
Revision 3 (July 2011)	Added section "IAP and DirectC Overview" (SAR 32071).	2
	Added section "IAP With Overall System Design" (SAR 32071).	4
	Added section "Design Overview" (SAR 32071).	6
Revision 2 (May 2011)	Added Solution 1: Programming FPGA Fabric/eNVM Directly from the Host PC Using UART section on page 3 and Hardware Implementation section on page 4 (SAR - 32071).	7, 4
	Added FPGA Fabric and eNVM sub-sections under Design Example on page 4 (SAR - 32071).	4

**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.



Revision*	Changes	Page
	Added Macro Settings section on page 18 (SAR - 32071).	21
Revision 1 (March 2011)	Modified the number '6' to '16' in Figure 5 (SAR 31182).	4

*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.

Superseded

Superseded



**Microsemi**

**Microsemi Corporate Headquarters**  
One Enterprise, Aliso Viejo CA 92656 USA  
Within the USA: +1 (949) 380-6100  
Sales: +1 (949) 380-6136  
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at [www.microsemi.com](http://www.microsemi.com).

© 2013 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.