

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

Table of Contents

Introduction	1
In-Application Programming (IAP) Overview	2
IAP Programming	2
Design Example	3
Solution 1: Programming FPGA Fabric/eNVM Directly from the Host PC Using UART	3
Solution 2: Programming FPGA Fabric Directly from the Host PC Using Ethernet	6
Solution 3: Programming FPGA Fabric/eNVM from the On-Board SPI Flash	7
Hardware Implementation	10
Software Implementation	11
Running the Design	12
Conclusion	26
Appendix A – Design Files	27
List of Changes	27

Introduction

SmartFusion™ intelligent mixed signal FPGAs contain a hard embedded microcontroller subsystem (MSS), programmable analog circuitry, and FPGA fabric consisting of logic tiles, SRAM, and PLLs. The MSS consists of a 100 MHz ARM® Cortex™-M3 processor, peripheral DMA (PDMA), embedded nonvolatile memory (eNVM), embedded SRAM (eSRAM), embedded flashROM (eFROM), external memory controller (EMC), Watchdog Timer, I²C, SPI, 10/100 Ethernet Controller, real-time counter (RTC), GPIO block, fabric interface controller (FIC), in-application programming (IAP) and System Registers. The programmable analog block contains analog computing engine (ACE) and analog front-end (AFE) consisting of ADCs, DACs, active bipolar prescalers (ABPS), Comparators, current monitors and temperature monitors.

The IAP block is the portion of the MSS that acts as an APB peripheral for the Cortex-M3 processor. IAP provides the ability to program the FPGA fabric and eNVM for upgrading the firmware/software image when the system is installed in the end products.

The intent of this application note is to describe IAP and DirectC and demonstrate the capability of SmartFusion FPGAs to program the FPGA fabric and eNVM using the IAP interface. 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 SmartFusion design flow is assumed from the user.

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

In-Application Programming (IAP) Overview

The IAP block is part of the MSS and acts as an APB peripheral for the Cortex-M3 processor. The hard 100 MHz 32-bit ARM Cortex-M3 processor in a SmartFusion device can be used to execute the DirectC code in conjunction with the IAP controller to update the FPGA fabric and embedded nonvolatile memory (eNVM).

IAP programming can be initiated by the user through a user interrupt, or within the Cortex-M3 application, or from the FPGA fabric/eNVM through an interrupt to the Cortex-M3 processor as defined at design time, depending on the user application. The Cortex-M3 processor, upon the request for IAP to program the fabric/eNVM from any of the designated communication peripherals, will check the JTAG status of the programming controller. This is indicated by a bit in the IAP_STATUS register. If it is in reset, the IAP_ENABLE is asserted.

The processor executes the programming algorithm (erase, program, and verify) controlling the JTAG state machine in the IAP block. The programming operation is done one row at a time. When the programming for a row is done, IAP sends an interrupt to the processor for the next row until the fabric is programmed.

The processor, upon interrupt, will initiate the next instruction/command based on the programming algorithm. It is the responsibility of the processor to clear the interrupt source upon servicing an interrupt request. This is done by writing to any register in the IAP.

In this design example the IAP source for programming the FPGA fabric or eNVM can only be from one of the following communication peripherals (which are connected to dedicated pins on chip):

- UART
- SPI
- MAC

The data bitstream is transferred through any of the communication peripherals (UART, SPI, or MAC) by the Cortex-M3 processor in chunks of bits required to program a single row. The Cortex-M3 processor shifts the bitstream in chunks of 128 bits into IAP and commands the IAP block through the programming algorithm. The FPGA array can be programmed one row at a time.

Programming error conditions are read out of the programming controller registers portion of the programming algorithm and interpreted in the firmware.

IAP Programming

IAP is intended only for the on-chip Cortex-M3 programmer to program on-chip flash targets (FPGA array, eNVM).

In summary, there are three main parts involved in programming the FPGA fabric or eNVM using IAP:

- The application software DirectC that is running on the Cortex-M3 processor initiates IAP and handles all external communications.
- The IAP software/driver that processes the data and applies the appropriate algorithm to program the device.
- The IAP hardware block that drives the programming controller to program the FPGA Fabric/eNVM.

Figure 1 shows the hardware and software partitioning of IAP.

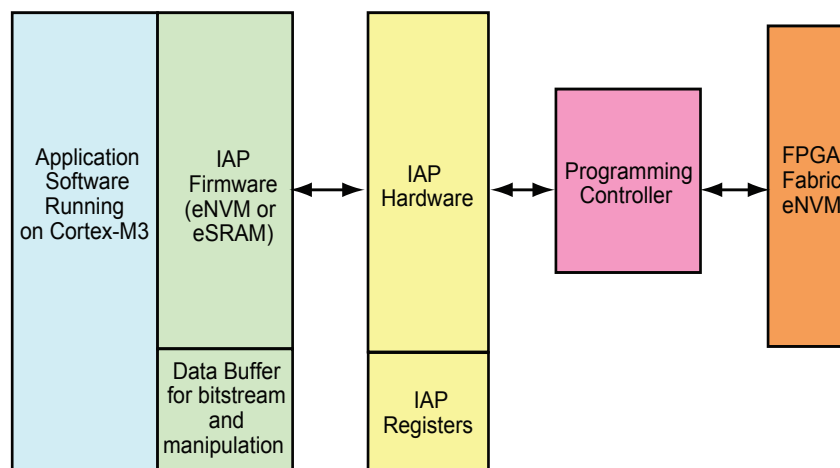


Figure 1 • Hardware and Software Partitioning of IAP

Design Example

The SmartFusion FPGA fabric or eNVM can be programmed using the IAP interface on the SmartFusion Development Kit and SmartFusion Evaluation Kit. 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"

Requirements and Recommendations

- Stable system power is needed during IAP programming. When you are designing a system, make sure that you have taken system level design considerations to avoid IAP program failures due to power interruptions. If IAP program fails due to power interruptions, you need to restart the IAP programming.
- You need to disable the Watchdog timer in software.
- Error detection and correction mechanism needs to be implemented to ensure programming data integrity.
- Application has to take care of failsafe mechanism for IAP programming directly from host PC using UART or Ethernet.

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

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 (SmartFusion Development Kit or SmartFusion Evaluation Kit).
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 until IAP completes verification.

On the Target

1. Initialize the UART on the target (SmartFusion Development Kit or SmartFusion Evaluation Kit) 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 and 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 IAP completes programming FPGA fabric or eNVM with the programming file (*.dat) from the host PC. Once IAP completes programming FPGA fabric or eNVM, Step 4 and Step 5 continue until IAP completes verification.

Figure 2 illustrates programming the FPGA fabric or eNVM directly from the host PC.

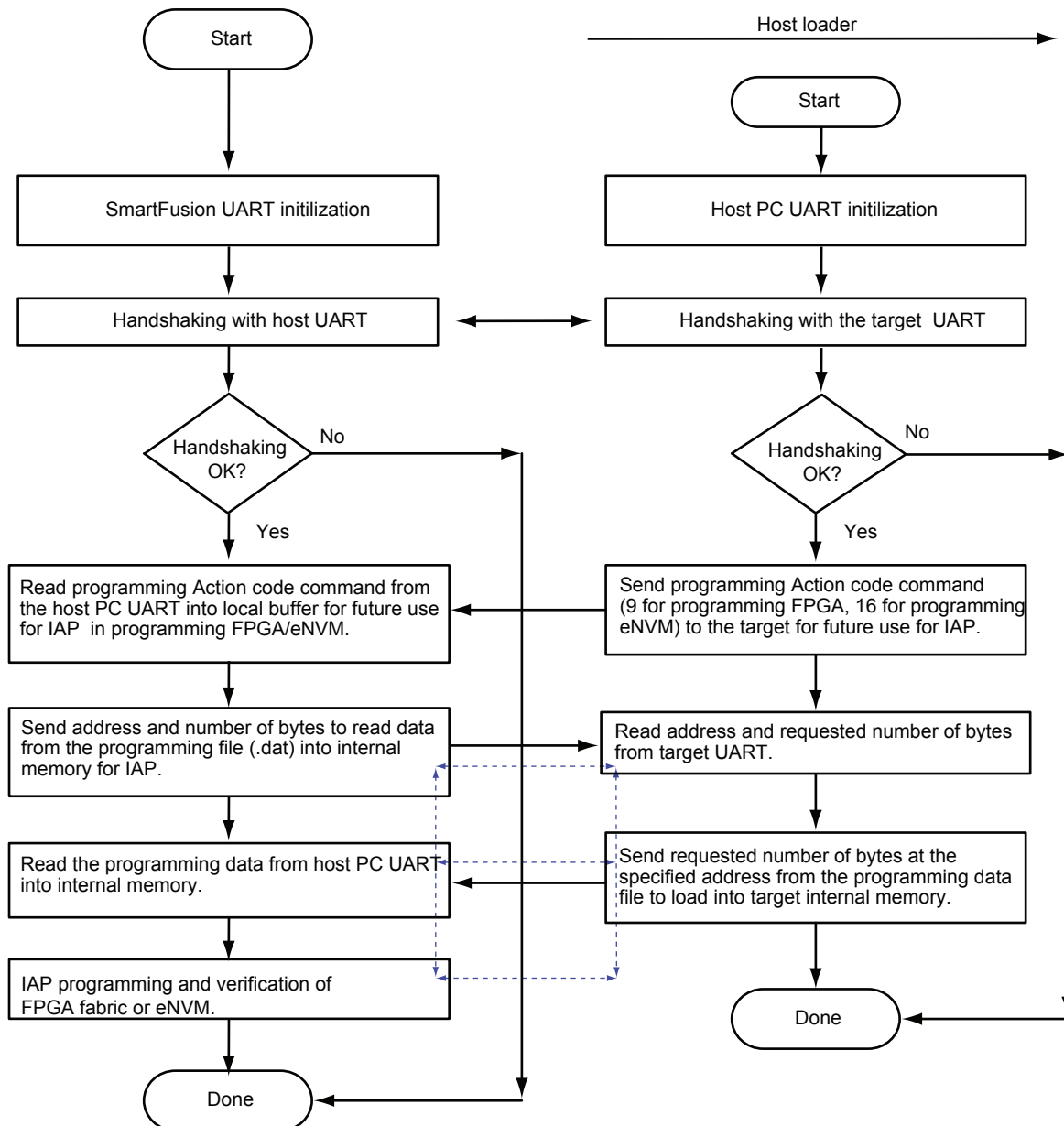


Figure 2 • Programming FPGA Directly from the Host PC Using UART

The SoC Products Group Libero® Integrated Design Environment (IDE) project and SoftConsole project are provided in the design files attached to this design example (refer to ["Appendix A – Design Files" on page 27](#)). Refer to the ["Running the Design" section on page 12](#) to understand how to load the FPGA fabric or eNVM programming file (*.dat) and perform the programming.

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

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 side, 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 FPGA fabric, Step 2 continues until IAP completes verification.

On the Target

1. On the target side, initialize the MAC 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 Ethernet interface on the target board, receiving whatever has been sent by the host PC over the network.
4. Step 2 and Step 3 continue till IAP completes programming FPGA fabric with the programming file (*.dat) from the host PC. Once IAP completes programming FPGA fabric, Step 2 and Step 3 continue until IAP completes verification.

Figure 3 illustrates programming the FPGA fabric directly from the host PC using Ethernet.

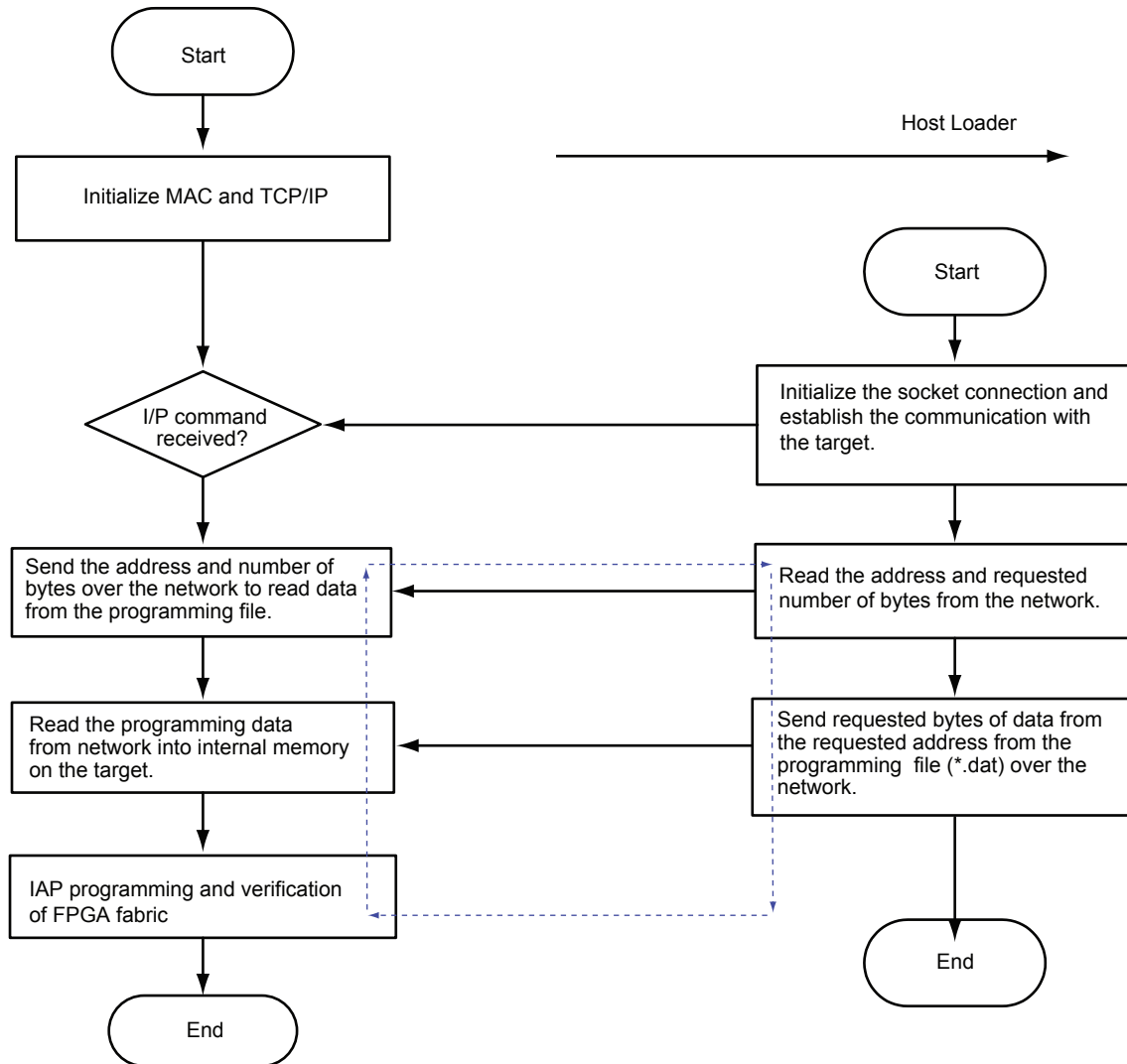


Figure 3 • Programming FPGA Directly from the Host PC Using Ethernet

The Libero IDE project and SoftConsole project are provided in the design files attached to this design example ("Appendix A – Design Files" on page 27). Refer to the "Running the Design" section on page 12 for information on loading the FPGA fabric programming file (*.dat) and programming the FPGA fabric.

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

This solution uses SPI flash to load the whole programming file (*.dat) into the SPI flash. The IAP interface 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 parts. Part one briefly introduces two methods for loading the FPGA fabric or eNVM programming file into the on-board SPI flash file system of the SmartFusion Development Kit or SmartFusion Evaluation Kit, or into on-board SPI flash with no file system.

Part two briefly describes programming the FPGA fabric or eNVM using the IAP interface from the on-board SPI flash file system or SPI flash with no file system.

On the Host PC

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 or SmartFusion Evaluation Kit).
3. Send the programming file to be loaded into SPI flash on the target until the entire file has been transferred to SPI flash.

On the Target

1. Initialize the UART on the target (SmartFusion Development Kit or SmartFusion Evaluation Kit) with a baud rate that matches the UART baud rate with the host tool (Memory Loader) that is running on host PC and 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 4 shows loading the FPGA fabric or eNVM programming file into SPI flash through UART.

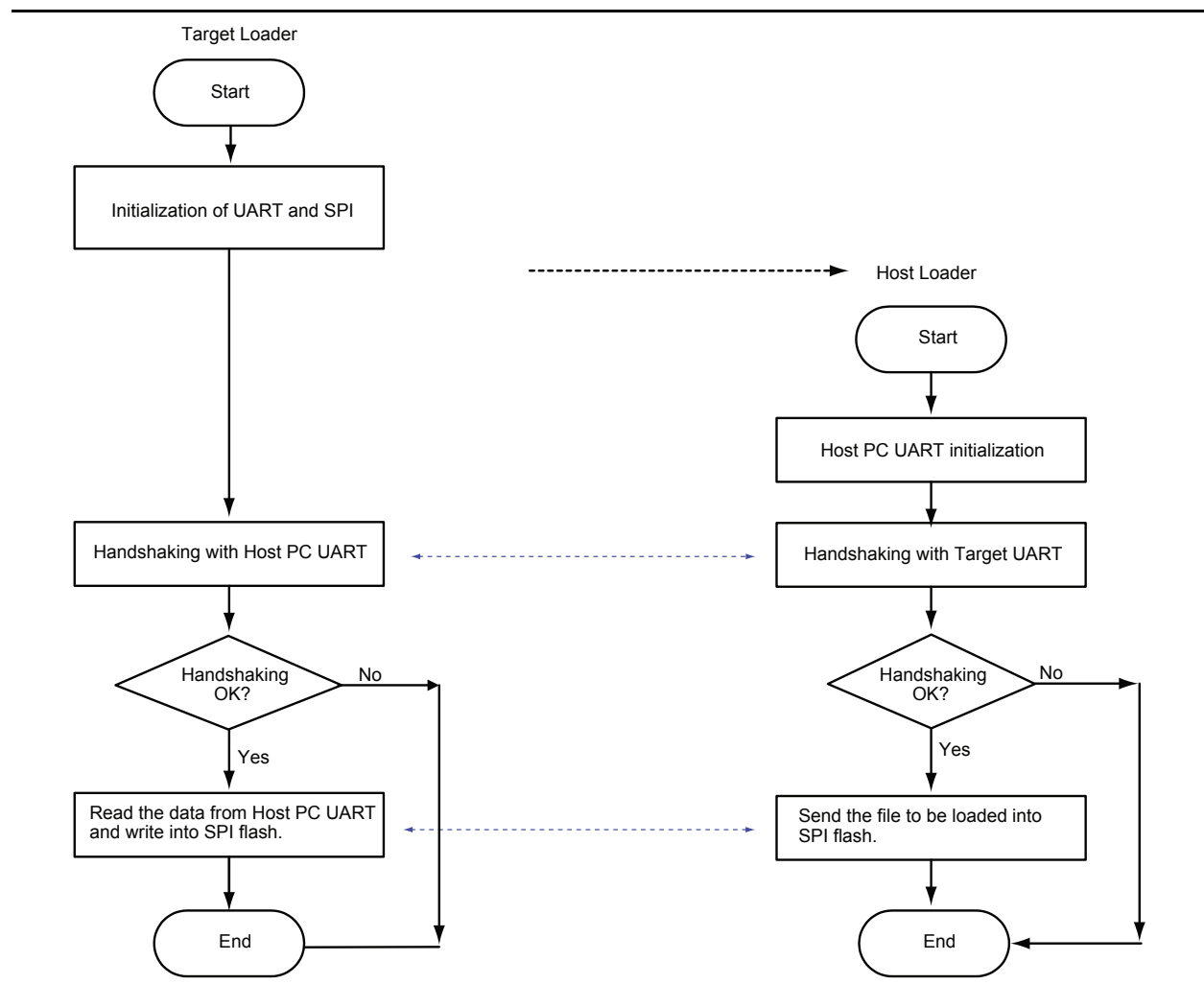


Figure 4 • SPI Flash Programming Through UART

Using TFTP

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

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 will be transferred over the network to the SPI flash on the target.

On the Target

1. Initialize the MAC, SPI, lwIP, and file system on the target board (SmartFusion Development Kit or SmartFusion Evaluation Kit).
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 5 shows loading the FPGA fabric programming file into SPI flash through the TFTP protocol.

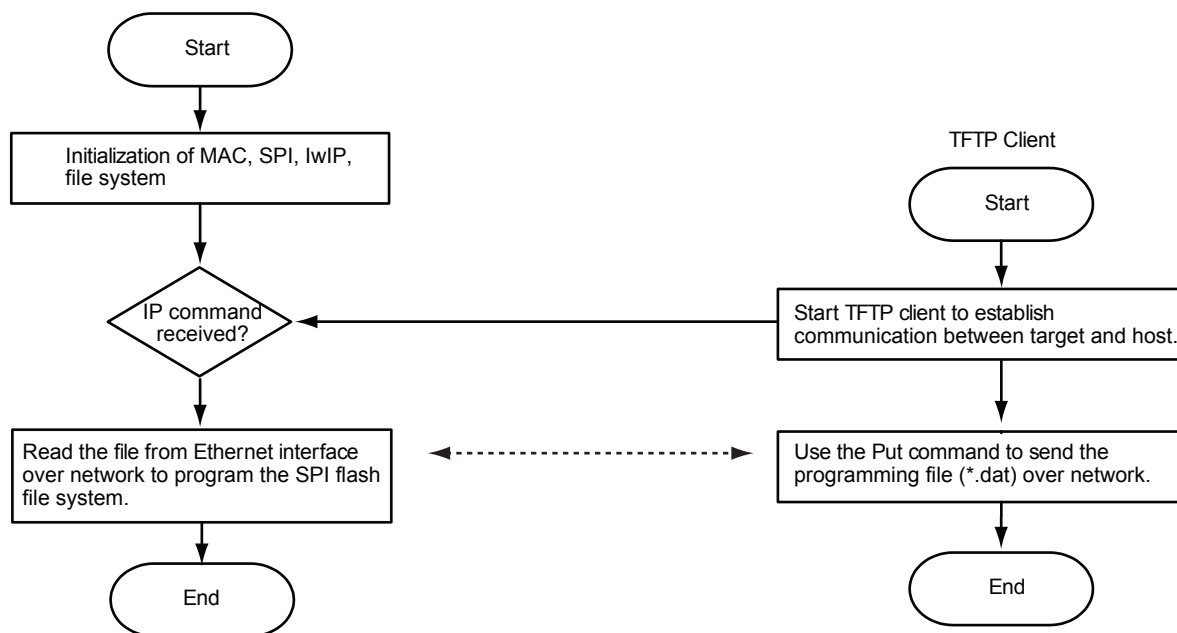


Figure 5 • SPI Flash Programming Through TFTP Protocol

Part Two

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

The Libero IDE project and SoftConsole project are provided in the design files attached to this design example ("Appendix A – Design Files" on page 27). Refer to the "Running the Design" section on page 12 for information on loading the programming file (*.dat) and programming the FPGA fabric or eNVM.

Hardware Implementation

For running the IAP application on the Cortex-M3 processor, configure the hardware with the required MSS peripherals (UART, SPI, or Ethernet) that need to be enabled.

The provided hardware can be used for any of the methods explained in this document.

DirectC (*.dat) File Generation

For FPGA Fabric Programming

- Create a Libero IDE project with 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 IDE 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.

Software Implementation

To perform IAP programming for the FPGA fabric or eNVM, you must integrate the IAP driver and the DirectC files into the application code. You can download the IAP driver using [Firmware Catalog](#) from the Microsemi SoC Products Group [website](#).

A paging mechanism has been used for IAP programming to access the data in eSRAM from the on-board storage or directly from the host PC using UART/Ethernet.

Paging is a method of copying a certain range of data from storage (SPI Flash or directly from host PC) containing the *.dat file and pasting it into a limited size internal memory buffer that DirectC can access. The DirectC code used has a maximum Page_buffer_size of 8 kbytes.

dp_get_page_data is the only function in the file that must interface with the communication peripheral of the image data file. Since the requested data blocks may not be contiguous, it must have random access to the data blocks. Its purpose is to fill the page buffer with valid data every time a new data block is needed.

The *dp_get_page_data* function takes two arguments:

- *image_requested_address*, which contains the relative address location of the first needed byte within the data block of the image file.
- *return_bytes* variable, which should be updated with the number of valid bytes available.

The solutions that use the UART method to load the programming file directly from the host PC into internal memory (eSRAM) or to a storage area, SPI flash, require handshake between the target and the host PC. Typically that has to be implemented in the main.c file before the IAP requires programming data to program the FPGA fabric or eNVM.

The solutions that use the Ethernet method use MAC 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 area file system, SPI flash, that is resident on the target board.

To use IAP programming, ENABLE_IAP_SUPPORT must be defined in DirectC/dpuser.h. You must set the *hardware_interface* variable, defined in DirectC/dpuser.c, to IAP_SEL.

The appropriate logic has been implemented in the file DirectC/dpcom.c (i.e., the function *dp_get_page_data*) for all the solutions so that communication with the peripherals can take place. The image file is loaded directly from the host PC or from the storage area SPI flash into internal memory eSRAM (page buffer). 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 provided can be used for any of the methods explained in this document.

Macro Settings

The following macros are valid only for Solution 3 and must be enabled/disabled in the SPI flash API file (spi_flash.h) to enable the appropriate board. The design example works with both the SmartFusion Development Kit and SmartFusion Evaluation Kit.

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

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

The following macro must be commented in the DirectC/dpuser.h file for Solution 1 as the UART port used for receiving the programming file from the host PC:

```
#define ENABLE_DEBUG
```

Running the Design

The design files provided for solutions 1 and 3 are compatible with the SmartFusion Development Kit and the SmartFusion Evaluation Kit. Solution 2 is compatible only with the SmartFusion Development Kit because there is no 50 MHz external clock source available on the SmartFusion Evaluation Kit. You can use these files to evaluate the IAP feature in SmartFusion.

Program the SmartFusion Development Kit or SmartFusion Evaluation Kit with the appropriate hardware design files ("[Appendix A – Design Files](#)" on page 27) using FlashPro and then power cycle the board.

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

On the Target

To program the FPGA fabric, invoke the SoftConsole IDE and browse for the SoftConsole project folder (*Programming_FPGAFabric\SW\UART_HostPC\IAP_UART_From_HostPC_ws*). Refer to "[Appendix A – Design Files](#)" on page 27 for more information. Select the **IAP_SC_prj** project and launch the debugger.

To program eNVM, invoke the SoftConsole IDE and browse for the SoftConsole project folder (*Programming_eNVM\SW\UART_HostPC\IAP_UART_From_HostPC_ws*). Refer to "[Appendix A – Design Files](#)" on page 27 for more information. Select the **IAP_SC_prj** project and launch the debugger.

On the Host PC

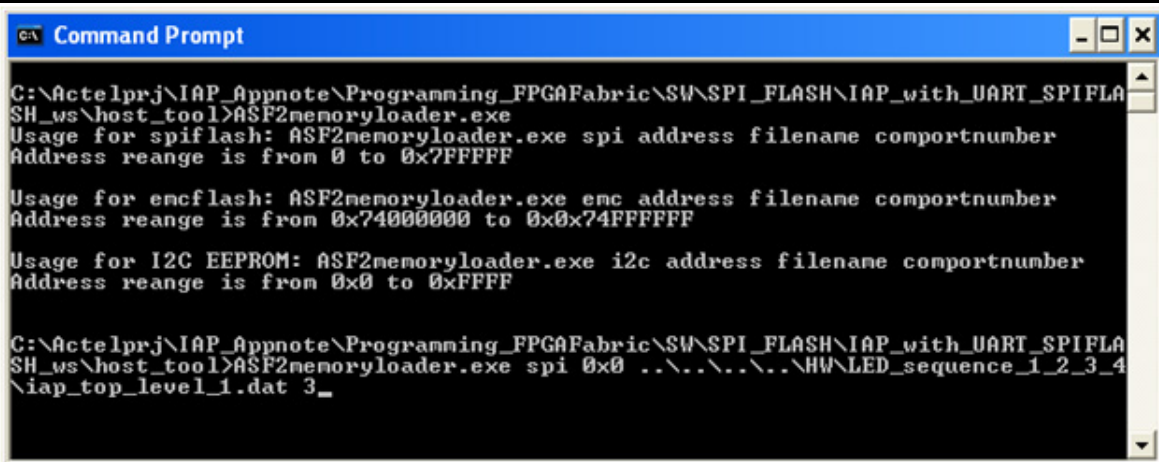
After running the debugger in SoftConsole, launch Memory Loader at the command prompt.

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

Run the Memory Loader tool at the command prompt. Before running the Memory Loader tool, make sure that COM port is not used by any other application, such as HyperTerminal or PuTTY.

Open a command prompt window in the host PC and change to the directory where the host tool is located (refer to "[Appendix A – Design Files](#)" on page 27). Type **ASF2memoryloader.exe**. Help on how to run the host loader is printed.

Figure 6 shows the Memory Loader (ASF2memoryloader) help for programming the FPGA fabric.



```
GA Command Prompt
C:\Actelprj\IAP_Appnote\Programming_FPGAFabric\SW\SPI_FLASH\IAP_with_UART_SPIFLA
SH_ws\host_tool>ASF2memoryloader.exe
Usage for spiflash: ASF2memoryloader.exe spi address filename comportnumber
Address reange is from 0 to 0x7FFFFFFF

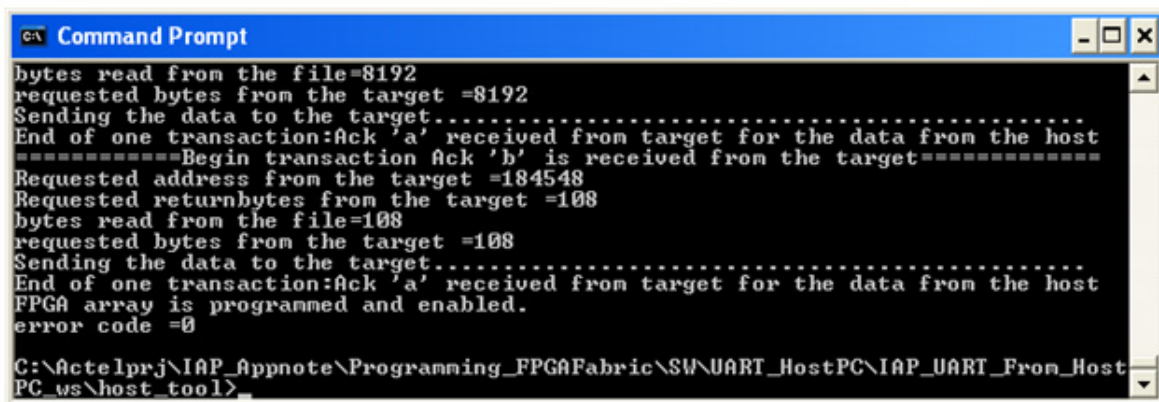
Usage for encflash: ASF2memoryloader.exe emc address filename comportnumber
Address reange is from 0x74000000 to 0x0x74FFFFFF

Usage for I2C EEPROM: ASF2memoryloader.exe i2c address filename comportnumber
Address reange is from 0x0 to 0xFFFF

C:\Actelprj\IAP_Appnote\Programming_FPGAFabric\SW\SPI_FLASH\IAP_with_UART_SPIFLA
SH_ws\host_tool>ASF2memoryloader.exe spi 0x0 ..\..\..\HW\LED_sequence_1_2_3_4
\iap_top_level_1.dat 3_
```

Figure 6 • Launching Memory Loader On Host PC

Figure 7 shows the debug messages while programming the FPGA fabric.



```

C:\ Command Prompt
bytes read from the file=8192
requested bytes from the target =8192
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 =184548
Requested returnbytes from the target =108
bytes read from the file=108
requested bytes from the target =108
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

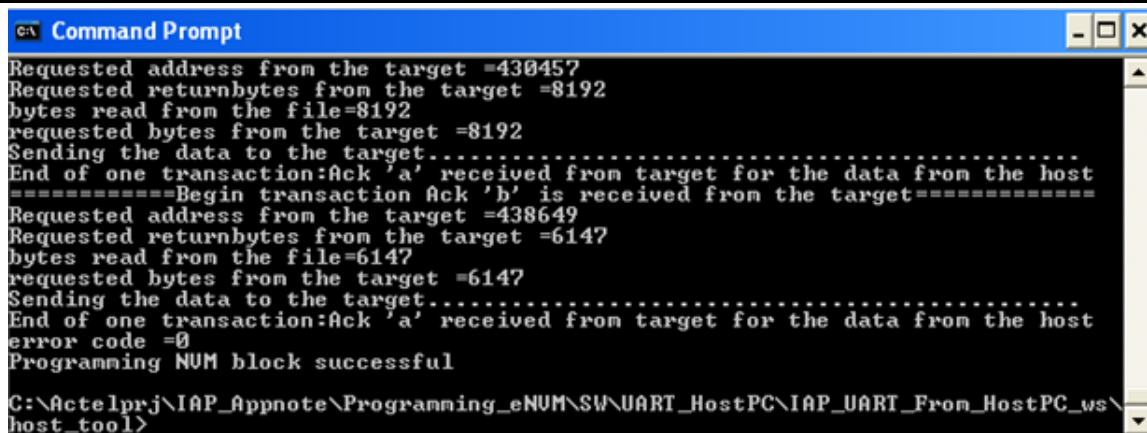
C:\Actelprj\IAP_Appnote\Programming_FPGAFabric\SW\UART_HostPC\IAP_UART_From_HostPC_ws\host_tool>
  
```

Figure 7 • Debug Messages after Running the Memory Loader on Host PC

You can observe the blinking LED sequence on the SmartFusion Development Kit or SmartFusion Evaluation Kit for the *.dat file that was launched on the command prompt. There should be a change in the running LED sequence. This is visual evidence that the new design has been programmed into the FPGA fabric.

In the case of programming eNVM, browse to the directory where the host tool is located (*Programming_eNVM\SW\UART_HostPC\IAP_UART_From_HostPC_ws*) and launch the memory loader (ASF2memoryloader) at the command prompt.

Figure 8 shows the debug messages while programming eNVM.



```

C:\ Command Prompt
Requested address from the target =430457
Requested returnbytes from the target =8192
bytes read from the file=8192
requested bytes from the target =8192
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 =438649
Requested returnbytes from the target =6147
bytes read from the file=6147
requested bytes from the target =6147
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:\Actelprj\IAP_Appnote\Programming_eNVM\SW\UART_HostPC\IAP_UART_From_HostPC_ws\host_tool>
  
```

Figure 8 • Debug Messages While Programming eNVM

In programming 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 board. This is visual evidence that the eNVM has been programmed.

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

This solution uses MAC and uIP TCP/IP stack to get the FPGA fabric programming file (*.dat) from the remote PC or local PC into internal memory (eSRAM) on the target, when requested by the target.

Socket application programming has been used on both the target and host PC. TCP/IP manages the requests in the form of packets between the target and host.

Board Setup

This solution requires an external 50 MHz clock source that needs to be fed to MAC_CLK and RMII PHY clock. This solution works only on the SmartFusion Development Kit. This solution cannot be tested on SmartFusion Evaluation Kit since there is no external 50 MHz clock source available.

The following method is one of the possible solutions for routing the external clock to RMII PHY clock (Pin Number 34) with the SmartFusion Development Kit.

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.

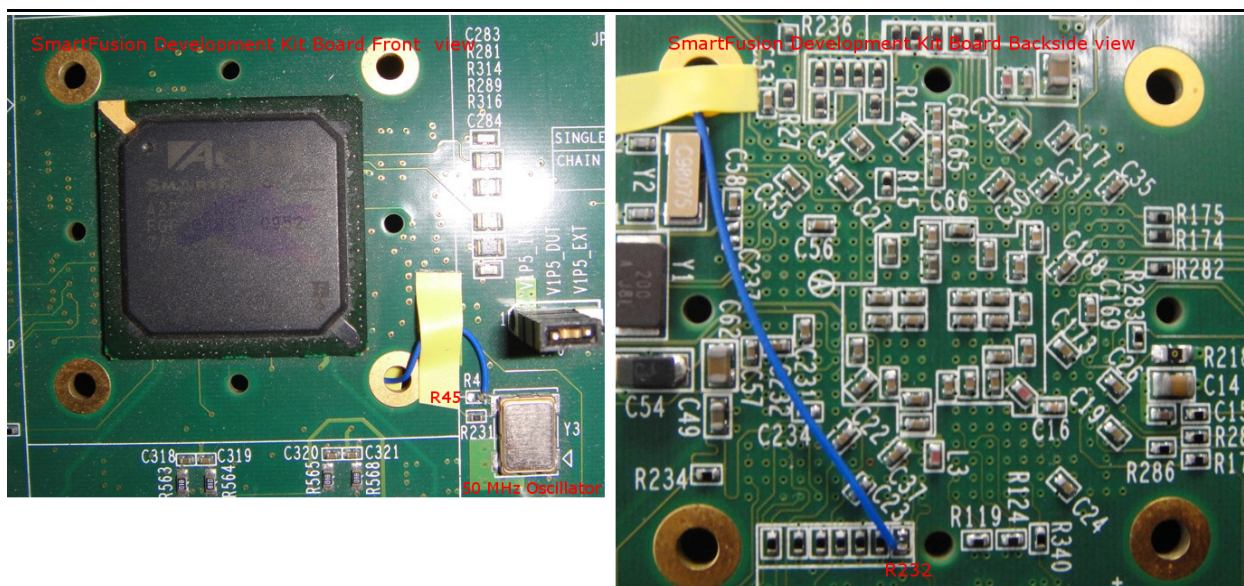


Figure 9 • Modified Board

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

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

\\IAP_Appnote\\Programming_FPGA\\Fabric\\HW\\HW_Ethernet_Host_PC_IAP\\SmartFusion_Ethernet_IAP_HW

Note: When you are designing a system, make sure that you are feeding an external 50MHz clock to MAC_CLK and RMII PHY. [Figure 10](#) illustrates the ethernet clocking scheme for a system with IAP programming.

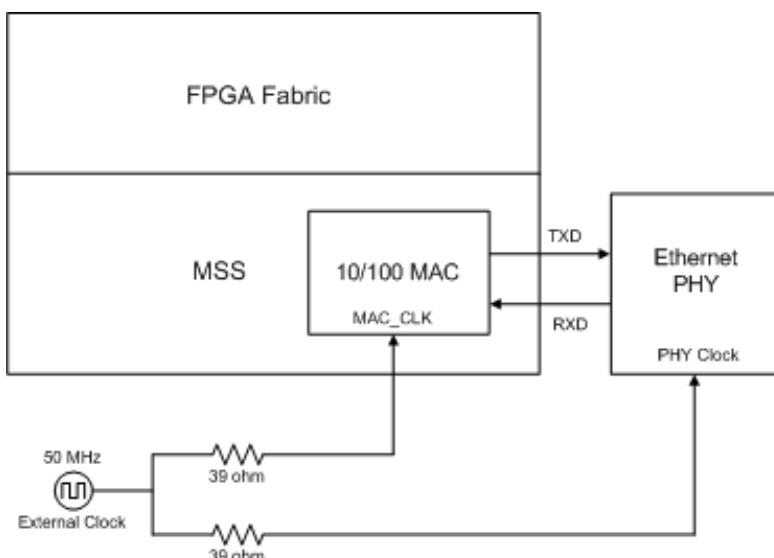


Figure 10 • Ethernet Clocking Scheme for a System with IAP

On the Target

To program the FPGA fabric, invoke the SoftConsole IDE and browse for the SoftConsole project folder (*Programming_FPGAFabric\SW\Ethernet_HostPC\IAP_Ethernet_From_HostPC_ws*). See ["Appendix A – Design Files"](#) on page 27 for more information. Select the project **Ethernet_From_HostPC_IAP_SC** 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.

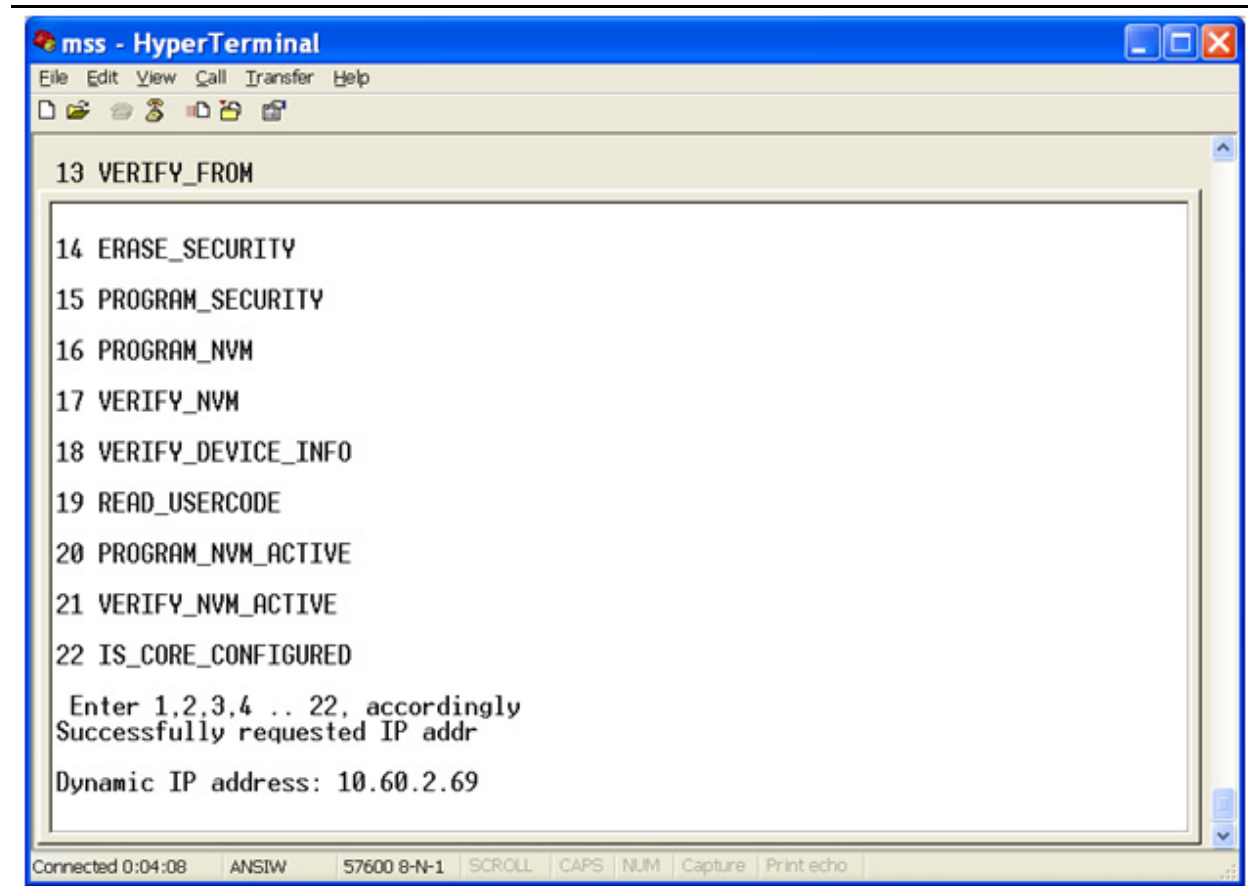


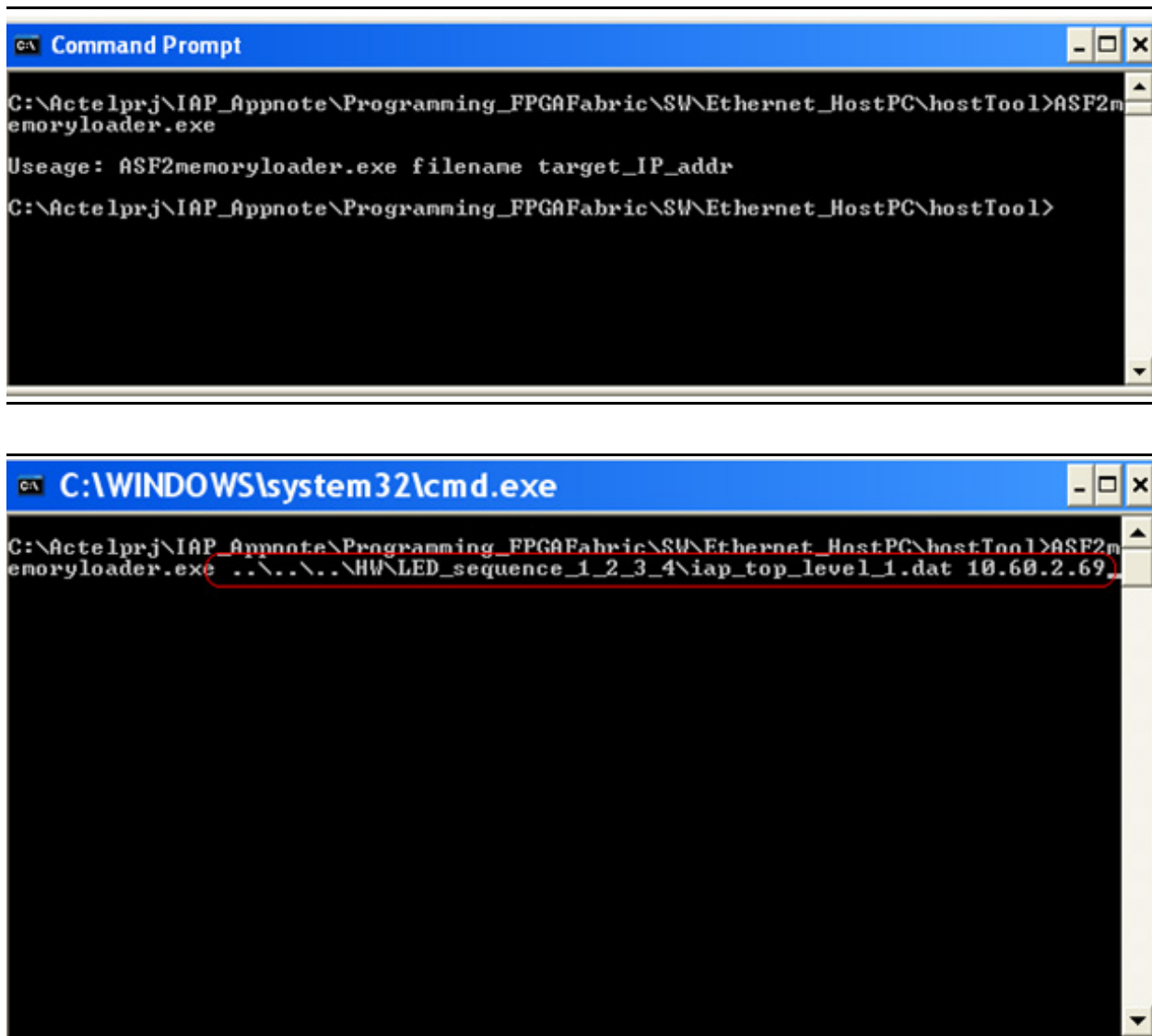
Figure 11 • HyperTerminal with the Dynamic IP Address and IAP Options

After running the debugger in SoftConsole, launch the Memory Loader with the required input parameters on the command prompt, as shown in [Figure 12](#).

Memory loader is an executable program (*.exe) provided in the host tool folder. It runs on the host machine 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 (refer to "[Appendix A – Design Files](#)" on page 27). Type **ASF2memoryloader.exe**. Help on how to run the Memory loader is printed.

Figure 12 shows the Memory Loader (ASF2memoryloader) help for programming the FPGA fabric.



The figure consists of two screenshots of a Windows Command Prompt window. The top screenshot shows the command prompt with the title 'Command Prompt'. The current directory is 'C:\Actelprj\IAP_Appnote\Programing_FPGAFabric\SW\Ethernet_HostPC\hostTool'. The command 'ASF2memoryloader.exe' has been entered, and the usage information is displayed: 'Useage: ASF2memoryloader.exe filename target_IP_addr'. The bottom screenshot shows the same command prompt with the title 'C:\WINDOWS\system32\cmd.exe'. The command 'ASF2memoryloader.exe ..\..\..\HW\LED_sequence_1_2_3_4\iap_top_level_1.dat 10.60.2.69' has been entered, and the first part of the command is highlighted with a red box.

```
Command Prompt
C:\Actelprj\IAP_Appnote\Programing_FPGAFabric\SW\Ethernet_HostPC\hostTool>ASF2memoryloader.exe
Useage: ASF2memoryloader.exe filename target_IP_addr
C:\Actelprj\IAP_Appnote\Programing_FPGAFabric\SW\Ethernet_HostPC\hostTool>

C:\WINDOWS\system32\cmd.exe
C:\Actelprj\IAP_Appnote\Programing_FPGAFabric\SW\Ethernet_HostPC\hostTool>ASF2memoryloader.exe ..\..\..\HW\LED_sequence_1_2_3_4\iap_top_level_1.dat 10.60.2.69
```

Figure 12 • Launching Memory Loader on Host PC

After launching the Memory Loader, choose option 09 in HyperTerminal to program FPGA fabric as shown in Figure 13.

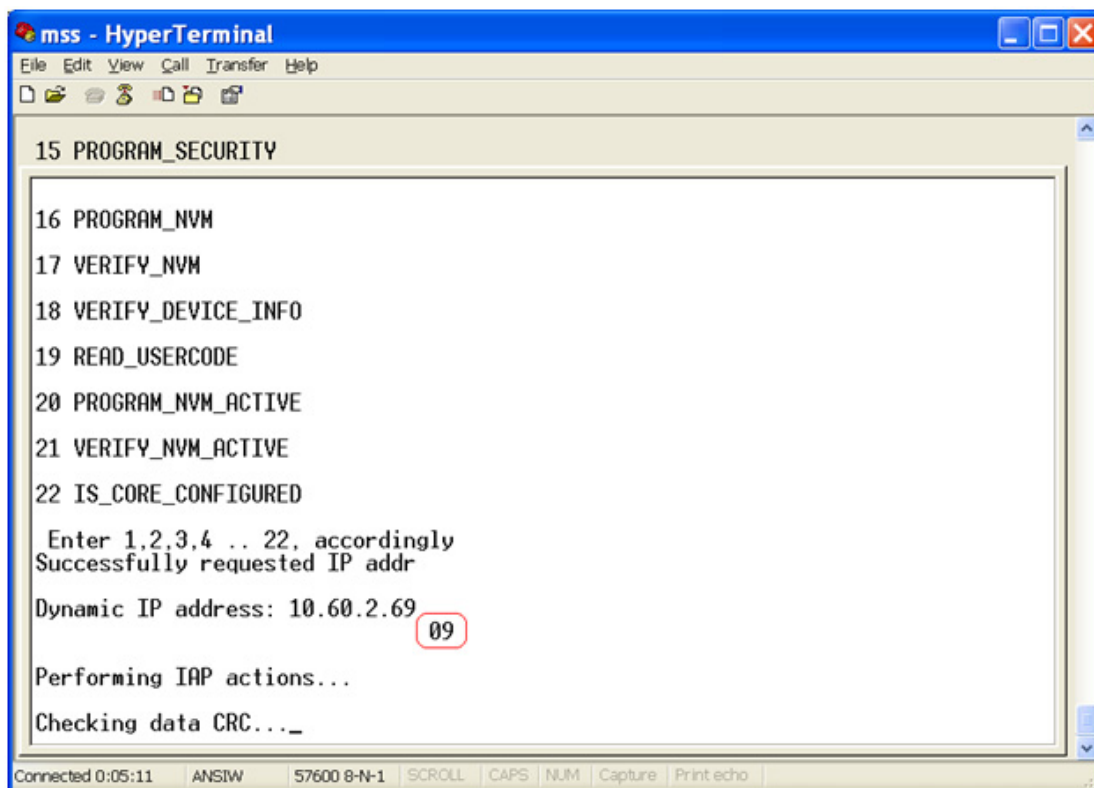


Figure 13 • HyperTerminal after Choosing Option 09 for Programming FPGA Fabric

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

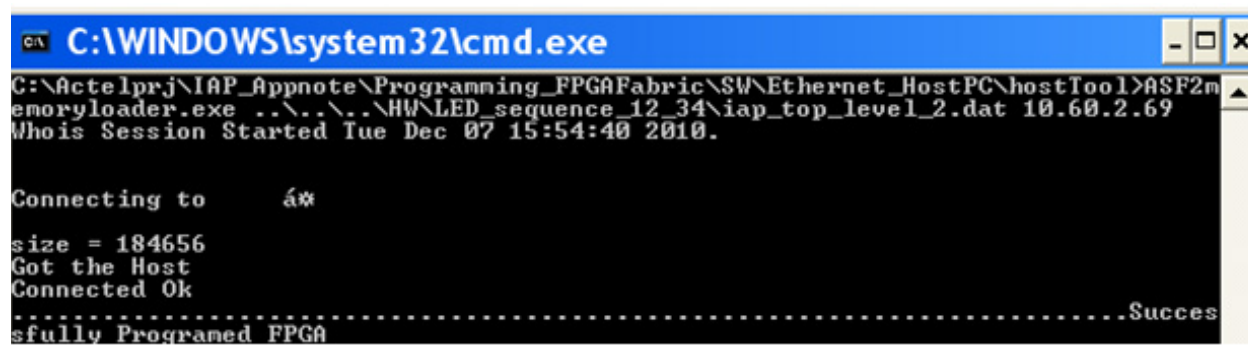


Figure 14 • Debug Messages while Programming the FPGA

You can observe the blinking LED sequence on the SmartFusion board for the *.dat file that was launched on the command prompt. There should be 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

This solution consists of two parts. The first part is loading the SPI flash with the FPGA fabric or eNVM programming file from the host PC. The second part is to program the FPGA fabric or eNVM with the programming file (*.dat) that is residing in on-board SPI flash.

Part 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 TFTP protocol.

Using the UART Interface

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

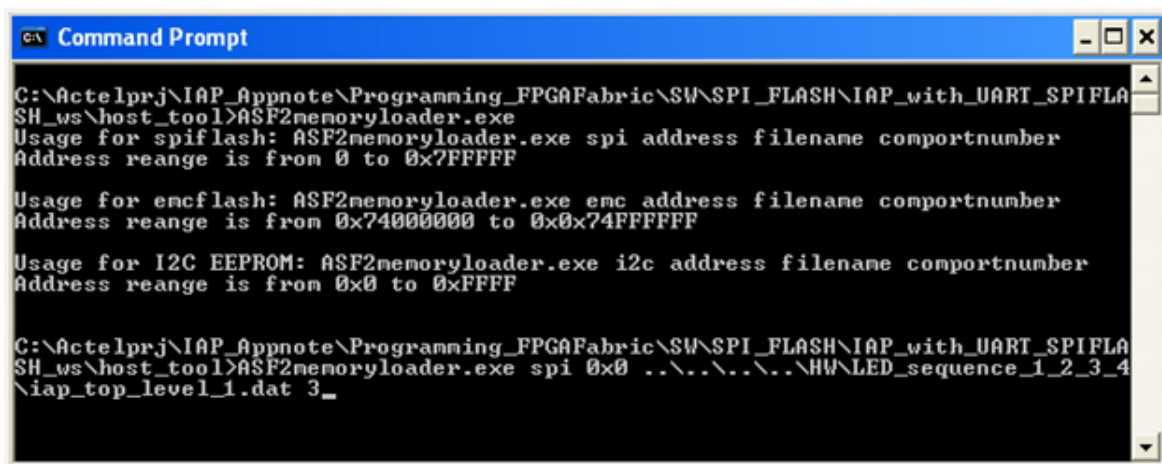
To program the FPGA fabric, invoke the SoftConsole IDE and browse for the SoftConsole project folder (*Programming_FPGAFabric\SW\SPI_FLASH\IAP_with_UART_SPIFLASH_ws*). Refer to ["Appendix A – Design Files" on page 27](#) for more information. Select the project **UART_SPIFlashLoad_SC_prj** and launch the debugger.

To program eNVM, invoke the SoftConsole IDE and browse for the SoftConsole project folder (*Programming_eNVM\SW\SPI_FLASH\IAP_with_UART_SPIFLASH_ws*). Refer to ["Appendix A – Design Files" on page 27](#) for more information. Select the project **UART_SPIFlashLoad_SC_prj** and launch the debugger.

After running the debugger in SoftConsole, launch the Memory Loader at the command prompt.

Open a command prompt window in the host PC and change to the directory where the host tool is located (see ["Appendix A – Design Files" on page 27](#)). Type **ASF2memoryloader.exe**. Help on how to run the Memory Loader is printed.

Figure 15 shows the Memory Loader (ASF2memoryloader) help for programming the FPGA fabric.



```

C:\Actelprj\IAP_Appnote\Programming_FPGAFabric\SW\SPI_FLASH\IAP_with_UART_SPIFLASH_ws\host_tool>ASF2memoryloader.exe
Usage for spiflash: ASF2memoryloader.exe spi address filename comportnumber
Address range is from 0 to 0x7FFFFFFF

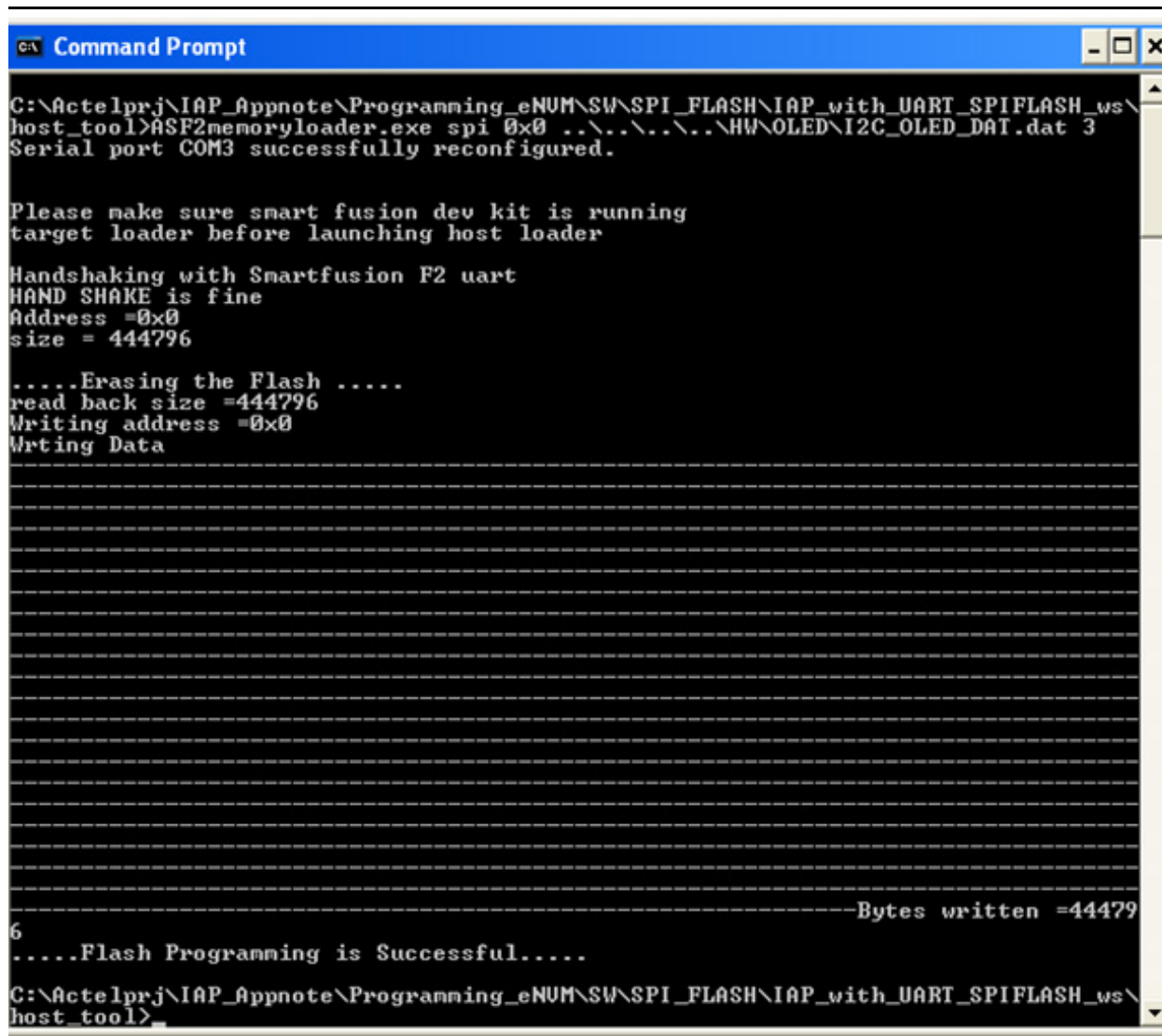
Usage for emcflash: ASF2memoryloader.exe emc address filename comportnumber
Address range is from 0x74000000 to 0x0x74FFFFFFF

Usage for I2C EEPROM: ASF2memoryloader.exe i2c address filename comportnumber
Address range is from 0x0 to 0xFFFF

C:\Actelprj\IAP_Appnote\Programming_FPGAFabric\SW\SPI_FLASH\IAP_with_UART_SPIFLASH_ws\host_tool>ASF2memoryloader.exe spi 0x0 ..\..\..\HW\LED_sequence_1_2_3_4\iap_top_level_1.dat 3_
  
```

Figure 15 • Launching Memory Loader on Host PC

Figure 16 shows the debug messages while programming SPI flash.



```
Command Prompt

C:\Actelprj\IAP_Appnote\Programming_eNVM\SW\SPI_FLASH\IAP_with_UART_SPIFLASH_ws\
host_tool>ASF2memoryloader.exe spi 0x0 ..\..\..\HW\OLED\I2C_OLED_DAT.dat 3
Serial port COM3 successfully reconfigured.

Please make sure smart fusion dev kit is running
target loader before launching host loader

Handshaking with Smartfusion F2 uart
HAND SHAKE is fine
Address =0x0
size = 444796

.....Erasing the Flash .....
read back size =444796
Writing address =0x0
Wrting Data

-----Bytes written =444796
6
.....Flash Programming is Successful.....

C:\Actelprj\IAP_Appnote\Programming_eNVM\SW\SPI_FLASH\IAP_with_UART_SPIFLASH_ws\
host_tool>
```

Figure 16 • 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\SW\SPI_FLASH\IAP_with_UART_SPIFLASH_ws\host_tool.`

Launch Memory Loader (ASF2memoryloader) at the command prompt.

Using TFTP

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

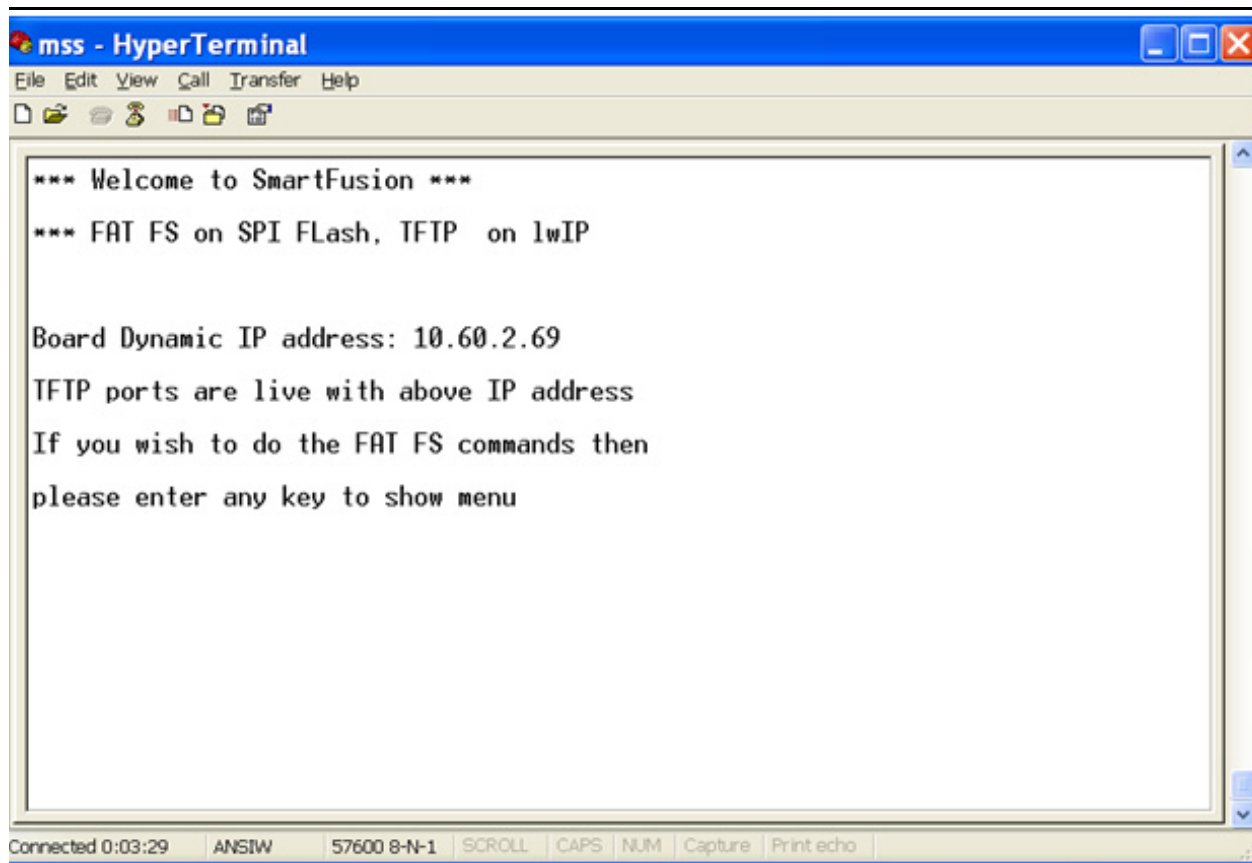
The TFTP server on the target, which includes MAC, lwIP, and file system on the storage area (SPI flash) allows file transfer capability to and from the storage area file system resident on the target board to the host PC.

The TFTP Server generates a dynamic IP 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 and browse for the SoftConsole project folder (`Programming_FPGAFabric\SW\SPI_FLASH\IAP_with_Ethernet_SPIFLASH_ws`).

Refer to "Appendix A – Design Files" on page 27 for more information. Select the project **Ethernet_SPIFlashLoad_SC_prj** and launch the debugger.

When you run the debugger in SoftConsole, the application prints the IP address on HyperTerminal, as shown in Figure 17.



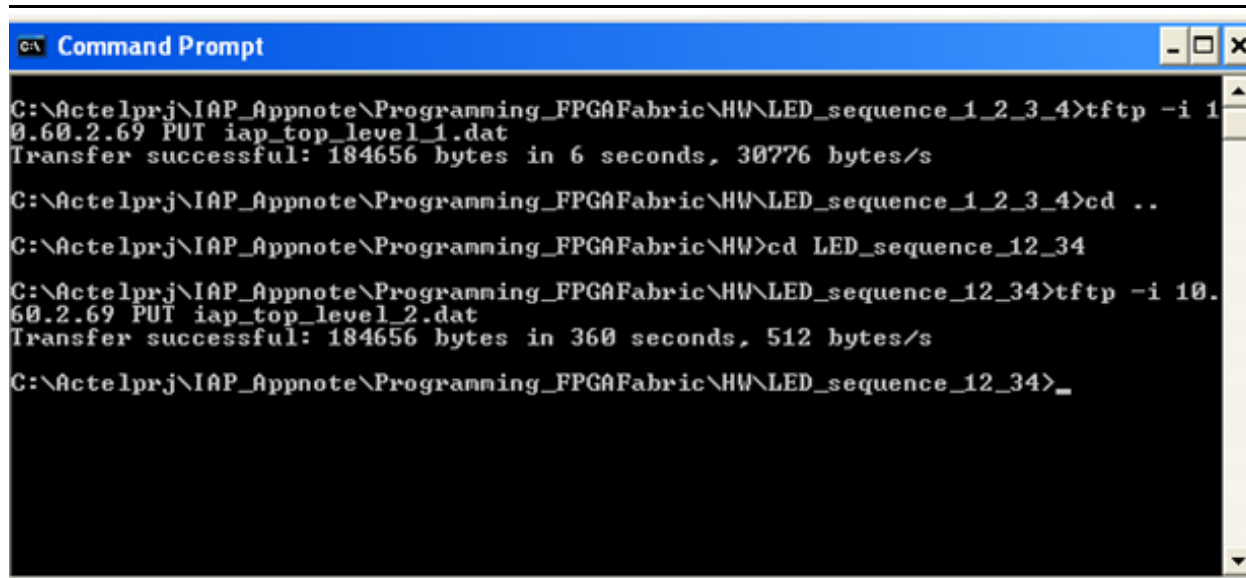
```
*** Welcome to SmartFusion ***
*** FAT FS on SPI Flash, TFTP on lwIP

Board Dynamic IP address: 10.60.2.69
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
```

Connected 0:03:29 ANSI 57600 8-N-1 SCROLL CAPS NUM Capture Print echo

Figure 17 • HyperTerminal with Dynamic IP, TFTP, and FAT FS Commands

After running the debugger in SoftConsole, launch the TFTP client at the command prompt. 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 18 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

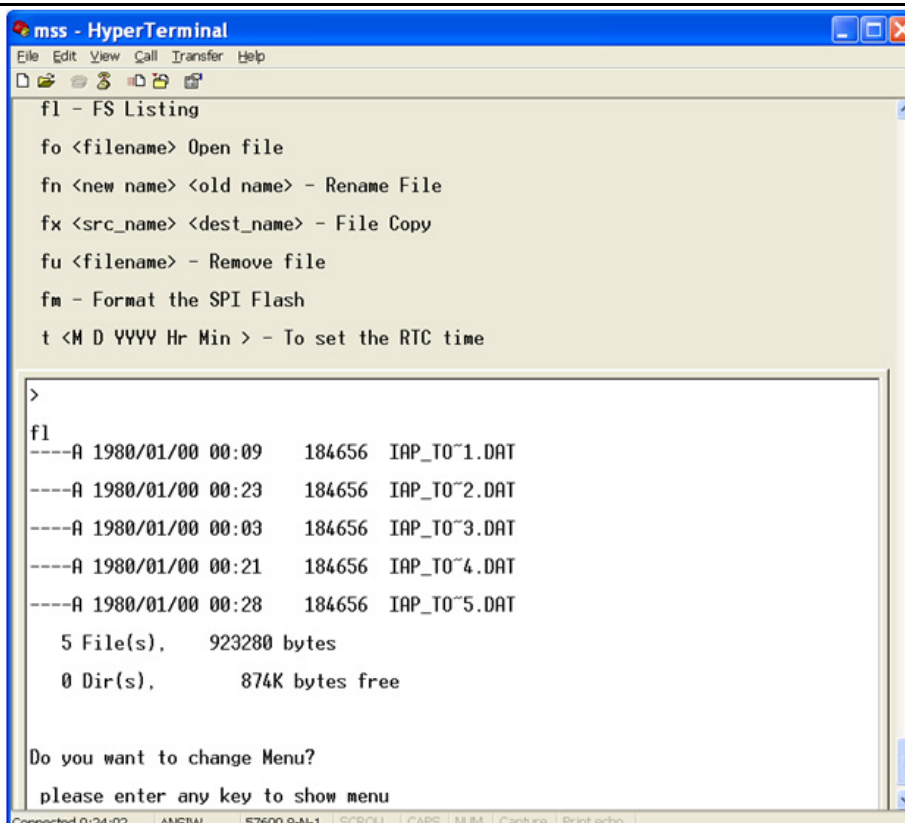
C:\Actelprj\IAP_Appnote\Programming_FPGA\Fabric\HW\LED_sequence_12_34>tftp -i 10.60.2.69 PUT iap_top_level_2.dat
Transfer successful: 184656 bytes in 360 seconds, 512 bytes/s

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

```

Figure 18 • Launching TFTP Client on Host PC

Type the command **fl** in HyperTerminal to see the list of stored files on SPI flash on the target board. Figure 19 shows the list of stored files on SPI flash.



```

mss - HyperTerminal
File Edit View Call Transfer Help

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 19 • 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 eNVM.

Part 2

This section covers demonstration of 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.

Stop the debugger in the SoftConsole project for the current project and close the current project, UART_SPIFlashLoad_SC_prj/Ethernet_SPIFlashLoad_SC_prj. Select the **IAP_SC_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 options on HyperTerminal. [Figure 20](#) shows HyperTerminal with IAP options.

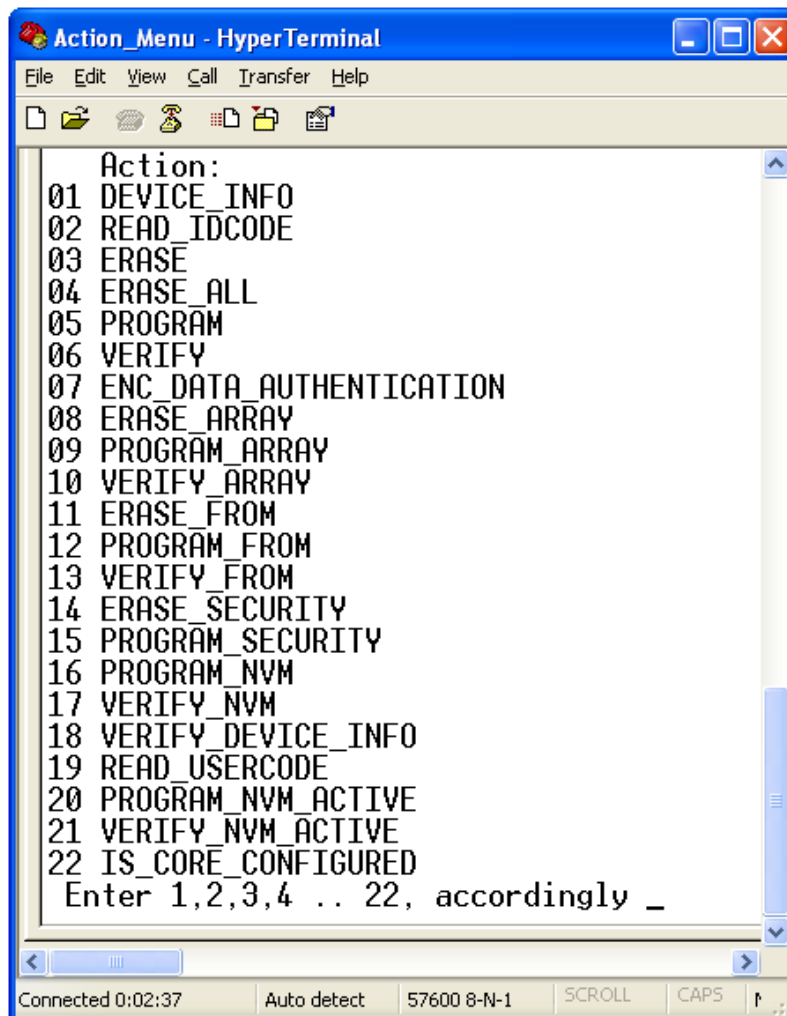
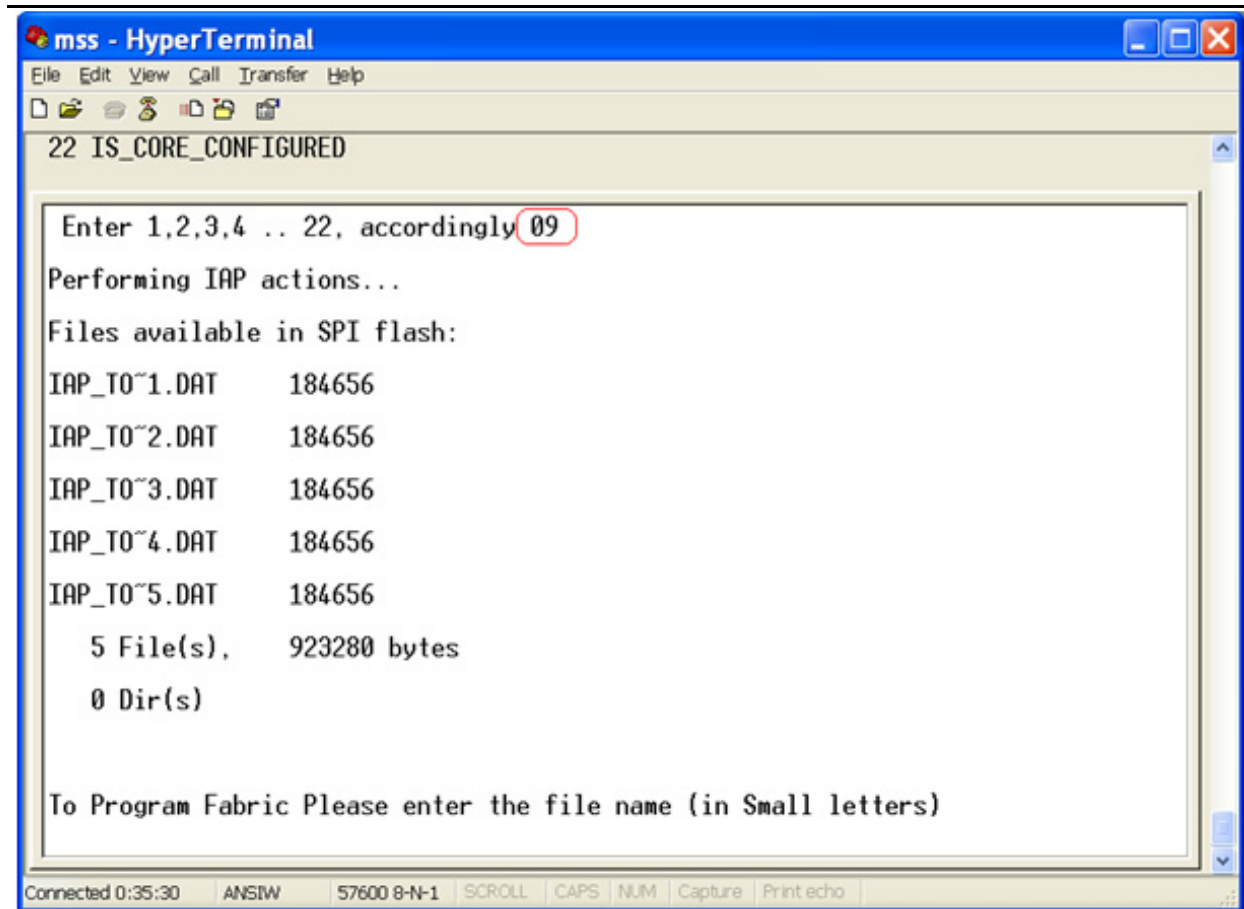


Figure 20 • Action Menu in HyperTerminal

You must select option 09 (program array) to program the FPGA fabric array. Figure 21 shows debug messages when the file system that is resident on the target board is present on SPI flash.



The screenshot shows a HyperTerminal window titled "mss - HyperTerminal". The menu text is as follows:

```
22 IS_CORE_CONFIGURED

Enter 1,2,3,4 .. 22, accordingly 09
Performing IAP actions...
Files available in SPI flash:
IAP_TO~1.DAT      184656
IAP_TO~2.DAT      184656
IAP_TO~3.DAT      184656
IAP_TO~4.DAT      184656
IAP_TO~5.DAT      184656
  5 File(s),      923280 bytes
  0 Dir(s)

To Program Fabric Please enter the file name (in Small letters)
```

The status bar at the bottom of the window displays: "Connected 0:35:30 | ANSIW | 57600 8-N-1 | SCROLL | CAPS | NUM | Capture | Print echo".

Figure 21 • Menu After Entering 09 (Program Array)

Figure 22 shows the debug messages after choosing one of the stored files for programming the FPGA fabric using the IAP interface.

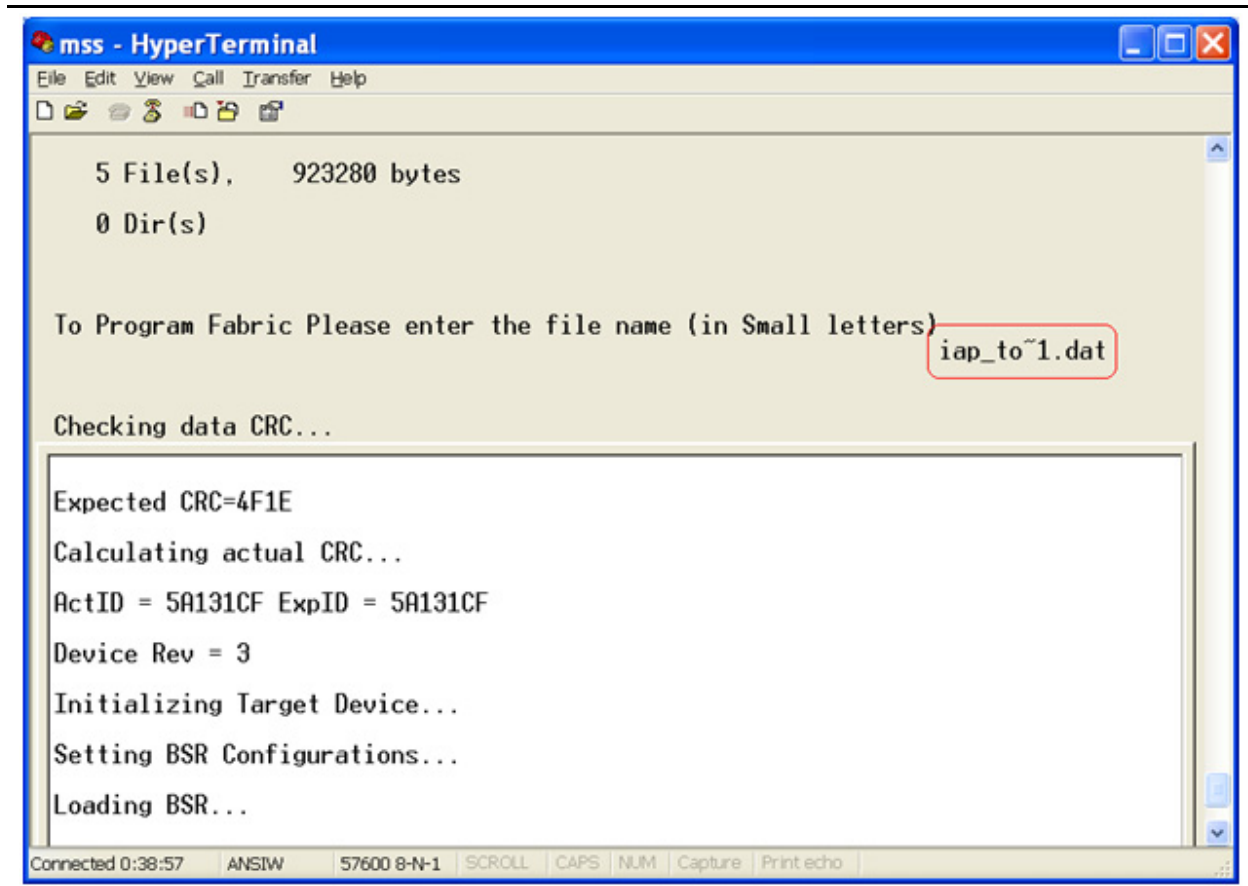


Figure 22 • Debug Messages during the FPGA Fabric Programming

For the UART interface method, there is no file system mounted on SPI flash. The IAP reads the programming file content from the SPI flash into eSRAM when requested by the IAP interface.

For the TFTP method, the file system is mounted on 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 completion of programming the FPGA fabric, you can observe the blinking LED sequence on the SmartFusion Development Kit or SmartFusion Evaluation Kit. There should be 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, eNVM programming is done with no file system mounted on SPI flash and the programming file for eNVM is loaded into SPI flash using the UART interface. The design files provided with this application note are used to program 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 23 on page 26](#) shows the debug messages during the programming when no file system is mounted on SPI flash.

After completion of programming eNVM, restart the board to see the message on the OLED for the *.dat file that was launched on the command prompt. You can observe the ENVM PROGRAMMING SUCCESSFUL message on OLED on the SmartFusion Development Kit or SmartFusion Evaluation Kit. This is visual evidence that eNVM has been programmed.



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 and SmartFusion Development Kit:

"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:

http://www.actel.com/documents/A2F_AC362_DF.zip

The design file consists of Libero IDE projects and SoftConsole software projects. Refer to the ReadMe.docx file for each solution included in the design file for directory structure and description.

List of Changes

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

Revision*	Changes	Page
Revision 1 (March 2011)	Modified the number '6' to '16' in Figure 2 (SAR 31182).	5

Note: *The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.



Microsemi Corporate Headquarters
2381 Morse Avenue, Irvine, CA 92614
Phone: 949-221-7100 · Fax: 949-756-0308
www.microsemi.com

Microsemi Corporation (NASDAQ: MSCC) offers the industry's most comprehensive portfolio of semiconductor technology. Committed to solving the most critical system challenges, Microsemi's products include high-performance, high-reliability analog and RF devices, mixed signal integrated circuits, FPGAs and customizable SoCs, and complete subsystems. Microsemi serves leading system manufacturers around the world in the defense, security, aerospace, enterprise, commercial, and industrial markets. Learn more at www.microsemi.com.

© 2011 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.