Spartan-3A DSP 3SD1800A MicroBlaze Processor Edition Kit Reference Systems

UG486 (v1.4) May 20, 2009

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
1/28/08	1.0	Initial Xilinx release.
6/2/08	1.1	Updated to EDK 10.1
12/19/08	1.2	Updated to EDK 10.1.3. Added information on the JFFS2 root file system and on the Apache Web server. Added chapter on the application to read, write, and erase the Flash.
1/15/09	1.2.1	Accompanying reference system published to proper link. Link to system updated.
1/27/09	1.3	Updated copyright date. Updated BlueCat Linux bootloader source code, bit file, and bin file with KDI image size.
5/20/09	1.4	Updated to EDK 11.1.

Table of Contents

Chapter 4: FlashRWE Software Application

About This Guide

The Embedded Development HW/SW Kit - Spartan®-3A DSP S3D1800A MicroBlaze™ Processor Edition showcases various features of the Spartan-3A DSP 1800A development board. This kit includes a reference system with a HelloWorld software application and a bootable BlueCat Linux image. This document describes the hardware platform, the HelloWorld software application, and the BlueCat Linux image.

The reference system is available at

https://secure.xilinx.com/webreg/clickthrough.do?cid=114560

Guide Contents

This manual contains the following chapters:

- Chapter 1, "Hardware Platform," provides an overview of the IP cores in the reference system. This chapter includes the reference system block diagram and address map.
- Chapter 2, "HelloWorld Software Application," describes the board tests in the application, how to execute the application, and how to boot the application from SPI Flash.
- Chapter 3, "LynuxWorks BlueCat Linux," includes information on how to execute the provided BlueCat Linux image and how to build a similar image using the BlueCat Linux development tools.
- Chapter 4, "FlashRWE Software Application" describes the available functions in the application and how to execute the application.

Hardware and Software Requirements

The hardware and software requirements are:

- Xilinx Spartan-3A DSP 1800A Development Board
- Xilinx Platform USB Download Cable or Parallel IV Download Cable
- RS232 Serial Cable
- Ethernet Cable
- Serial Communications Utility Program (such as HyperTerminal)
- Xilinx Platform Studio (XPS) 11.1
- ISE® 11.1

References

References used throughout this user guide are listed below.

1. *BlueCat Linux User's Guide*
2. *BlueCat Linux Board Support Guide for Xilinx Spartan-3E 1600E Boards*
3. UG485 *Getting Started with the Spartan-3A DSP 1800A Starter Platform User Guide*
4. XAPP1106 *Using and Creating Flash Files for the MicroBlaze™ Development Kit - Spartan-3A DSP Starter Platform*

Additional Resources

To find additional documentation, see the Xilinx website at:

http://www.xilinx.com/literature.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

http://www.xilinx.com/support.

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
`Courier font`	Messages, prompts, and program files that the system displays	`speed grade: - 100`
`Courier bold`	Literal commands that you enter in a syntactical statement	**`ngdbuild`** *design_name*
Helvetica bold	Commands that you select from a menu	**File → Open**
	Keyboard shortcuts	**Ctrl+C**
Italic font	Variables in a syntax statement for which you must supply values	**`ngdbuild`** *design_name*
	References to other manuals	See the *Development System Reference Guide* for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.

Convention	Meaning or Use	Example
Square brackets [ ]	An optional entry or parameter. However, in bus specifications, such as **bus[7:0]**, they are required.	**ngdbuild** [*option_name*] *design_name*
Braces { }	A list of items from which you must choose one or more	**lowpwr =**{**on**|**off**}
Vertical bar |	Separates items in a list of choices	**lowpwr =**{**on**|**off**}
Vertical ellipsis . . .	Repetitive material that has been omitted	IOB #1: Name = QOUT' IOB #2: Name = CLKIN' . . .
Horizontal ellipsis ...	Repetitive material that has been omitted	**allow block** *block_name loc1 loc2 ... locn;*

Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section "Additional Resources" for details. Refer to "Title Formats" in Chapter 1 for details.
Red text	Cross-reference link to a location in another document	See Figure 2-5 in the *Virtex-II Platform FPGA User Guide.*
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest speed files.

Chapter 1

Hardware Platform

Introduction

This reference system targets the Spartan-3A DSP 1800A development board. The system is created to run the HelloWorld software application described in Chapter 2, "HelloWorld Software Application.", the BlueCat Linux image described in Chapter 3, "LynuxWorks BlueCat Linux.", and the FlashRWE software application described in Chapter 4, "FlashRWE Software Application". The system uses the MicroBlaze processor with cache turned on for both the instruction cache and the data cache. As shown in Figure 1-1, the system includes various IP cores used in embedded systems.

See Table 1-1 for the address map of the system.

Block Diagram

The block diagram for the system is shown in Figure 1-1.

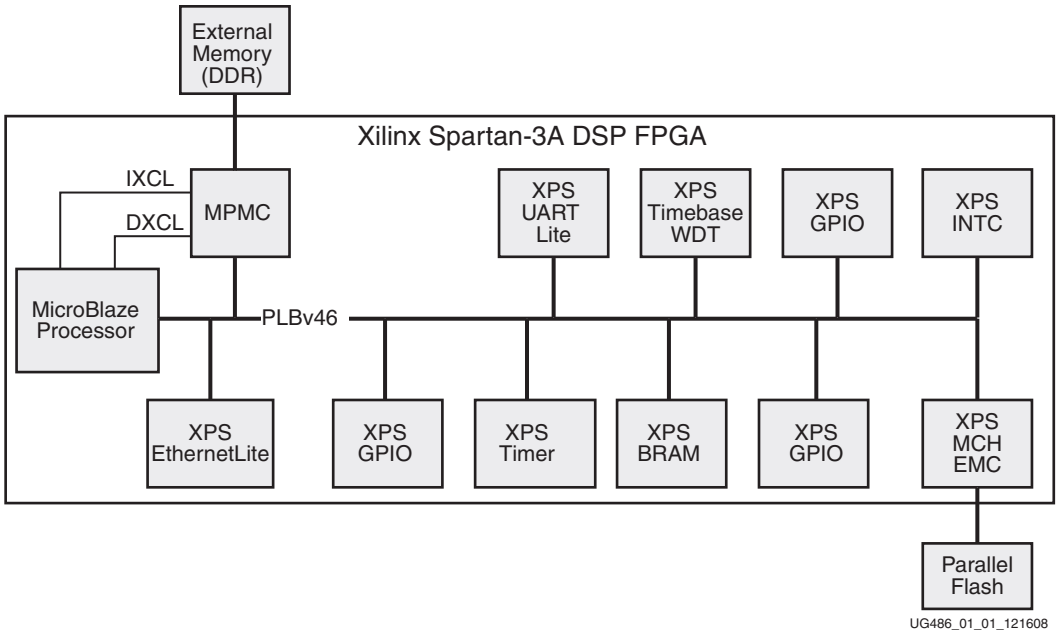


Figure 1-1: **Block Diagram**

Address Map

The address map for the IP cores in the reference system is given in Table 1-1.

Table 1-1: **Reference System Address Map**

Instance	Peripheral	Base Address	High Address
dlmb_cntlr	lmb_bram_if_cntlr	0x00000000	0x00001FFF
ilmb_cntlr	lmb_bram_if_cntlr	0x00000000	0x00001FFF
debug_module	mdm	0x84400000	0x8440FFFF
xps_bram_if_cntlr_1	xps_bram	0x85A10000	0x85A1FFFF
Ethernet_MAC	xps_ethernetlite	0x81000000	0x8100FFFF
Push_Buttons	xps_gpio	0x81400000	0x8140FFFF
LEDs_8Bit	xps_gpio	0x81420000	0x8142FFFF
DIP_Switches_8Bit	xps_gpio	0x81440000	0x8144FFFF
xps_intc_0	xps_intc	0x81800000	0x8180FFFF
xps_timebase_wdt_1	xps_timebase_wdt	0x83A00000	0x83A0FFFF
xps_timer_1	xps_timer	0x83C00000	0x83C0FFFF
RS232_Uart_1	xps_uartlite	0x84000000	0x8400FFFF
FLASH	xps_mch_emc	0x87000000	0x87FFFFFF
DDR2_SDRAM	mpmc	0x88000000	0x8FFFFFFF

System Configuration

This system runs off a reference clock frequency of 125 MHz from the oscillator on the board. The PLBv46 bus and MicroBlaze processor run at 62.5 Mhz, while the DDR2 memory runs at 125 MHz.

MicroBlaze Processor Configuration

The MicroBlaze processor is configured with the Memory Management Unit (MMU) enabled. The MMU is enabled by setting the MicroBlaze parameter C_USE_MMU to 3. This parameter implements the MMU in Virtual mode. In Virtual mode, the MMU controls effective-address to physical-address mapping and supports memory protection. Virtual mode provides greater control over memory protection. Protection and relocation enable system software to support multitasking. This capability gives the appearance of simultaneous or near-simultaneous execution of multiple programs.

The instruction cache and data cache are both enabled, with a cache size of 4KB. The cacheable block of main memory is accessed via the XCL Port Interface Modules (PIM) of the Multi-Port Memory Controller (MPMC).

More information about the MMU, the instruction cache, and the data cache, can be found in the *MicroBlaze Processor Reference Guide*.

XPS EthernetLite Configuration

The BlueCat Linux RTOS requires that the XPS EthernetLite has the interrupts set to on. In the BlueCat Linux demonstration, the Ethernet MAC can run at 10 Mbps or 100 Mbps, depending on the attached network. No other special settings are needed.

XPS MCH EMC Configuration

The XPS MCH EMC memory controller is connected to an external Intel J3 Parallel Flash device, which is used to store the hardware configuration bitstream and bootloader application, as well as the BlueCat Linux kernel image.

XPS UART Lite Configuration

The XPS UART Lite core is configured to use interrupts and is set to a baud rate of **115200**, **8** data bits, and no parity.

Chapter 2

HelloWorld Software Application

Introduction

The HelloWorld software application is a simple application that exercises a few of the board features. When the application is run, it will first flash the LEDs and read the DIP and push button switches. Then, the user can select from a list of menu options, including options to allow the user to select a target memory and read or write an address with necessary data.

The methods for downloading and running the HelloWorld software application are listed below:

- Use a debugger, such as XMD (provided as part of the EDK tools), to download the executable file directly into BRAM, through the MicroBlaze Debug Module (MDM). This method is described in the section "Executing the HelloWorld Software Application".
- Program Flash memory with the HelloWorld software application. This method is described in the section "Booting the HelloWorld Application from Serial Flash". Once Flash memory is programmed, the HelloWorld software application can be run by setting the FPGA configuration mode pins to SPI mode and either powering up the development board or depressing the PROG button on the board.

Note: A warning box will appear during some of the steps in this chapter. The warning box states that "Software development features in XPS are deprecated, and will be removed in the next major release". Click OK to safely ignore this warning. To turn off this warning completely, navigate to Edit→Preferences in XPS. Select Application Preferences and check the box that states "Do not show "Software Features Deprecated" dialog box".

Executing the HelloWorld Software Application

To execute the HelloWorld software application, the hardware bitstream must be programmed to the Spartan-3A DSP device and the HelloWorld software application loaded into BRAM. Programming the bitstream can be done by either downloading the pre-built bitstream from the `ready_for_download` directory or generating and downloading it from XPS. Similarly, the HelloWorld executable can be downloaded from the `ready_for_download` directory or built through XPS.

Executing the HelloWorld Application Using the Pre-Built Bitstream

To execute the application using the files in the `ready_for_download` directory in the project root directory, follow the subsequent steps:

1. Connect the Platform USB cable or the Parallel IV JTAG cable between the host computer and the Spartan-3A DSP 1800A Starter Board (J2).

2. Connect the serial cable between the host computer and the RS232 port (P2) on the Spartan-3A DSP 1800A Starter Board.
3. Apply power to the Spartan-3A DSP 1800A Starter Board.
4. Start a HyperTerminal (or similar) session on the host computer with the settings shown in Figure 2-1. Select the COM port corresponding to the connected serial port on the host computer. Set the Baud Rate to **115200**, Data bits to **8** bits, Parity to **None**, Stop bits to **1** bit, and Flow control to **None**.

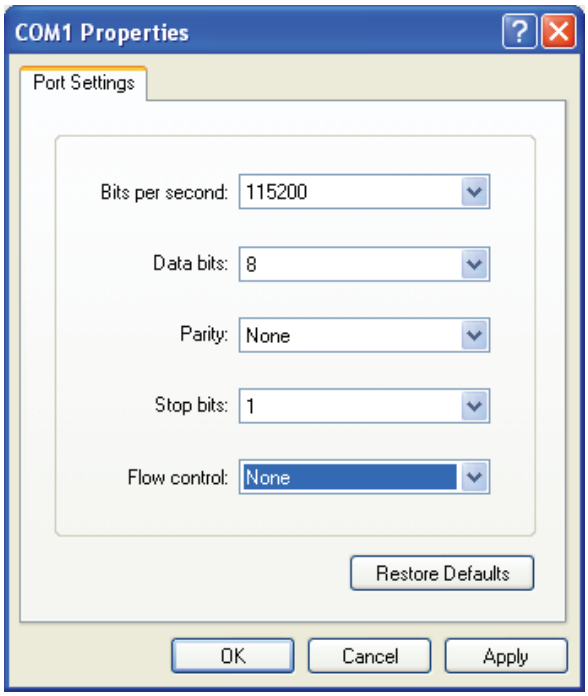


Figure 2-1: **HyperTerminal Settings**

5. In an EDK shell, change directories to the `ready_for_download` directory.
6. Use iMPACT to download the bitstream by using the following command:

   ```
   $ impact -batch ug486.cmd
   ```

7. Invoke XMD and connect to the processor by using the following command:

   ```
   $ xmd -opt ug486.opt
   ```

8. Download the HelloWorld software application into BRAM by using the following command:

   ```
   XMD% dow helloworld_executable.elf
   ```

9. To start the HelloWorld software application running, use the following XMD command:

   ```
   XMD% run
   ```

   a. After the HelloWorld software application runs, the HyperTerminal output will be as shown in Figure 2-2.

```
**************************************************************
**************************************************************
**          Xilinx Spartan-3A DSP 1800A Starter Kit         **
**************************************************************
**************************************************************

Walking the LEDs test.. Observe the LEDs...

 LEDs test PASSED.

Writing pseudo random data at address... 0x88000FFC
Reading pseudo random data at address..  0x88000FFC

 Memory Test PASSED!

 Press any key to continue....

 Type <Menu> for options

->
```

UG486_02_02_121608

Figure 2-2: **HelloWorld Output**

   b. For an explanation of the available tests in the application, see the section "Commands in the HelloWorld Software Application".

Executing the HelloWorld Software Application from XPS

To execute the reference system using XPS, follow these steps:

1. Perform steps 1-4 in the "Executing the HelloWorld Application Using the Pre-Built Bitstream" section.
2. Open the reference system project in XPS.
3. In the Applications tab, select the **helloworld** project for BRAM initialization by right-clicking on the project and selecting **Mark to Initialize BRAMs**. Ensure that no other applications are marked for BRAM initialization.
4. Implement the hardware design and create the hardware bitstream by selecting **Hardware** → **Generate Bitstream** in XPS.
5. Download the bitstream to the board by selecting **Device Configuration** → **Download Bitstream** in XPS. After the bitstream has downloaded, the HelloWorld application will execute from BRAM.
   a. After the HelloWorld software application runs, the HyperTerminal output will be as shown in Figure 2-2.
   b. For an explanation of the available commands in the application, see the section "Commands in the HelloWorld Software Application".

Commands in the HelloWorld Software Application

After the HelloWorld application is executed, type **Menu** into the terminal console to bring up the HelloWorld menu of tests, which is shown in Figure 2-3.

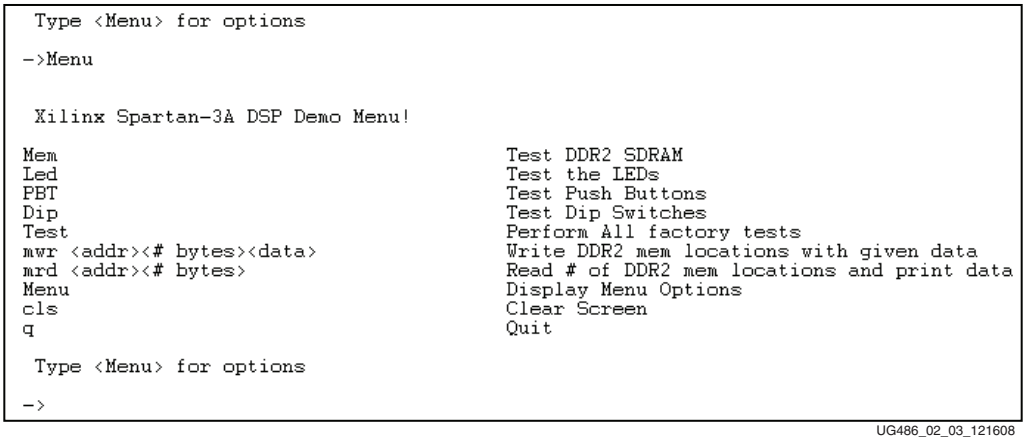


Figure 2-3: **HelloWorld Menu**

Table 2-1 lists the commands that are available in the HelloWorld application and describes each one.

Table 2-1: **Description of the HelloWorld Commands**

Command	Description
Mem	The Mem test performs a destructive 32-bit wide memory test on the DDR2 SDRAM memory. This test erases, writes, reads, and verifies the DDR2 memory in the Spartan-3A DSP 1800A development board. The results of the test will be displayed in the HyperTerminal.
Led	The Led test flashes each LED with a delay so that it is visible. Once all the LEDs are flashed, it sends the test pass message to the HyperTerminal.
PBT	The PBT test reads the push buttons and sends a message to the HyperTerminal on the value of the push button pressed. Hold down the push buttons before running the command.
Dip	The Dip test reads the DIP switches on the Spartan-3A DSP 1800A development board and displays the results to the HyperTerminal.
Test	This performs all the factory tests mentioned above for the Spartan-3A DSP 1800A development board and displays the results to the HyperTerminal.
mwr <addr><# bytes><data>	This test writes the given data to the DDR2 memory locations specified. The address range should be within the DDR2 base address and high address.
mrd <addr><# bytes>	This test reads the number of bytes specified from the DDR2 memory location given. The address range should be within the DDR2 base address and high address.
Menu	This command lists the menu options for the user.
cls	This command clears the HyperTerminal screen.

Booting the HelloWorld Application from Serial Flash

This section includes steps on how to program the HelloWorld application into the SPI Flash. These steps refer to XAPP1106: Using and Creating Flash Files for the MicroBlaze Development Kit - Spartan-3A DP 1800A Starter Platform, which includes details on how to use, create, and boot SPI Flash files for the MicroBlaze Spartan-3A DSP 1800A Edition Development Kit. Flash files that have already been generated are provided, or the user can create new files for programming the Flash.

Programming the Flash with the Provided Files

Flash files that have already been generated and are ready to use can be found in the `<project root directory>/ready_for_download/Flash_files/` directory.

1. Follow the steps outlined in the section "Programming the Serial Flash with the Provided HelloWorld Files" in XAPP1106. These steps will provide information on how to program the Flash with the Helloworld application
2. Set the jumpers on the Spartan-3A DSP 1800A board to SPI mode (M1 and M2 closed). Apply power to the development board or press the PROG button if power is already applied. This will boot the FPGA in SPI mode.

Generating New Flash Files and Programming the Flash

Instead of using the pregenerated files, the user can generate new files for programming the Flash device. This section details the steps for creating new Flash files and programming them into the Flash device.

1. Follow the steps outlined in the section "Creating and Programming New Serial Flash Files for the Helloworld Application" in XAPP1106. These steps will provide information on how to program the Flash with the Helloworld application.
2. Set the jumpers on the Spartan-3A DSP 1800A board to SPI mode (M1 and M2 closed) and press the PROG button. This will boot the FPGA in SPI mode.

Chapter 3

LynuxWorks BlueCat Linux

Introduction

The BlueCat Linux reference system demonstrates BlueCat Linux running on the MicroBlaze soft processor with the MMU enabled. An example BlueCat Linux image is provided that is tailored to the Spartan-3A DSP 1800A Edition Development Kit board and the hardware platform that is described in Chapter 1, "Hardware Platform.". Two BlueCat Linux images are provided with the hardware system, one that boots with a ramdisk root file system and one that uses a Journalling Flash File System, version 2 (JFFS2). The Spartan-3A DSP 1800A Development Kit also includes example demo directories, which allow the user to rebuild the example kernel images with the LynuxWorks BlueCat Linux development tools.

The methods for downloading and running the BlueCat Linux kernel demonstration are listed below.

- Use a debugger, such as XMD (provided as part of the EDK tools), to download the image file directly into DDR2, through the MicroBlaze Debug Module (MDM). This method is described in the section "Executing the BlueCat Linux Images".
- Program Flash memory with the BlueCat Linux image. This method is described in the section "Booting the BlueCat Linux Image from Parallel Flash". Once Flash memory is programmed, the BlueCat Linux demonstration can be run by setting the FPGA configuration mode pins to BPI mode and either powering up the development board or depressing the PROG button on the board.

Note: A warning box will appear during some of the steps in this chapter. The warning box states that "Software development features in XPS are deprecated, and will be removed in the next major release". Click OK to safely ignore this warning. To turn off this warning completely, navigate to Edit→Preferences in XPS. Select Application Preferences and check the box that states "Do not show "Software Features Deprecated" dialog box".

Executing the BlueCat Linux Images

Two BlueCat Linux images are provided with the hardware system. One BlueCat Linux image uses a ramdisk root file system. The root file system is included in the image so it is a self contained image that can be booted quickly. The other BlueCat Linux image uses a JFFS2 root file system. The JFFS2 file system must be written to the Flash memory before the Linux image can be booted, but it allows for persistent storage. This section details how to execute the different BlueCat Linux images.

Executing the BlueCat Linux Image with a Ramdisk File System

To boot the BlueCat Linux reference system, the hardware bitstream must be programmed to the Spartan-3A DSP device and the BlueCat Linux kernel image must be downloaded to the DDR2 memory. Programming the bitstream can be done by either downloading the pre-built bitstream from the `ready_for_download` directory or generating and downloading it from XPS. The BlueCat Linux kernel image is downloaded from the `bclinux_images` directory.

Executing the BlueCat Linux Image Using the Pre-Built Bitstream

To execute the reference system using the files inside the `ready_for_download` directory in the project root directory, follow these steps:

1. Connect the Platform USB cable or the Parallel IV JTAG cable between the host computer and the Spartan-3A DSP 1800A Starter Board (J2).
2. Connect the serial cable between the host computer and the RS232 port (P2) on the Spartan-3A DSP 1800A Starter Board.
3. Apply power to the Spartan-3A DSP 1800A Starter Board.
4. Start a HyperTerminal (or similar) session on the host computer with the settings shown in Figure. Select the COM port corresponding to the connected serial port on the host computer. Set the Baud Rate to **115200**, Data bits to **8** bits, Parity to **None**, Stop bits to **1** bit, and Flow control to **None**, as shown in Figure 3-1.

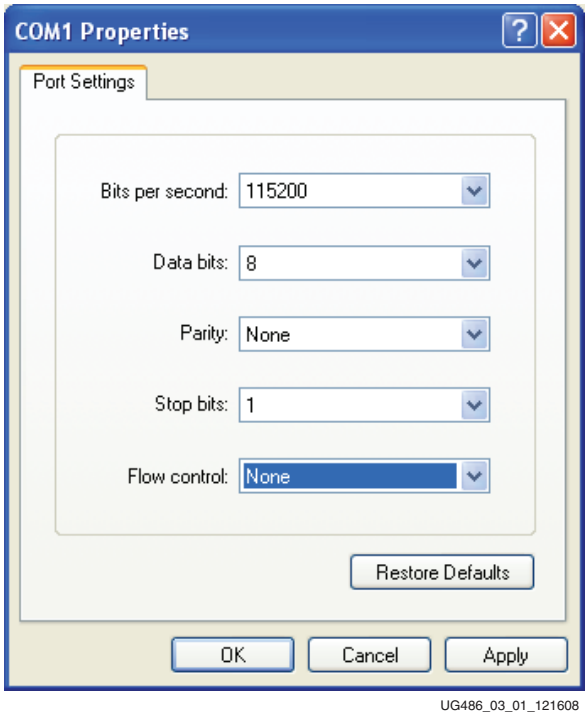


Figure 3-1: **HyperTerminal Settings**

5. Through XPS, launch an EDK shell by selecting **Project → Launch EDK Shell.**
6. In the EDK shell, change directories to the `ready_for_download` directory.
7. Use iMPACT to download the bitstream by using the following command:

   ```
   $ impact -batch ug486.cmd
   ```

8. Invoke XMD and connect to the processor by using the following command:

   ```
   $ xmd -opt ug486.opt
   ```

9. Download the BlueCat Linux kernel image into DDR2 memory at the starting location `0x88000000` using the following command:

   ```
   XMD% dow -data ../bclinux_images/s3adsp_devl_kit_demo.kdi
   0x88000000
   ```

   Note: It may take several minutes to download the BlueCat Linux image into memory.

10. To start the kernel image running and boot BlueCat Linux, use the following XMD command:

    ```
    XMD% con 0x88000000
    ```

a. After BlueCat Linux boots, the HyperTerminal output will be as shown in Figure 3-2.

```
Linux version 2.6.13.4 () (gcc version 4.1.1) #1 Fri Apr 24 16:18:30 EDT
2009
On node 0 totalpages: 32768
  DMA zone: 32768 pages, LIFO batch:15
  Normal zone: 0 pages, LIFO batch:1
  HighMem zone: 0 pages, LIFO batch:1
Built 1 zonelists
Kernel command line: ramdisk_size=65535 xilinx_emc_part_conf=0-7:8-20:21-99:100-127 hda=
bswap hdb=bswap hdc=bswap hdd=bswap root=101
xps_intc_2.00.a INTC at 0x81800000 mapped to 0xFDFFF000
PID hash table entries: 1024 (order: 10, 16384 bytes)
xps_timer_1.01.a TIMER at 0x83C00000 mapped to 0xFDFFE000
Console: Xilinx UART Lite
Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)
Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)
Memory: 121728k available
Calibrating delay loop... 30.82 BogoMIPS (lpj=154112)
Mount-cache hash table entries: 512
NET: Registered protocol family 16
JFFS2 version 2.2. (NAND) (C) 2001-2003 Red Hat, Inc.
xgpio00 #0 at 0x81440000 mapped to 0xC8060000 device: 10,185 using IRQ#3
xgpio01 #1 at 0x81420000 mapped to 0xC8080000 device: 10,186 not using IRQ
xgpio02 #2 at 0x81400000 mapped to 0xC80A0000 device: 10,187 using IRQ#4
ttyS0 at MMIO 0x84000000 (irq = 2) is a Xilinx UART Lite
io scheduler noop registered
io scheduler anticipatory registered
io scheduler deadline registered
io scheduler cfq registered
RAMDISK driver initialized: 16 RAM disks of 65535K size 1024 blocksize
eth0: using fifo mode.
eth0: No PHY detected.  Assuming a PHY at address 0.
eth0: Xilinx EMACLite #0 at 0x81000000 mapped to 0xC80C0000, irq=1
EMC Flash on Xilinx board: Found 1 x16 devices at 0x0 in 8-bit bank
 Intel/Sharp Extended Query Table at 0x0031
cfi_cmdset_0001: Suspend erase on write disabled.
Using buffer write method
0: offset=0x0,size=0x20000,blocks=128
Registering a 16MB EMC Flash at 0x87000000
EMC Flash MTD driver: Configuration of partitions is 0-7:8-20:21-99:100-127
Creating 4 MTD partitions on "EMC Flash on Xilinx board":
0x00000000-0x00100000 : "EMC Flash on Xilinx board"
0x00100000-0x002a0000 : "EMC Flash on Xilinx board"
0x002a0000-0x00c80000 : "EMC Flash on Xilinx board"
0x00c80000-0x01000000 : "EMC Flash on Xilinx board"
EMC Flash MTD driver: Configured 4 partitions
NET: Registered protocol family 2
IP route cache hash table entries: 2048 (order: 1, 8192 bytes)
TCP established hash table entries: 8192 (order: 4, 65536 bytes)
TCP bind hash table entries: 8192 (order: 3, 32768 bytes)
TCP: Hash tables configured (established 8192 bind 8192)
TCP reno registered
TCP bic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
RAMDISK: Compressed image found at block 2229816
Freeing BlueCat RFS memory: 6325k freed
VFS: Mounted root (ext2 filesystem).
Freeing unused kernel memory: 72k freed
INIT: version 2.85 booting
Starting Apache HTTP server
Running the DHCP client
INIT: Entering runlevel: 1

myhostname login:
```

UG486_03_02_052009

Figure 3-2: **BlueCat Linux Boot Output - Ramdisk File System**

b. Log into BlueCat Linux by using the username **`root`**.

c. For example commands to run in BlueCat Linux, see the section "Executing BlueCat Linux Commands".

Executing the BlueCat Linux Image from XPS

To execute the reference system using XPS, follow these steps:

1. Perform steps 1-4 in the "Executing the BlueCat Linux Image Using the Pre-Built Bitstream" section.
2. Open the reference system project in XPS.
3. Implement the hardware design and create the hardware bitstream by selecting **Hardware** → **Generate Bitstream** in XPS.
4. Download the bitstream to the board by selecting **Device Configuration** → **Download Bitstream** in XPS.
5. Select **Debug** → **Launch XMD...** to launch an XMD command window.
6. In XMD, download the BlueCat Linux kernel image into DDR2 memory at the starting location `0x88000000` using the following command:

   ```
   XMD% dow -data bclinux_images/s3adsp_devl_kit_demo.kdi
   0x88000000
   ```

   Note: This step may take several minutes to download the BlueCat Linux image into memory.
7. To start the kernel image running and boot BlueCat Linux, use the following XMD command:

   ```
   XMD% con 0x88000000
   ```

   a. After BlueCat Linux boots, the HyperTerminal output will be as shown in Figure 3-2.

   b. Log into BlueCat Linux by using the username **`root`**.

   c. For example commands to run in BlueCat Linux, see the section "Executing BlueCat Linux Commands".

Executing the BlueCat Linux Image with a JFFS2 File System

To boot the BlueCat Linux reference system, the hardware bitstream must be programmed to the Spartan-3A DSP device, the BlueCat Linux kernel image must be downloaded to the DDR2 memory, and the root file system must be written to the parallel Flash. Programming the bitstream can be done by either downloading the pre-built bitstream from the `ready_for_download` directory or generating and downloading it from XPS. The BlueCat Linux kernel image is downloaded from the `bclinux_images` directory. The root file system is found in the `bclinux_images` directory and can be programmed to the Flash device in XPS after the blocks of Flash that hold the file system have been erased.

Executing the BlueCat Linux Image Using the Pre-Built Bitstream

To execute the reference system using the files inside the `ready_for_download` directory in the project root directory, follow these steps:

1. Connect the Platform USB cable or the Parallel IV JTAG cable between the host computer and the Spartan-3A DSP 1800A Starter Board (J2).
2. Connect the serial cable between the host computer and the RS232 port (P2) on the Spartan-3A DSP 1800A Starter Board.
3. Apply power to the Spartan-3A DSP 1800A Starter Board.

4. Start a HyperTerminal (or similar) session on the host computer with the settings shown in Figure. Select the COM port corresponding to the connected serial port on the host computer. Set the Baud Rate to **115200**, Data bits to **8** bits, Parity to **None**, Stop bits to **1** bit, and Flow control to **None**, as shown in Figure 3-3.

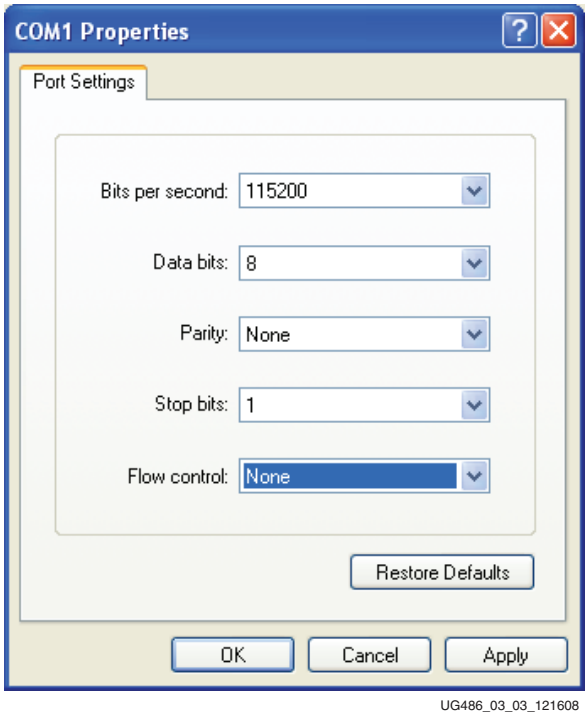


Figure 3-3: **HyperTerminal Settings**

5. Through XPS, launch an EDK shell by selecting **Project → Launch EDK Shell.**
6. In the EDK shell, change directories to the `ready_for_download` directory.
7. Use iMPACT to download the bitstream by using the following command:

   ```
   $ impact -batch ug486.cmd
   ```

8. Invoke XMD and connect to the processor by using the following command:

   ```
   $ xmd -opt ug486.opt
   ```

9. Download the FlashRWE software application into BRAM using the following command:

   ```
   XMD% dow flashrwe_executable.elf
   ```

10. To start the FlashRWE software application running, use the following XMD command:

    ```
    XMD% run
    ```

    After the FlashRWE software application runs, the HyperTerminal will display the Main Menu.

11. With the FlashRWE program, erase blocks 21-99 of Flash. These blocks are the location that the BlueCat Linux kernel image will expect the JFFS2 root file system. To erase the blocks, perform the following steps, as shown in Figure 3-4.
    a. Enter 3 at the Main Menu.
    b. In the Flash Erase Menu, enter 2.
    c. Enter 21 as the starting block.
    d. Enter 99 as the ending block.

```
Main Menu:
1 - Read Flash Contents
2 - Write to Flash
3 - Erase Flash
4 - Exit the Flash Program
Enter selection: 3

Flash Erase Menu:
1 - Erase Bytes of Flash
2 - Erase Blocks of Flash
3 - Erase the Entire Flash
4 - Exit to Main Menu
Enter selection: 2
Enter starting block (0-127) to erase: 21
Enter ending block (21-127) to erase: 99
Erasing blocks 21-99 of Flash...
....................
Erased the Flash memory contents successfully
```

UG486_03_04_121608

Figure 3-4: **Erase Blocks 21-99 of the Flash**

12. In XMD, stop and reset the processor. Then, exit XMD.

```
XMD% stop
XMD% rst
XMD% exit
```

13. In XPS, select **Device Configuration → Program Flash Memory**.

14. In the Program Flash Memory dialog box, choose the file to program to be `/bclinux_images/s3adsp_devl_kit_demo_flash.jffs2`. Enter the offset to be `0x002A0000`. The external DDR2 memory is set as the Scratch Memory. The Program Flash Memory settings are shown in Figure 3-5.

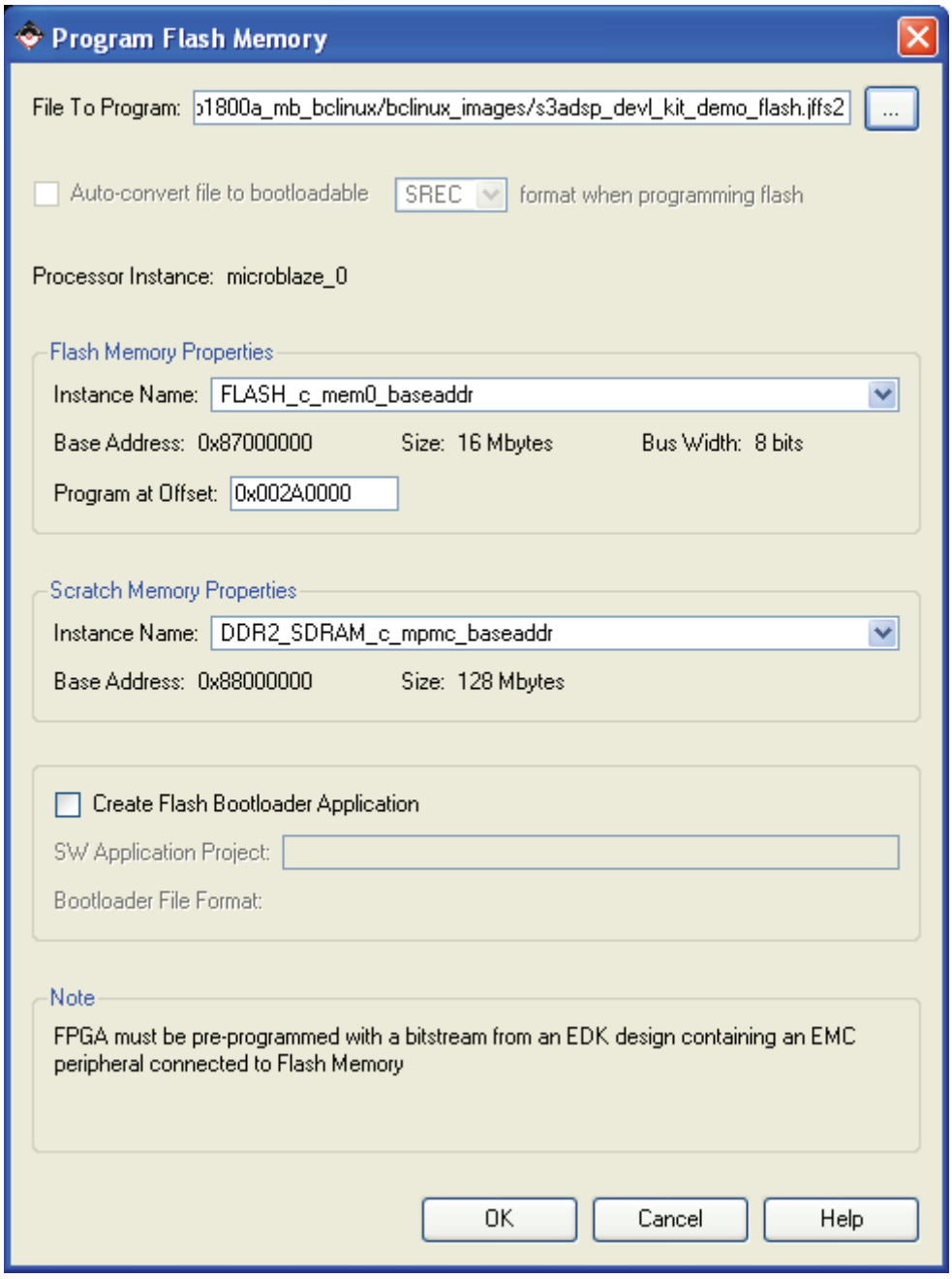


Figure 3-5: **Program Flash Memory Box for the JFFS2 File System**

15. In an EDK shell in the `ready_for_download` directory, invoke XMD and connect to the processor by the following command:

```
$ xmd -opt ug486.opt
```

16. Download the BlueCat Linux kernel image that uses the JFFS2 file system into DDR2 memory at the starting location `0x88000000` using the following command:

    ```
    XMD% dow -data ../bclinux_images/s3adsp_devl_kit_demo_flash.kdi
    0x88000000
    ```

    Note: It may take several minutes to download the BlueCat Linux image into memory.

17. To start the kernel image running and boot BlueCat Linux, use the following XMD command:

    ```
    XMD% con 0x88000000
    ```

    a. After BlueCat Linux boots, the HyperTerminal output will be as shown in Figure 3-6.

```
Linux version 2.6.13.4 () (gcc version 4.1.1) #1 Fri Dec 12 16:49:47 EST
2008
On node 0 totalpages: 32768
  DMA zone: 32768 pages, LIFO batch:15
  Normal zone: 0 pages, LIFO batch:1
  HighMem zone: 0 pages, LIFO batch:1
Built 1 zonelists
Kernel command line: rootfstype=jffs2 noinitrd rw xilinx_emc_part_conf=0-7:8-20:21-99:10
0-127 hda=bswap hdb=bswap hdc=bswap hdd=bswap root=1f03
xps_intc_1.00.a INTC at 0x81800000 mapped to 0xFDFFF000
PID hash table entries: 1024 (order: 10, 16384 bytes)
xps_timer_1.00.a TIMER at 0x83C00000 mapped to 0xFDFFE000
Console: Xilinx UART Lite
Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)
Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)
Memory: 128000k available
Calibrating delay loop... 30.82 BogoMIPS (lpj=154112)
Mount-cache hash table entries: 512
NET: Registered protocol family 16
JFFS2 version 2.2. (NAND) (C) 2001-2003 Red Hat, Inc.
xgpio00 #0 at 0x81420000 mapped to 0xC8060000 device: 10,185 not using IRQ
xgpio01 #1 at 0x81400000 mapped to 0xC8080000 device: 10,186 using IRQ#4
xgpio02 #2 at 0x81440000 mapped to 0xC80A0000 device: 10,187 using IRQ#3
ttyS0 at MMIO 0x84000000 (irq = 2) is a Xilinx UART Lite
io scheduler noop registered
io scheduler anticipatory registered
io scheduler deadline registered
io scheduler cfq registered
RAMDISK driver initialized: 16 RAM disks of 8192K size 1024 blocksize
eth0: using fifo mode.
eth0: No PHY detected.  Assuming a PHY at address 0.
eth0: Xilinx EMACLite #0 at 0x81000000 mapped to 0xC80C0000, irq=1
EMC Flash on Xilinx board: Found 1 x16 devices at 0x0 in 8-bit bank
 Intel/Sharp Extended Query Table at 0x0031
cfi_cmdset_0001: Suspend erase on write disabled.
Using buffer write method
0: offset=0x0,size=0x20000,blocks=128
Registering a 16MB EMC Flash at 0x87000000
EMC Flash MTD driver: Configuration of partitions is 0-7:8-20:21-99:100-127
Creating 4 MTD partitions on "EMC Flash on Xilinx board":
0x00000000-0x00100000 : "EMC Flash on Xilinx board"
0x00100000-0x002a0000 : "EMC Flash on Xilinx board"
0x002a0000-0x00c80000 : "EMC Flash on Xilinx board"
0x00c80000-0x01000000 : "EMC Flash on Xilinx board"
EMC Flash MTD driver: Configured 4 partitions
NET: Registered protocol family 2
IP route cache hash table entries: 2048 (order: 1, 8192 bytes)
TCP established hash table entries: 8192 (order: 4, 65536 bytes)
TCP bind hash table entries: 8192 (order: 3, 32768 bytes)
TCP: Hash tables configured (established 8192 bind 8192)
TCP reno registered
TCP bic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
RAMDISK: Couldn't find valid RAM disk image starting at 0.
VFS: Mounted root (jffs2 filesystem).
Freeing unused kernel memory: 72k freed
INIT: version 2.85 booting
Starting Apache HTTP server
Running the DHCP client
INIT: Entering runlevel: 1

myhostname login:
```

UG486_03_06_052009

Figure 3-6: **BlueCat Linux Boot Output - JFFS2 File System**

b. Log into BlueCat Linux by using the username **root**.

c. For example commands to run in BlueCat Linux, see the section "Executing BlueCat Linux Commands".

Executing the BlueCat Linux Image from XPS

To execute the reference system using XPS, follow these steps:

1. Perform steps 1-4 in the "Executing the BlueCat Linux Image Using the Pre-Built Bitstream" section.
2. Open the reference system project in XPS.
3. Implement the hardware design and create the hardware bitstream by selecting **Hardware** → **Generate Bitstream** in XPS.
4. Download the bitstream to the board by selecting **Device Configuration** → **Download Bitstream** in XPS.
5. Right click the FlashRWE software application project and select **Build Project** to create the executable file.
6. Select **Debug** → **Launch XMD...** to launch an XMD command window.
7. Download the FlashRWE software application into BRAM using the following command:

   ```
   XMD% dow FlashRWE/executable.elf
   ```

8. To start the FlashRWE software application running, use the following XMD command:

   ```
   XMD% run
   ```

   After the FlashRWE software application runs, the HyperTerminal will display the Main Menu.

9. With the FlashRWE program, erase blocks 21-99 of Flash. These blocks are the location that the BlueCat Linux kernel image will expect the JFFS2 root file system. To erase the blocks, perform the following steps.

   a. Enter 3 at the Main Menu.

   b. In the Flash Erase Menu, enter 2.

   c. Enter 21 as the starting block.

   d. Enter 99 as the ending block.

The steps for erasing blocks are shown in Figure 3-7 also.

```
Main Menu:
1 - Read Flash Contents
2 - Write to Flash
3 - Erase Flash
4 - Exit the Flash Program
Enter selection: 3

Flash Erase Menu:
1 - Erase Bytes of Flash
2 - Erase Blocks of Flash
3 - Erase the Entire Flash
4 - Exit to Main Menu
Enter selection: 2
Enter starting block (0-127) to erase: 21
Enter ending block (21-127) to erase: 99
Erasing blocks 21-99 of Flash...
....................
Erased the Flash memory contents successfully
```

UG486_03_07_121608

Figure 3-7: **Erase Blocks 21-99 of the Flash**

10. In XMD, stop and reset the processor. Then, exit XMD.

    ```
    XMD% stop
    XMD% rst
    XMD% exit
    ```

11. In XPS, select **Device Configuration → Program Flash Memory**.

12. In the Program Flash Memory dialog box, choose the file to program to be `/bclinux_images/s3adsp_devl_kit_demo_flash.jffs2`. Enter the offset to be `0x002A0000`. The external DDR2 memory is set as the Scratch Memory. The Program Flash Memory settings are shown in Figure 3-8.

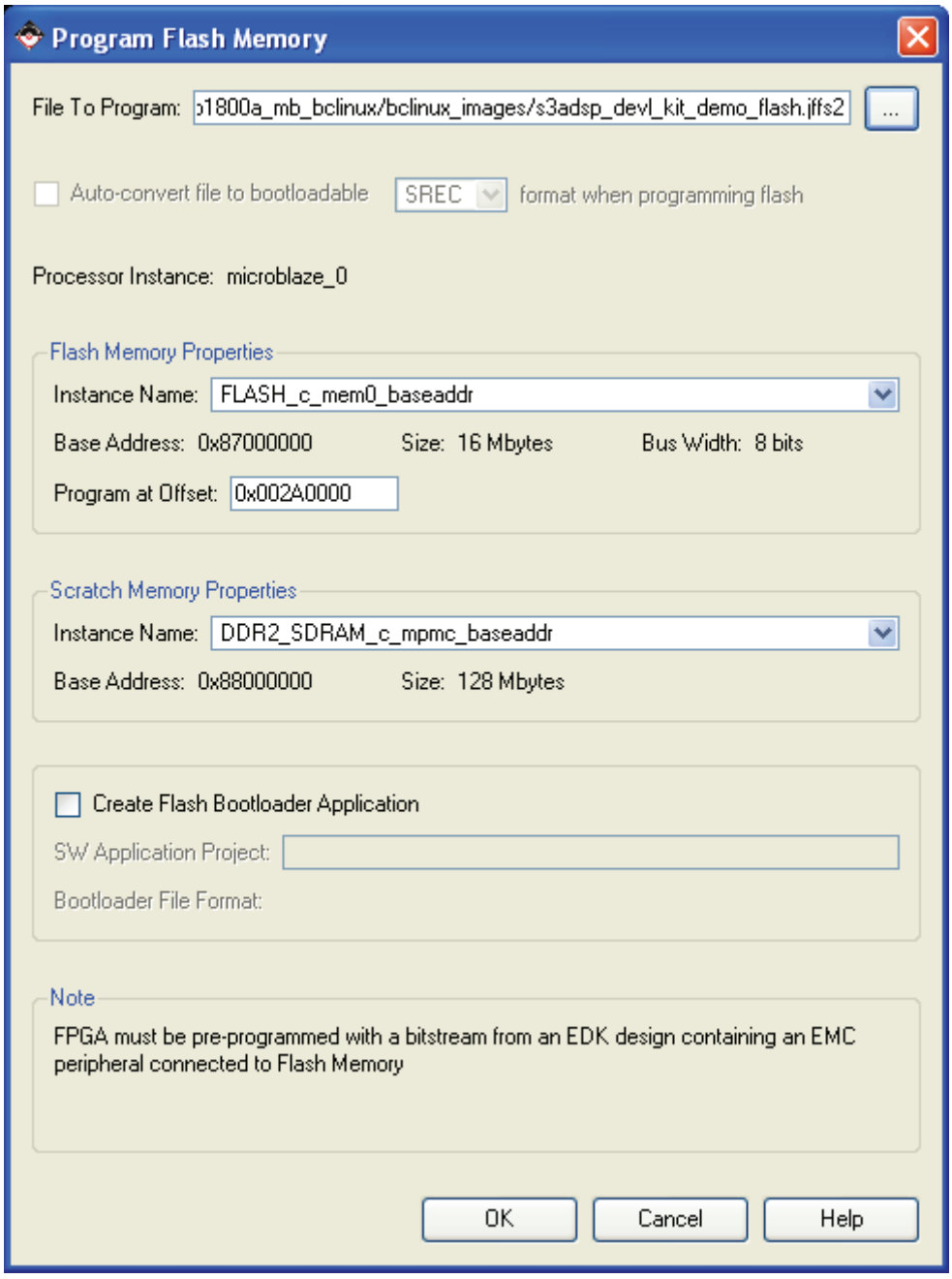


Figure 3-8: **Program Flash Memory Box for the JFFS2 File System**

13. Select **Debug** → **Launch XMD...** to launch an XMD command window.
14. Download the BlueCat Linux kernel image that uses the JFFS2 file system into DDR2 memory at the starting location `0x88000000` using the following command:

```
XMD% dow -data bclinux_images/s3adsp_devl_kit_demo_flash.kdi
0x88000000
```

Note: It may take several minutes to download the BlueCat Linux image into memory.

15. To start the kernel image running and boot BlueCat Linux, use the following XMD command:

    ```
    con 0x88000000
    ```

    a. After BlueCat Linux boots, the HyperTerminal output will be as shown in Figure 3-6.

    b. Log into BlueCat Linux by using the username **root**.

    c. For example commands to run in BlueCat Linux, see the section "Executing BlueCat Linux Commands".

Executing BlueCat Linux Commands

The BlueCat Linux images provided with the development kit support many basic Linux commands. The list of commands and tools available to be run are found under the `/bin` directory.

This BlueCat Linux kernel was built with networking support enabled, therefore it supports several network utilities when connected to a live network or connected directly to a remote computer.

The provided BlueCat Linux images include DHCP client support, and will try to retrieve an IP address during boot up. If unable to retrieve an IP address, the DHCP client will time out and an IP address will need to be set manually to use the networking features.

To view the Ethernet configuration settings, use the command **ifconfig**. Example results of using this command for the eth0 (Ethernet) and lo (Local Loopback) ports are shown in Figure 3-9. In the figure, the board IP address is 192.168.0.127. The board IP address can be changed by issuing the command **ifconfig eth0** *IP_address*.

```
-bash-3.00# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:00:C0:A3:E5:44
          inet addr:192.168.0.127  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:277 errors:0 dropped:0 overruns:0 frame:0
          TX packets:189 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:420486 (410.6 KiB)  TX bytes:103834 (101.4 KiB)
          Interrupt:2

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

UG486_03_09_121608

Figure 3-9: **Ethernet Configuration Settings**

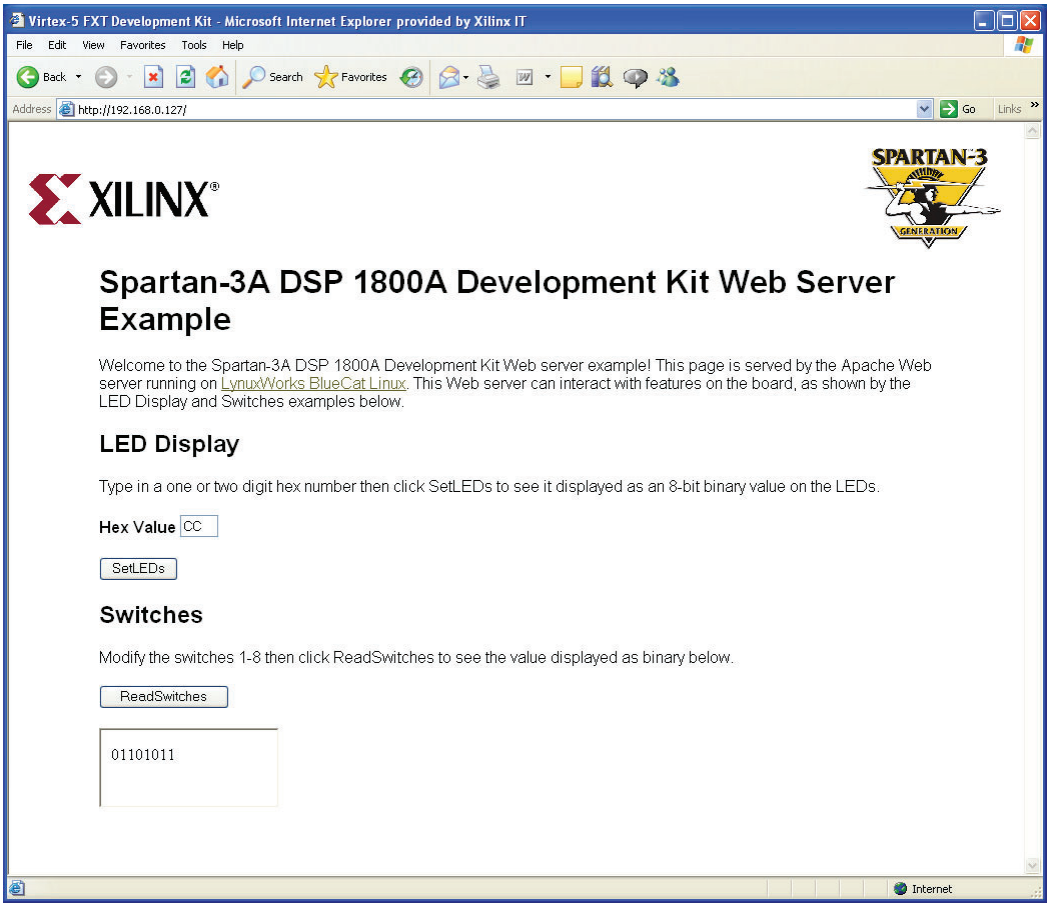
To ping a remote computer at IP address 172.17.1.200 from the development board, this example command string, **ping -c 4 172.17.1.200,** is used to ping the remote computer 4 times.

To FTP from a networked computer to the board, issue the command **`ftp`** *board_IP_address*. Files can now be transferred back and forth via FTP.

To telnet from a networked computer to the board, issue the command **`telnet`** *board_IP_address*. All of the Linux commands can now be performed remotely as if the user was logged into the console on a HyperTerminal.

Web Server Demonstration

The kernel images provided with the kit include the ability to run the Apache Web server. During boot up, the Web server will begin to run. The user can view the Web page from the Web server by going to http://<board_ip_address>. The Web page that is served is shown in Figure 3-10. On the Web page, the user can interact with the LEDs and the switches on the board. To set the LEDs, enter in a one or two digit hexadecimal number, then press **SetLEDs**. This will display the binary equivalent of the number on the LEDs. The Web page will also display the value of the DIP switches. Change the DIP switches on the board, then press the ReadSwitches button to update the Web page with the new switches value in binary.



UG486_03_10_121608

Figure 3-10: **Web Page Displayed by the Web Server**

Building the BlueCat Linux Kernel Image

This section briefly describes the process for rebuilding the kernel image that is included with this reference system. To rebuild the kernel, the BlueCat Linux distribution must be obtained from LynuxWorks. For more information on the LynuxWorks BlueCat Linux distribution, see the *BlueCat Linux User's Guide* for Release 5.4.

The steps described in this section include using the BlueCat Linux Spartan-3E BSP. Even though this reference system is for Spartan-3A DSP, the Spartan-3E BSP contains the needed components to build a working image for the Spartan-3A DSP. For more information on the Spartan-3E BSP, see the BlueCat Linux Board Support Guide for Xilinx Spartan-3E 1600E Boards.

The *BlueCat Linux User's Guide* and the *BlueCat Linux Board Support Guide for Xilinx Spartan-3E 1600E Boards* can be obtained from LynuxWorks at:
http://www.lynuxworks.com/support/bluecat/docs.php3

These steps assume the kernel is being built on a host system running Red Hat Enterprise Linux 4.0. All of the Linux commands must be run using a bash shell.

Installing the BlueCat Linux Distribution

These steps describe how to install the BlueCat Linux core components with the Spartan-3E 1600E BSP. For more information on the directory structures of the LynuxWorks BlueCat Linux distribution and the installation procedures, see the *BlueCat Linux User's Guide* reference above.

1. To install the BlueCat Linux core components on the host machine, follow the steps outlined in the "Installing the Default Configuration" section in the Introduction and Installation chapter of the *BlueCat Linux User's Guide*.
2. To install the Spartan-3E BSP on the host machine, follow the steps outlined in the "Installing Target Board Support" section in the Introduction and Installation chapter of the *BlueCat Linux User's Guide*.

   Note: When running the commands in these steps, bsp = sp3e.
3. After the SP3E BSP is installed, support for it must be activated in the bash shell. To activate the SP3E BSP, follow the steps in the "Activating Support for a Target Board" section in the Introduction and Installation chapter of the *BlueCat Linux User's Guide*.

Using the Provided Demonstration Directories

BlueCat Linux demonstration directories are provided with the reference system which will allow the user to rebuild the BlueCat Linux images that are included. These demo directories can be found in `<project root directory>/bclinux_demo/`. These directories should be unzipped and placed in the BlueCat Linux distribution in `$BLUECAT_PREFIX/demo/`. The provided demo directories can be built similar to the LynuxWorks BlueCat Linux demonstrations in the BlueCat Linux development environment. In the EDK project, there are two zipped demo directories. The `s3adsp_devl_kit_demo.zip` file is the demo directory to recreate the BlueCat Linux image that boots with a ramdisk file system. The `s3adsp_devl_kit_demo_flash.zip` file is the demo directory to recreate the BlueCat Linux image that boots with a JFFS2 file system.

Getting the MLD File Set

The MLD file set is included in the project directory at <project root directory>/bsp/linux_bc54_v1_00_a. This MLD fies set is for building BlueCat Linux images on a Linux host computer.

Generating the BSP

With the use of the BlueCat Linux MLD, EDK can update the BlueCat Linux kernel source tree to match a specific hardware configuration. Follow these steps to generate the BSP and update the BlueCat Linux kernel source tree.

1. Open the reference system in XPS.
2. Select **Software** → **Software Platform Settings...** under XPS.
3. In the Software Platform Settings window, select **linux_bc54** in the OS field, as shown in Figure 3-11.

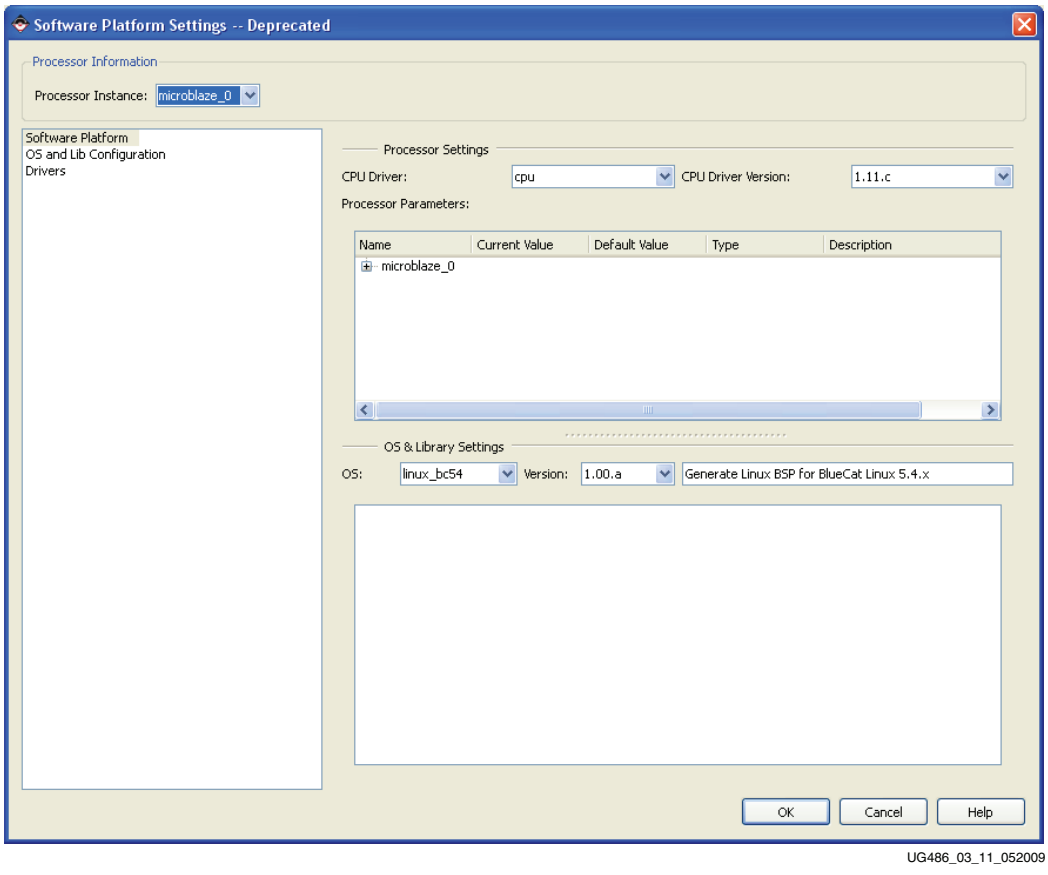


Figure 3-11: **Select BlueCat Linux for the OS**

4. Select the OS and Libraries option on the left of the Software Platform Settings window. Fill in the fields as follows:

   BLUECAT_PREFIX:

   <BlueCat Linux installation point>/usr/src/linux

   KERNEL_CONFIG:

   Ramdisk file system:

   <BlueCat_Linux_install_point>/demo/s3adsp_devl_kit_demo/s3adsp_devl_kit_demo.config

   Flash file system:

   <BlueCat_Linux_install_point>/demo/s3adsp_devl_kit_demo_flash/s3adsp_devl_kit_demo_flash.config

   An example showing these fields for the ramdisk file system is in Figure 3-12.

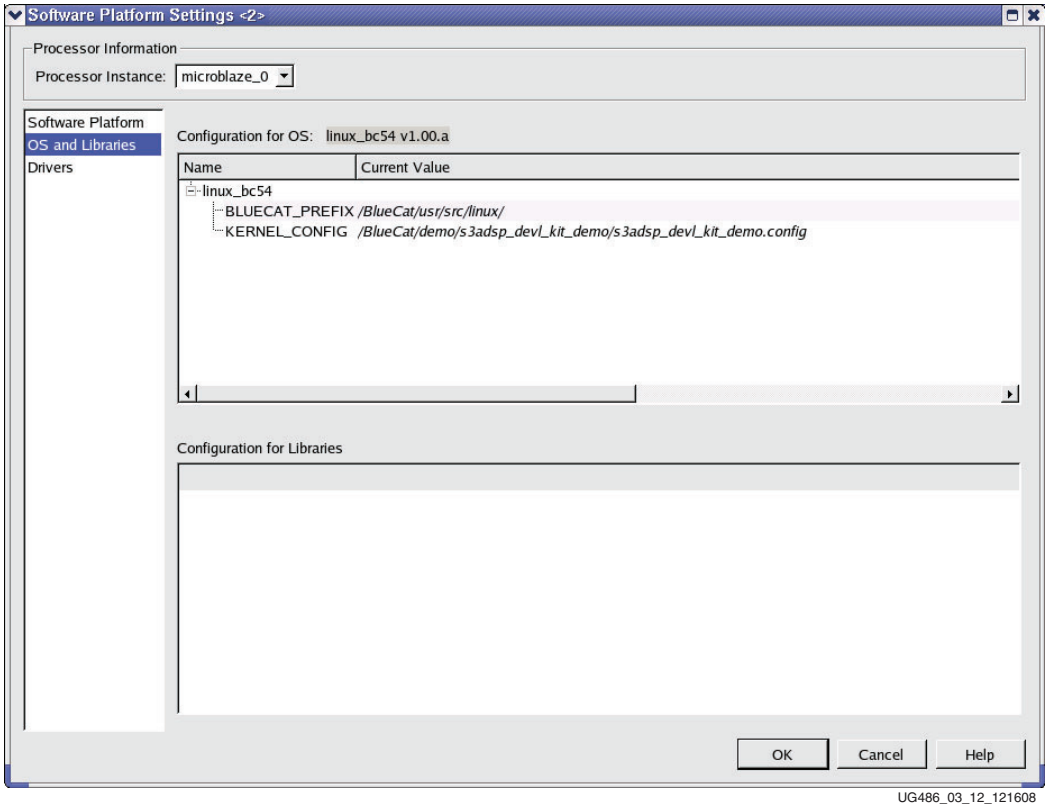


Figure 3-12: **Set the BlueCat Linux Paths**

Note: You must specify the full Linux path in these fields. You can not use "~" as a short cut for /home/<user>.

5. Click **OK** to save the changes and close the Software Platform Settings window.
6. In XPS, select **Software** → **Generate Libraries and BSPs**. This will update the BlueCat Linux kernel source tree.

Rebuilding the Kernel Image

This is the final step to create a bootable BlueCat Linux kernel image. To recreate the image provided with this reference system, follow the subsequent steps.

1. A patch is provided so that the Flash memory can be used to hold a file system. The patch is located in the `/bclinux_demo` directory in the project. The `cfi_patch` file must be copied to the BlueCat Linux installation location, $BLUECAT_PREFIX, then the patch can be applied.

   ```
   BlueCat:$ cd $BLUECAT_PREFIX
   BlueCat:$ cp <project root directory/bclinux_demo/cfi_patch .
   BlueCat:$ patch -p0 < cfi_patch
   ```

2. To force all kernel components to rebuild, clean the kernel tree by using the following commands:

   ```
   BlueCat:$ cd $BLUECAT_PREFIX/usr/src/linux
   BlueCat:$ make mrproper
   ```

3. Navigate to the appropriate demo directory.

   Ramdisk file system:

   ```
   BlueCat:$ cd $BLUECAT_PREFIX/demo/s3adsp_devl_kit_demo
   ```

   Flash file system:

   ```
   BlueCat:$ cd
   $BLUECAT_PREFIX/demo/s3adsp_devl_kit_demo_flash
   ```

4. Run the following command to see the menu for the Linux kernel configuration:

   ```
   BlueCat:$ make menuconfig
   ```

5. To rebuild the Linux image that uses the JFFS2 Flash file system, ensure that the kernel configuration has enabled support for JFFS2. Navigate the menu to **File Systems** → **Miscellaneous filesystems.** Select the menu item to include Journalling Flash File System v2 (JFFS2) support. Under the main menu, navigate to **Device Drivers** → **Memory Technology Devices (MTD).** Select the menu item to include Memory Technology Device (MTD) support. Under MTD support, select the menu item to include MTD partitioning support.
6. Under the Linux Kernel configuration menu, also make sure that support is enabled for **General setup** → **System V IPC**. The Apache Web server requires this support.
7. Exit the Linux kernel configuration menu, saving the new configuration if changes were made.
8. Clean any prebuilt image files.

   ```
   BlueCat:$ make clean
   ```

9. Build the kernel, root filesystem, and bootable image file.

   ```
   BlueCat:$ make all
   ```

   This command produces a .kdi file which is the BlueCat Linux image and is composed of a compressed kernel image and a compressed RAM disk root file system. The image will be stored in one of the following locations, depending on which demo was built.

   Ramdisk file system:

```
$BLUECAT_PREFIX/demo/s3adsp_devl_kit_demo/s3adsp_devl_kit_demo.kdi
```

Flash file system:

```
$BLUECAT_PREFIX/demo/s3adsp_devl_kit_demo_flash/s3adsp_devl_kit_demo_flash.kdi
```

10. To run the newly created kernel image, refer to the steps in the section "Executing the BlueCat Linux Images". When downloading the kernel image through XMD into DDR2 memory, put in the path to the new kernel image instead of the path to the pre-built kernel image in the `bclinux_images` directory.

Booting the BlueCat Linux Image from Parallel Flash

To boot the BlueCat Linux image from parallel Flash, the Linux image and a bootloader application must be programmed into Flash. The bootloader application copies the Linux image from Flash to DDR2 memory and boots BlueCat Linux. The steps in this section refer to XAPP1106: Using and Creating Flash Files for the MicroBlaze Development Kit - Spartan-3A DP 1800A Starter Platform, which includes details on how to use, create, and boot BPI Flash files for the MicroBlaze Spartan-3A DSP 1800A Edition Development Kit.

Flash files that have already been generated are provided, or the user can create new files for programming the Flash. Refer to Table 1 in XAPP1106 for an address map of the parallel Flash after programming the BlueCat Linux files.

Note: The options for bitgen in the <project root directory>/etc/bitgen.ut file need to be set to use CCLK for these steps.

Programming the Flash with the Provided Files

Flash files that have already been generated and are ready to use can be found in the `<project root directory>/ready_for_download/Flash_files/` directory. A bootloader, `bootloader_bclinux`, is also provided in the reference system for bootloading the BlueCat Linux image.

1. Follow the steps outlined in the section "Programming the Parallel Flash with the Provided BlueCat Linux Files" in XAPP1106. These steps will provide information on how to program the Flash with the Linux image.
2. Set the jumpers on the Spartan-3A DSP 1800A board to BPI mode (M0 and M2 closed). Apply power to the development board or press the PROG button if power has already been applied. This will boot the FPGA in BPI mode. The terminal output should be as in Figure 3-6.

Generating New Flash Files and Programming the Flash

Instead of using the pregenerated files, the user can generate new files for programming the Flash device. This section details the steps for creating new Flash files and programming them into the Flash device.

1. Follow the steps outlined in the section "Creating and Programming New Parallel Flash Files for BlueCat Linux" in XAPP1106. These steps will provide information on how to program the Flash with the Linux image.
2. Set the jumpers on the Spartan-3A DSP 1800A board to BPI mode (M0 and M2 closed) and press the PROG button. This will boot the FPGA in BPI mode. The terminal output should be as in Figure 3-6.

Chapter 4

FlashRWE Software Application

Introduction

The FlashRWE software application is a simple application that provides functions to read, write, and erase the parallel Flash device. When the application is run, it will print a menu with options to enter the read, write, or erase menus.

One method for downloading and running the FlashRWE software application is listed below:

- Use a debugger, such as XMD (provided as part of the EDK tools), to download the executable file directly into BRAM. This method is described in the section "Executing the FlashRWE Software Application".

Note: A warning box will appear during some of the steps in this chapter. The warning box states that "Software development features in XPS are deprecated, and will be removed in the next major release". Click OK to safely ignore this warning. To turn off this warning completely, navigate to Edit→Preferences in XPS. Select Application Preferences and check the box that states "Do not show "Software Features Deprecated" dialog box".

Executing the FlashRWE Software Application

To execute the FlashRWE software application, program the hardware bitstream to the Spartan-3A DSP device and load the FlashRWE software application into BRAM. Program the bitstream by downloading the pre-built bitstream from the `ready_for_download` directory or generate and download it from XPS. Similarly, the FlashRWE executable can be downloaded from the `ready_for_download` directory or built and downloaded through XPS.

Executing the FlashRWE Application Using the Pre-Built Bitstream

To execute the application using the files inside the `ready_for_download` directory in the project root directory, follow these steps:

1. Connect the Platform USB cable or the Parallel IV JTAG cable between the host computer and the Spartan-3A DSP 1800A development board.
2. Connect the serial cable between the host computer and the RS232 port on the Spartan-3A DSP 1800A development board.

3. Apply power to the Spartan-3A DSP 1800A development board.
4. Start a HyperTerminal (or similar) session on the host computer with the settings shown in Figure 4-1. Select the COM port corresponding to the connected serial port on the host computer. Set the Baud Rate to **115200**, Data bits to **8** bits, Parity to **None**, Stop bits to **1** bit, and Flow control to **None**.

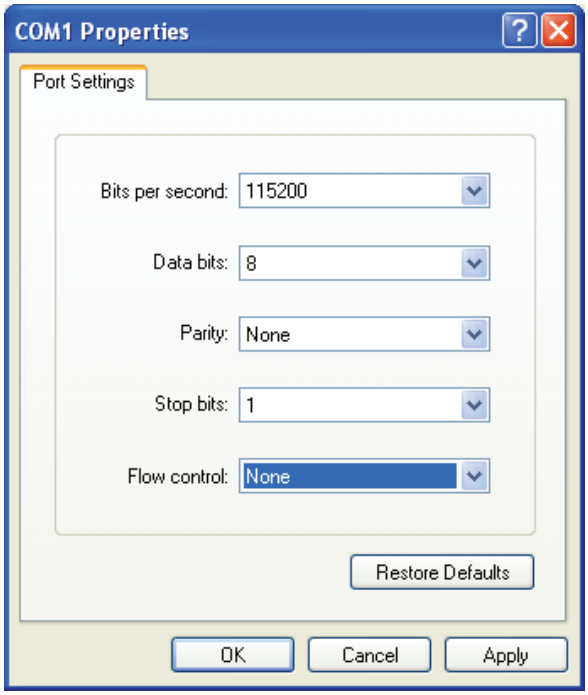


Figure 4-1: **HyperTerminal Settings**

5. In an EDK shell, change directories to the `ready_for_download` directory.
6. Use iMPACT to download the bitstream by using the following command:

   ```
   $ impact -batch ug486.cmd
   ```

7. Invoke XMD and connect to the processor by the following command:

   ```
   $ xmd -opt ug486.opt
   ```

8. Download the FlashRWE software application into BRAM using the following command:

   ```
   XMD% dow flashrwe_executable.elf
   ```

9. To start the FlashRWE software application running, use the following XMD command:

   ```
   XMD% run
   ```

   a. After the FlashRWE software application runs, the HyperTerminal will display the Main Menu, as shown in Figure 4-2, page 44.

   b. For an explanation of the available functionality in the application, see the section "Functions in the FlashRWE Software Application".

Executing the FlashRWE Software Application from XPS

To execute the reference system using XPS, follow these steps:

1. Perform steps 1-4 in the "Executing the FlashRWE Application Using the Pre-Built Bitstream" section.
2. Open either the reference system project in XPS.
3. Implement the hardware design and create the hardware bitstream by selecting **Hardware** → **Generate Bitstream** in XPS.
4. In the Applications tab, build the **FlashRWE** project by right-clicking on the project and selecting **Build Project**. This will create the software executable for the application.
5. Download the bitstream to the board by selecting **Device Configuration** → **Download Bitstream** in XPS.
6. After the bitstream has downloaded, launch the XMD by selecting **Debug** → **Launch XMD...** in XPS.

7. Download the FlashRWE application executable using the following command in XMD:

   ```
   XMD% dow FlashRWE/executable.elf
   ```

8. To run the software application, use the **run** command in XMD.
   a. After the FlashRWE software application runs, the HyperTerminal will display the Main Menu, as shown in Figure 4-2, page 44.
   b. For an explanation of the available commands in the application, see the section "Functions in the FlashRWE Software Application".

Functions in the FlashRWE Software Application

After the FlashRWE application is executed, the Main Menu will be printed out to the terminal application. The Main Menu is shown in Figure 4-2. The Main Menu allows the options to enter into the read, write, and erase submenus.

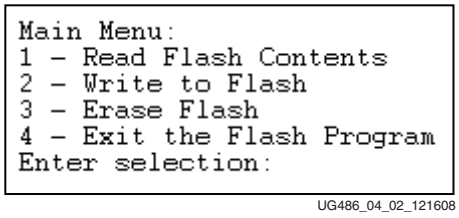


Figure 4-2: **Main Menu**

Table 4-1 lists and describes the functions that are available in the FlashRWE application.

Table 4-1: **Summary of the Available Functions in the FlashRWE Application**

Submenu	Function	Description	User Inputted Parameters	Document Link
Flash Read	Read Bytes of Flash	Read and print out individual bytes of data from the parallel Flash device.	Address offset, Number of bytes	"Read Bytes of Flash," page 46
	Check if Flash is Empty	Checks all bytes of data in the parallel Flash device and determines if the Flash device is fully erased.	None	"Check if Flash is Empty," page 46
	Exit to Main Menu	Exit the submenu and return to the main menu	None	N/A

Table 4-1: **Summary of the Available Functions in the FlashRWE Application**

Submenu	Function	Description	User Inputted Parameters	Document Link
Flash Write	Write Incrementing Numbers to Flash	Write incrementing numbers, 0x0-0xFF to bytes in the Flash.	Address offset, Number of bytes	"Write Incrementing Numbers to Flash," page 48
	Write Bytes to Flash	Write bytes of data to Flash	Address offset, Bytes of data, Number of times to write the data	"Write Bytes to Flash," page 49
	Exit to Main Menu	Exit the submenu and return to the main menu	None	N/A
Flash Erase	Erase Bytes of Flash	Erases the block(s) of Flash that the specified bytes reside in	Address offset, Number of bytes	"Erase Bytes of Flash," page 51
	Erase Blocks of Flash	Erases a single block or a range of consecutive blocks of Flash	Starting block, Ending block	"Erase Blocks of Flash," page 51
	Erase the Entire Flash	Erases the entire Flash device	None	"Erase the Entire Flash," page 52
	Exit to Main Menu	Exit the submenu and return to the main menu	None	N/A
None (Main menu)	Exit the Flash Program	Quits the FlashRWE program	None	N/A

Flash Read Menu

The Flash Read Menu is displayed after entering 1 at the Main Menu prompt. The Flash Read Menu has options to read bytes of Flash, check if the Flash is empty, or exit back to the Main Menu. The Flash Read Menu is shown in Figure 4-3.

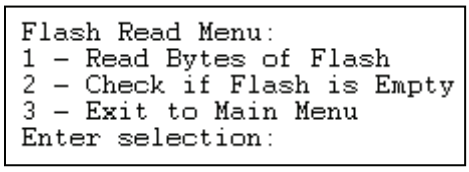


Figure 4-3: **Flash Read Menu**

Read Bytes of Flash

The Read Bytes of Flash function can be executed by entering 1 at the Flash Read Menu prompt. Once executed, this function reads individual bytes of data out of the Flash device and displays the address and data.

The following are the parameters the user must supply values for:

- Address offset: Offset into the Flash from which to start reading the first byte of data
- Number of bytes: Total number of bytes to read and display data

Example input and output from this function is shown in Figure 4-4. In the figure, 50 bytes of data is read starting from the address `0x87100000`.

```
Flash Read Menu:
1 - Read Bytes of Flash
2 - Check if Flash is Empty
3 - Exit to Main Menu
Enter selection: 1
Enter address offset (0-FFFFFF): 100000
Enter number of bytes to read (0-128): 50

0x87100000: 00 01 02 03 04 05 06 07
0x87100008: 08 09 0A 0B 0C 0D 0E 0F
0x87100010: 10 11 12 13 14 15 16 17
0x87100018: 18 19 1A 1B 1C 1D 1E 1F
0x87100020: 20 21 22 23 24 25 26 27
0x87100028: 28 29 2A 2B 2C 2D 2E 2F
0x87100030: 30 31
```

UG486_04_04_121608

Figure 4-4: **Read Bytes of Flash Example Input/Output**

Check if Flash is Empty

The Check if Flash is Empty function can be executed by entering 2 at the Flash Read Menu prompt. Once executed, this function checks if the entire Flash device is completely erased and empty. This function reads all the individual bytes of data in the Flash device and compares the data to `0xFF`. If all bytes of data are `0xFF`, the Flash device is completely erased. Otherwise, a set of the locations that were not empty and the data in those locations is outputted to the terminal. The total number of locations found that were not empty is also outputted. This function takes approximately 22 seconds to run.

Figure 4-5 shows example output from the function when the Flash is empty. Figure 4-6 shows example output from the function when the Flash is not empty.

```
Flash Read Menu:
1 - Read Bytes of Flash
2 - Check if Flash is Empty
3 - Exit to Main Menu
Enter selection: 2
Checking if the Flash is empty...
................................
Flash is completely empty
```

UG486_04_05_121608

Figure 4-5: **Check if Flash is Empty Example Output - Flash is Empty**

```
Flash Read Menu:
1 - Read Bytes of Flash
2 - Check if Flash is Empty
3 - Exit to Main Menu
Enter selection: 2
Checking if the Flash is empty...
.Flash is not erased at 0x87003000; Data is 0
Flash is not erased at 0x87003001; Data is 1
Flash is not erased at 0x87003002; Data is 2
Flash is not erased at 0x87003003; Data is 3
Flash is not erased at 0x87003004; Data is 4
Flash is not erased at 0x87050000; Data is FE
Flash is not erased at 0x87050001; Data is ED
Flash is not erased at 0x87050002; Data is FE
Flash is not erased at 0x87050003; Data is ED
.Flash is not erased at 0x87100000; Data is 0
Flash is not erased at 0x87100001; Data is 1
Flash is not erased at 0x87100002; Data is 2
Flash is not erased at 0x87100003; Data is 3
Flash is not erased at 0x87100004; Data is 4
Flash is not erased at 0x87100005; Data is 5
Flash is not erased at 0x87100006; Data is 6
Flash is not erased at 0x87100007; Data is 7
Flash is not erased at 0x87100008; Data is 8
Flash is not erased at 0x87100009; Data is 9
Flash is not erased at 0x8710000A; Data is A
Flash is not erased at 0x8710000B; Data is B
Flash is not erased at 0x8710000C; Data is C
Flash is not erased at 0x8710000D; Data is D
Flash is not erased at 0x8710000E; Data is E
Flash is not erased at 0x8710000F; Data is F
............................
Flash is not completely empty
508 total bytes of Flash were found to not be empty
The first 25 locations of Flash that were not empty were printed out
```

UG486_04_06_121608

Figure 4-6: **Check if Flash is Empty Example Output - Flash is not Empty**

Flash Write Menu

The Flash Write Menu is displayed after entering 2 at the Main Menu prompt. The Flash Write Menu has options to write incrementing numbers to Flash, write bytes to Flash, or exit back to the Main Menu. The Flash Write Menu is shown in Figure 4-7.

```
Flash Write Menu:
1 - Write Incrementing Numbers to Flash
2 - Write Bytes to Flash
3 - Exit to the Main Menu
Enter selection:
```

UG486_04_07_121608

Figure 4-7: **Flash Write Menu**

Write Incrementing Numbers to Flash

The Write Incrementing Numbers to Flash function can be executed by entering 1 at the Flash Write Menu prompt. Once executed, this function writes incrementing numbers (0x0-0xFF) to bytes in the Flash. After reaching 0xFF, the function wraps back to 0x0 and continues writing incrementing numbers to bytes in the Flash. If there is data found in the Flash at the bytes specified for writing, the function will prompt the user to either erase the block(s) of Flash where the bytes reside, or to cancel the write request.

The following are the parameters the user must supply values for:

- Address offset: Offset into the Flash from which to start writing the first byte of data
- Number of bytes: Total number of bytes to write incrementing numbers

Figure 4-8 shows example output from the function when the bytes to write are already empty. In Figure 4-8, 112 bytes of data is being written to address 0x87CC0000. Figure 4-9 shows example output from the function when the bytes to write have previously stored data. In Figure 4-9, 38 bytes of data were input to be written to address 0x87003000, but 5 bytes of Flash were found to have previous data. To proceed with the write, the user must allow the entire block (0x87000000-0x8701FFFF) to be erased.

```
Flash Write Menu:
1 - Write Incrementing Numbers to Flash
2 - Write Bytes to Flash
3 - Exit to the Main Menu
Enter selection: 1
Enter address offset (0-FFFFFF): CC0000
Enter number of bytes to write (0-2048): 112

Completed writing incrementing numbers to Flash
```

UG486_04_08_121608

Figure 4-8: **Write Incrementing Numbers Example Input and Output - Empty Bytes**

```
Flash Write Menu:
1 - Write Incrementing Numbers to Flash
2 - Write Bytes to Flash
3 - Exit to the Main Menu
Enter selection: 1
Enter address offset (0-FFFFFF): 3000
Enter number of bytes to write (0-2048): 38
In the desired write location, 5 bytes of Flash were found to already have data
1 - Erase the block(s) that contain the data and proceed with the write
2 - Cancel the write request and keep the original data
Enter selection: 1

Completed writing incrementing numbers to Flash
```

UG486_04_09_121608

Figure 4-9: **Write Incrementing Numbers Example Input and Output - Full Bytes**

Write Bytes to Flash

The Write Bytes to Flash function can be executed by entering 2 at the Flash Write Menu prompt. Once executed, this function writes bytes of data to Flash. If there is data found in the Flash at the bytes specified for writing, the function will prompt the user to either erase the block(s) of Flash where the bytes reside, or to cancel the write request.

The following are the parameters the user must supply values for:

- Address offset: Offset into the Flash from which to start writing the first byte of data
- Bytes of data: The actual data to write to bytes in Flash
- Number of times: The number of times to write the specified bytes of data

Figure 4-10 shows example input and output from the function when the bytes to write are already empty. In Figure 4-10, the data string `0x9078563412` is being written 5 times to address `0x87400000`. Figure 4-11 shows example input and output from the function when the bytes to write have previously stored data. In Figure 4-11, the data string `0xABBCCDDEEFFA` was input to be written 28 times to the address `0x87E20000`, but 21 bytes of Flash were found to have previous data. To proceed with the write, the user must allow the entire block (`0x87E20000-0x87E3FFFF`) to be erased.

```
Flash Write Menu:
1 - Write Incrementing Numbers to Flash
2 - Write Bytes to Flash
3 - Exit to the Main Menu
Enter selection: 2
Enter address offset (0-FFFFFF): 400000
Enter 0-16 bytes of data to write: 9078563412
Enter number of times to write the data (0-32): 5

Completed writing bytes of data to Flash
```

UG486_04_10_121608

Figure 4-10: **Write Bytes Example Input and Output - Empty Bytes**

```
Flash Write Menu:
1 - Write Incrementing Numbers to Flash
2 - Write Bytes to Flash
3 - Exit to the Main Menu
Enter selection: 2
Enter address offset (0-FFFFFF): E20000
Enter 0-16 bytes of data to write: ABBCCDDEEFFA
Enter number of times to write the data (0-32): 28
In the desired write location, 21 bytes of Flash were found to already have data
1 - Erase the block(s) that contain the data and proceed with the write
2 - Cancel the write request and keep the original data
Enter selection: 1

Completed writing bytes of data to Flash
```

UG486_04_11_121608

Figure 4-11: **Write Bytes Example Input and Output - Full Bytes**

Flash Erase Menu

The Flash Erase Menu is displayed after entering 3 at the Main Menu prompt. The Flash Erase Menu has options to erase bytes of Flash, erase blocks of Flash, erase the entire Flash, or exit back to the Main Menu. The Flash Erase Menu is shown in Figure 4-12.

```
Flash Erase Menu:
1 - Erase Bytes of Flash
2 - Erase Blocks of Flash
3 - Erase the Entire Flash
4 - Exit to Main Menu
Enter selection:
```

UG486_04_12_121608

Figure 4-12: **Flash Erase Menu**

Erase Bytes of Flash

The Erase Bytes of Flash function can be executed by entering 1 at the Flash Erase Menu prompt. Once executed, this function erases the block(s) of Flash that the specified bytes reside in.

The following are the parameters the user must supply values for:

- Address offset: Offset into the Flash to the first byte desired to be erased
- Number of bytes: Total number of bytes to erase

  Caution! Because Flash memory must be erased in blocks, the number of bytes actually erased may be larger than the inputted number of bytes to erase, depending on the address offset and number of bytes specified.

Example input and output from this function is shown in Figure 4-13. In the figure, it is specified to erase 400 bytes at address `0x8704A000`. However, because of Flash memory characteristics, the entire block that these bytes reside in will be erased, which will be the address range `0x87040000-0x8705FFFF`.

```
Flash Erase Menu:
1 - Erase Bytes of Flash
2 - Erase Blocks of Flash
3 - Erase the Entire Flash
4 - Exit to Main Menu
Enter selection: 1
Enter address offset (0-FFFFFF): 4A000
Enter number of bytes to erase (0-393216): 400
Erased the Flash memory contents successfully
```

UG486_04_13_121608

Figure 4-13: **Erase Bytes of Flash Example Input and Output**

Erase Blocks of Flash

The Erase Blocks of Flash function can be executed by entering 2 at the Flash Erase Menu prompt. Once executed, this function erases the specified range of block(s) of Flash. If the maximum number of blocks (128 blocks) is specified to be erased, this function will take approximately 2 minutes to complete.

The following are the parameters the user must supply values for:

- Starting block: First block in the range of blocks to be erased
- Ending block: Last block in the range of blocks to be erased

Example input and output from this function is shown in Figure 4-14. In the figure, blocks 103-127 have been erased, which corresponds to the address range `0x87CE000000-0x87FFFFFF`.

```
Flash Erase Menu:
1 - Erase Bytes of Flash
2 - Erase Blocks of Flash
3 - Erase the Entire Flash
4 - Exit to Main Menu
Enter selection: 2
Enter starting block (0-127) to erase: 103
Enter ending block (103-127) to erase: 127
Erasing blocks 103-127 of Flash...
........
Erased the Flash memory contents successfully
```

UG486_04_14_121608

Figure 4-14: **Erase Blocks of Flash Example Input and Output**

Erase the Entire Flash

The Erase the Entire Flash function can be executed by entering 3 at the Flash Erase Menu prompt. Once executed, this function erases the entire Flash device. The function will take approximately 2 minutes to complete.

Example output from this function is shown in Figure 4-15.

```
Flash Erase Menu:
1 - Erase Bytes of Flash
2 - Erase Blocks of Flash
3 - Erase the Entire Flash
4 - Exit to Main Menu
Enter selection: 3
Erasing the entire Flash...
..................................
Erased the Flash memory contents successfully
```

UG486_04_15_121608

Figure 4-15: **Erase the Entire Flash Example Output**