

Virtex-5 FXT PowerPC 440 and MicroBlaze Edition Kit Reference Systems

UG511 (v1.2.1) July 30, 2009





Xilinx is disclosing this user guide, manual, release note, and/or specification (the "Documentation") to you solely for use in the development of designs to operate with Xilinx hardware devices. You may not reproduce, distribute, republish, download, display, post, or transmit the Documentation in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Xilinx expressly disclaims any liability arising out of your use of the Documentation. Xilinx reserves the right, at its sole discretion, to change the Documentation without notice at any time. Xilinx assumes no obligation to correct any errors contained in the Documentation, or to advise you of any corrections or updates. Xilinx expressly disclaims any liability in connection with technical support or assistance that may be provided to you in connection with the Information.

THE DOCUMENTATION IS DISCLOSED TO YOU "AS-IS" WITH NO WARRANTY OF ANY KIND. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DOCUMENTATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT OF THIRD-PARTY RIGHTS. IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOSS OF DATA OR LOST PROFITS, ARISING FROM YOUR USE OF THE DOCUMENTATION.

© 2008-2009 Xilinx, Inc. All rights reserved.

XILINX, the Xilinx logo, the Brand Window, and other designated brands included herein are trademarks of Xilinx, Inc. The PowerPC name and logo are registered trademarks of IBM Corp. and are used under license. All other trademarks are the property of their respective owners.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
7/29/08	1.0	Initial Xilinx release.
12/19/08	1.1	Updated to 10.1.3. Added content on Flash file system. Updated steps on programming parallel Flash. Added chapter on software application to read, write, and erase Flash.
1/26/09	1.1.1	Updated reference system links.
5/21/09	1.2	Updated to 11.1. Added steps for applying the xlitemac patch. Updated section on the MLD file set to use the 11.1 file set included in the project.
7/30/09	1.2.1	Updated ref system CID numbers; updated to current date.

Table of Contents

Preface: About This Guide

Guide Contents	7
Hardware and Software Requirements	7
References	8
Additional Resources	8
Conventions	8
Typographical	8
Online Document	9

Chapter 1: Hardware Platform

Introduction	11
PowerPC 440 Processor Reference System	11
Block Diagram	11
Address Map	12
System Configuration	12
MicroBlaze Processor Reference System	13
Block Diagram	13
Address Map	13
System Configuration	14

Chapter 2: HelloWorld Software Application

Introduction	15
Executing the HelloWorld Software Application	15
Executing the HelloWorld Application Using the Pre-Built Bitstream	15
Executing the HelloWorld Software Application from XPS	18
Commands in the HelloWorld Software Application	19
Bootting the HelloWorld Application from Serial Flash	20

Chapter 3: LynuxWorks BlueCat Linux

Introduction	23
Executing the BlueCat Linux Images	23
Executing the BlueCat Linux Image with a Ramdisk File System	23
Executing the BlueCat Linux Image Using the Pre-Built Bitstream	24
Executing the BlueCat Linux Image from XPS	26
Executing the BlueCat Linux Image with a JFFS2 File System	26
Executing the BlueCat Linux Image Using the Pre-Built Bitstream	27
Executing the BlueCat Linux Image from XPS	32
Executing BlueCat Linux Commands	35
Web Server Demonstration	36
Building the BlueCat Linux Kernel Image	37
Installing the BlueCat Linux Distribution	37
Using the Provided Demo Directories	37

Getting the MLD File Set	38
Generating the BSP	38
Rebuilding the Kernel Image	39
Booting the BlueCat Linux Image from Parallel Flash	41
Programming the Flash with the Provided Files	41
Programming the KDI File	41
Programming the BIN File	43
Programming the Root File System	44
Running the Design	45
Generating New Flash Files and Programming the Flash	46
Programming the KDI File and Creating the Bootloader	46
Creating and Programming the BIN File	49
Programming the Root File System	50
Running the Design	52

Chapter 4: FlashRWE Software Application

Introduction	53
Executing the FlashRWE Software Application	53
Executing the FlashRWE Application Using the Pre-Built Bitstream	53
Executing the FlashRWE Software Application from XPS	55
Functions in the FlashRWE Software Application	55
Flash Read Menu	57
Read Bytes of Flash	57
Check if Flash is Empty	57
Flash Write Menu	59
Write Incrementing Numbers to Flash	59
Write Bytes to Flash	60
Flash Erase Menu	61
Erase Bytes of Flash	61
Erase Blocks of Flash	62
Erase the Entire Flash	63

About This Guide

The Embedded Development HW/SW Kit - Virtex®-5 FXT PowerPC® 440 and MicroBlaze™ Processor Edition showcases various features of the Virtex-5 FXT ML507 development board. This kit includes two hardware systems with a HelloWorld software application, bootable BlueCat Linux image, and a Flash reading, writing, and erasing application. This document describes the hardware platform, the HelloWorld software application, the BlueCat Linux images, and the Flash software application.

The reference systems are available at:

MicroBlaze system:

<https://secure.xilinx.com/webreg/clickthrough.do?cid=135330>

PowerPC 440 system:

<https://secure.xilinx.com/webreg/clickthrough.do?cid=135331>

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 images 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 ML507 Development Board
- Xilinx Platform USB Download Cable or Parallel IV Download Cable
- RS232 Serial Cable
- Ethernet Cable
- Serial Communications Utility Program (e.g. 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. [UG083](#) Getting Started Tutorial for ML401/ML402/ML403/ML405 Evaluation Platforms
4. [UG200](#) Embedded Processor Block in Virtex-5 FPGAs

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 <i>Development System Reference Guide</i> for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are <i>not</i> 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 [<i>option_name</i>] <i>design_name</i>
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 <i>block_name loc1</i> <i>loc2 ... locn</i> ;

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 <i>Virtex-II Platform FPGA User Guide</i> .
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest speed files.

Hardware Platform

Introduction

The Virtex-5 FXT Development Kit includes two reference systems that target the ML507 development board. One reference system is based on the PowerPC 440 (PPC440) processor and the other reference system is based on the MicroBlaze processor. Both the PPC440 processor reference system and the MicroBlaze processor reference system are created to run BlueCat Linux. The BlueCat Linux images are described in [Chapter 3, “LinuxWorks BlueCat Linux”](#). The HelloWorld software application described in [Chapter 2, “HelloWorld Software Application,”](#) and the FlashRWE software application described in [Chapter 4, “FlashRWE Software Application”](#) can be run on either the MicroBlaze processor reference system or the PPC440 processor reference system. Both the PowerPC and MicroBlaze processor systems are described in this chapter.

PowerPC 440 Processor Reference System

Block Diagram

The block diagram for the PowerPC 440 processor reference system is shown in [Figure 1-1](#).

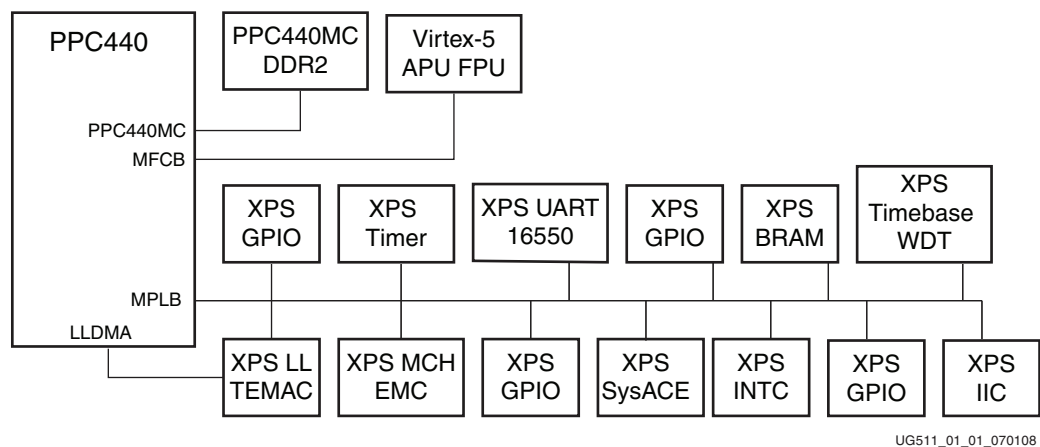


Figure 1-1: PowerPC 440 Processor System Block Diagram

Address Map

The address map for the IP cores in the PowerPC 440 processor reference system is given in [Table 1-1](#).

Table 1-1: PowerPC 440 Processor Reference System Address Map

Instance	Peripheral	Base Address	High Address
xps_bram_if_cntlr_1	xps_bram	0xFFFFF0000	0xFFFFFFFF
LEDs_8Bit	xps_gpio	0x81400000	0x8140FFFF
LEDs_Positions	xps_gpio	0x81420000	0x8142FFFF
Push_Buttons_5Bit	xps_gpio	0x81440000	0x8144FFFF
DIP_Switches_8Bit	xps_gpio	0x81460000	0x8146FFFF
xps_timebase_wdt_1	xps_timebase_wdt	0x83A00000	0x83A0FFFF
xps_intc_0	xps_intc	0x81800000	0x8180FFFF
IIC_EEPROM	xps_iic	0x81600000	0x8160FFFF
Hard_Ethernet_MAC	xps_ll_temac	0x81C00000	0x81C0FFFF
SysACE_CompactFlash	xps_sysace	0x83600000	0x8360FFFF
xps_timer_1	xps_timer	0x83C00000	0x83C0FFFF
RS232_Uart_1	xps_uart16550	0x83E00000	0x83E0FFFF
DDR2_SDRAM	ppc440mc_ddr2	0x00000000	0xFFFFFFFF
FLASH	xps_mch_emc	0xFC000000	0xFDFFFFFF

System Configuration

The PPC440 reference system uses the PowerPC 440 processor block with a processor frequency of 400 MHz. The Memory Interface Block of the processor block is connected to the PPC440MC DDR2 memory controller and is set to operate at a frequency of 200 MHz. The PLB v4.6 bus is connected to the MPLB port of the processor block, which allows the XPS peripherals to be connected as slaves on the bus. The PLB v4.6 bus frequency is 100 MHz.

More information about the PowerPC 440 processor and the embedded processor block can be found in UG200, *Embedded Processor Block in Virtex-5 FPGAs*.

The LocalLink connection of the XPS LL TEMAC core is connected to the Hard DMA device on the processor block. In the BlueCat Linux demonstration, the Ethernet MAC can run at 10 Mb/s, 100 Mb/s, or 1000 Mb/s, depending on the attached network.

MicroBlaze Processor Reference System

Block Diagram

The block diagram for the MicroBlaze processor reference system is shown in [Figure 1-2](#).

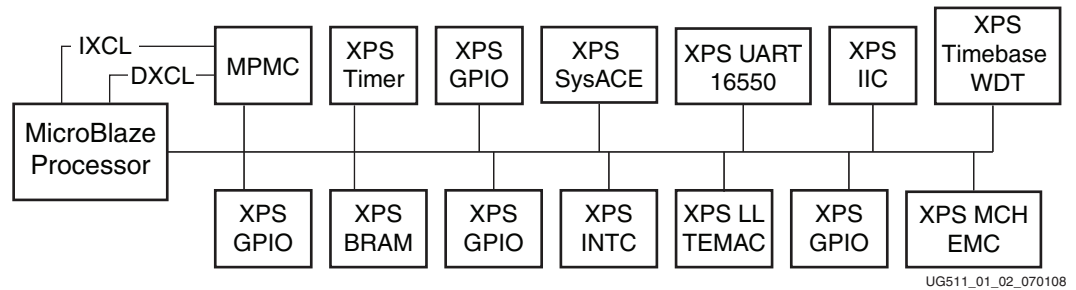


Figure 1-2: MicroBlaze Processor System Block Diagram

Address Map

The address map for the IP cores in the MicroBlaze processor reference system is given in [Table 1-2](#).

Table 1-2: MicroBlaze Processor 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	0x88310000	0x8831FFFF
LEDs_8Bit	xps_gpio	0x81400000	0x8140FFFF
LEDs_Positions	xps_gpio	0x81420000	0x8142FFFF
Push_Buttons_5Bit	xps_gpio	0x81440000	0x8144FFFF
DIP_Switches_8Bit	xps_gpio	0x81460000	0x8146FFFF
xps_timebase_wdt_1	xps_timebase_wdt	0x83A00000	0x83A0FFFF
xps_intc_0	xps_intc	0x81800000	0x8180FFFF
IIC_EEPROM	xps_iic	0x81600000	0x8160FFFF
Hard_Ethernet_MAC	xps_ll_temac	0x81C00000	0x81C0FFFF
SysACE_CompactFlash	xps_sysace	0x83600000	0x8360FFFF
xps_timer_1	xps_timer	0x83C00000	0x83C0FFFF
RS232_Uart_1	xps_uart16550	0x83E00000	0x83E0FFFF
FLASH	xps_mch_emc	0x8C000000	0x8DFFFFFF
DDR2_SDRAM	mpmc	0x90000000	0x9FFFFFFF
DDR2_SDRAM(SDMA)	mpmc	0x84600000	0x8460FFFF

System Configuration

The MicroBlaze processor reference system runs off a reference clock frequency of 100 MHz from the oscillator on the board. The PLBv46 bus and the MicroBlaze processor run at a frequency of 100 MHz and the DDR2 runs at 200 MHz in this system.

The MicroBlaze processor is configured with the Memory Management Unit (MMU) enabled. The MMU is enabled and is implemented in Virtual mode by setting the MicroBlaze processor parameter C_USE_MMU to 3. 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 of the MicroBlaze processor are both enabled. The cacheable block of main memory is accessed via the XCL Port Interface Modules (PIM) of the Multi-Port Memory Controller (MPMC).

The MicroBlaze processor system uses the XPS LL TEMAC FPGA with the Virtex-5 Hard TEMAC FPGA to provide Ethernet functionality. The Ethernet MAC can run at 10 Mb/s, 100 Mb/s, or 1000 Mb/s, depending on the attached network.

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

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. The user can then select from a list of menu options, including options to select a target memory and read/write an address with necessary data. The HelloWorld software application can run on either the MicroBlaze or the PowerPC 440 processor reference system.

Two 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. 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, program the hardware bitstream to the Virtex-5 FX device and load the HelloWorld 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 HelloWorld executable can be downloaded from the `ready_for_download` directory or built and downloaded through XPS.

Executing the HelloWorld 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 Virtex-5 FX70T ML507 development board.

2. Connect the serial cable between the host computer and the RS232 port on the Virtex-5 FX70T ML507 development board.
3. Apply power to the Virtex-5 FX70T ML507 development 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**.

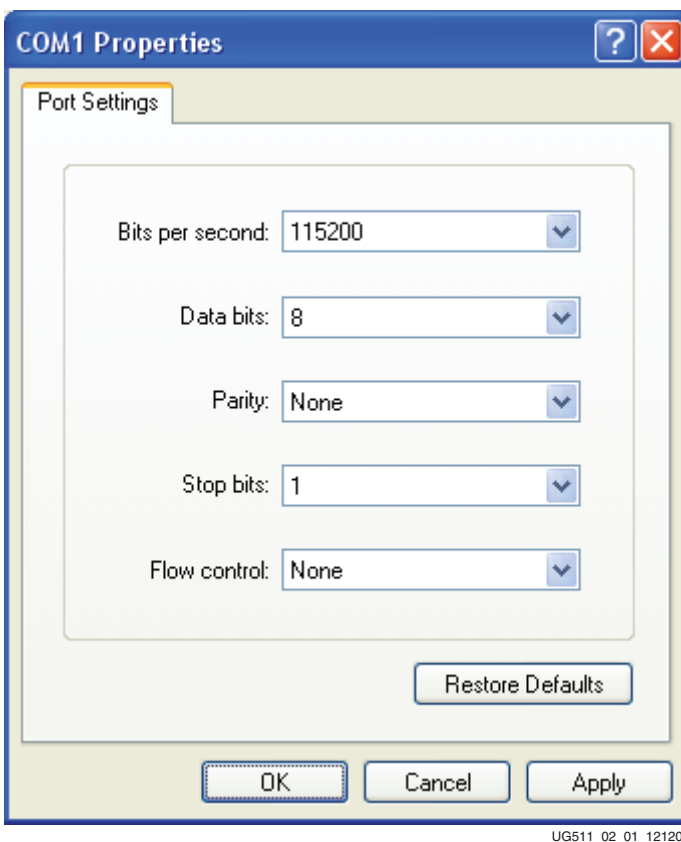


Figure 2-1: HyperTerminal Settings

5. 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 in either the MicroBlaze or PowerPC processor reference system.
7. Use iMPACT to download the bitstream by using the following command:

```
$ impact -batch ug511.cmd
```
8. Invoke XMD and connect to the processor by the following command:

```
$ xmd -opt ug511.opt
```
9. Download the HelloWorld software application into BRAM using the following command:

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


10. 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 Virtex-5 FXT ML507 Development Kit      **
*****
*****

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

LEDs test PASSED.

Writing pseudo random data at address... 0x20FFC
Reading pseudo random data at address... 0x20FFC
Memory Test PASSED!

Press any key to continue...

Type <Menu> for options

->
```

UG511_02_02_121208

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 either the MicroBlaze or PowerPC 440 processor 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 **helloworld** 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 HelloWorld application executable using the following command in XMD:
`dow helloworld/executable.elf`
8. To run the software application, use the **run** command in XMD.
 - 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](#).

Note: The Menu command options are case sensitive.

```

Type <Menu> for options
->Menu

Xilinx Virtex-5 ML507 Development kit Demo Menu!

Mem          Test DDR2 SDRAM
Led           Test the LEDs
PBT           Test the push buttons
Flash         Test parallel Flash memory
Test          Perform all factory tests
mwr <addr><# bytes><data> Write DDR2 mem locations with given data
mrd <addr><# bytes>      Read # of DDR2 mem locations and print data
Menu          Display Menu options
cls           Clear screen
q             Quit

Type <Menu> for options
->

```

UG511_02_03_121208

Figure 2-3: HelloWorld Menu

[Table 2-1](#) lists and describes the commands that are available in the HelloWorld application.

Table 2-1: Description of the HelloWorld Commands

Command	Description
Mem	The Mem test performs a destructive 32-bit wide memory test on a 132K byte block of the DDR2 SDRAM memory. This test erases, writes, reads, and verifies the DDR2 memory in the Virtex-5 FX70T ML507 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 instructs the user to push the West, South, East, North, and Center buttons and see the specified LEDs glow.

Table 2-1: Description of the HelloWorld Commands (Cont'd)

Command	Description
Flash	This test unlocks, erases, writes, reads, and verifies the Parallel Flash memory in the Virtex-5 FX70T ML507 development board. The results of the test will be displayed in the HyperTerminal.
Test	This performs all the factory tests mentioned above for the Virtex-5 FX70T ML507 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 serial Flash. These steps includes details on how to use, create, and boot serial Flash files for the Virtex-5 FX70T ML507 Development Kit.

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. Open the reference system project in XPS.
2. Disconnect the cable attached to header J1 (the header on the left side of the board) from the Xilinx download cable.
3. Connect JTAG flying wires from the Xilinx download cable to the J2 header using the pin labels as a guide on how to make the connections. The 7-pin J2 header is located to the right of the FPGA and just above the LCD panel.
4. Set the configuration address DIP switches to 00010101 (bits 4, 6, and 8 ON).
5. Remove the inserted ML507 CF card if present, and press the **Prog** button to erase the FPGA.
6. In XPS, compile a bitstream, download.bit, that includes the system configuration and the HelloWorld application. Compile the bitstream by marking the HelloWorld application to **Initialize BRAMs**, then selecting **Device Configuration** → **Update Bitstream** in XPS.
7. Copy the hardware bitstream from <project root directory>/implementation/download.bit to the <project root directory>/ready_for_download/Flash_files/ directory and rename it helloworld.bit to replace the current helloworld.bit file in the directory.
8. Format the BIT file to an MCS file using the cmd file **impact -batch convert_bits_to_mcs.cmd** from the ready_for_download/flash_files directory.

9. Launch iMPACT, then double-click **Direct SPI Configuration** in the iMPACT Flows window to program the SPI Flash device.
10. Right-click the Direct SPI Configuration tab, then select **Add SPI Device...**
11. Navigate to the <project root directory>/ready_for_download/flash_files/helloworld.mcs created above, then click Open.
12. In the Select Device Part Name drop-down dialog box, select **M25P32**, then click **OK**.
13. Click **OK** in the Device Programming Properties box.
14. The Direct SPI Configuration tabbed window displays a diagram of a single SPI PROM. Right-click on the SPI PROM, then select **Program**.
15. On the board, change the configuration address / mode DIP switches to **00000101** (bits 6 and 8 ON).
16. Press the **Prog** button. The design takes about 10 seconds to finish loading and begin to run. The serial output is shown in [Figure 2-3](#).

LynuxWorks BlueCat Linux

Introduction

BlueCat Linux can be targeted to run on the PowerPC 440 processor or the MicroBlaze soft processor with the MMU enabled. Example BlueCat Linux images are provided that are tailored to the Virtex-5 FXT ML507 development board and the hardware systems that are described in [Chapter 1, “Hardware Platform.”](#) Two BlueCat Linux images are provided with each hardware system, one that boots with a ramdisk root file system and one that uses a Journalling Flash File System, version 2 (JFFS2). The Virtex-5 FXT 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 demonstrations are listed below.

- Use a debugger, such as XMD (provided as part of the EDK tools), to download the image file directly into DDR2. 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 configuration mode switches to Platform Flash 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 each 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. One 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 image, the hardware bitstream must be programmed to the Virtex-5 FXT 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 system root directory, follow these steps:

1. Connect the Platform USB cable or the Parallel IV JTAG cable between the host computer and the Virtex-5 FXT ML507 development board.
2. Connect the serial cable between the host computer and the RS232 port on the Virtex-5 FXT ML507 development board.
3. Apply power to the Virtex-5 FXT ML507 development board.
4. Start a HyperTerminal (or similar) session on the host computer. 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**.
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 ug511.cmd
```

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

```
$ xmd -opt ug511.opt
```

9. Download the BlueCat Linux kernel image into DDR2 memory using one of the following commands, depending on the system.

PPC440 System:

```
XMD% dow -data ../bclinux_images/v5fxt_dev1_kit_demo.kdi
0x03000000
```

MicroBlaze System:

```
XMD% dow -data ../bclinux_images/v5fxt_dev1_kit_demo.kdi
0x90000000
```

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

10. To start the kernel image running and boot BlueCat Linux, use one of the following XMD commands, depending on the system.

PPC440 System:

```
XMD% con 0x03007000
```

MicroBlaze System:

```
XMD% con 0x90000000
```

- a. After BlueCat Linux boots, the HyperTerminal output will be similar to the output shown in [Figure 3-1](#).

```
Linux version 2.6.13.4 ( ) (gcc version 3.4.3) #1 Thu Apr 23 11
:45:05 EDT 2009
Xilinx ML507 Reference System (Virtex-5 FXT)
Built 1 zonelists
Kernel command line: ramdisk_size=65536 xilinx_emc_part_conf=0-25:26-45:46-125:126-254
hda=bswap hdb=bswap hdc=bswap hdd=bswap root=101
Xilinx INTC #0 at 0x81800000 mapped to 0xFDFFE000
PID hash table entries: 2048 (order: 11, 32768 bytes)
Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
Memory: 250880k available (1724k kernel code, 464k data, 84k init, 0k highmem)
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 0x81460000 mapped to 0xD1060000 device: 10,185 using IRQ#7
xgpio01 #1 at 0x81400000 mapped to 0xD1080000 device: 10,186 not using IRQ
xgpio02 #2 at 0x81420000 mapped to 0xD10A0000 device: 10,187 not using IRQ
xgpio03 #3 at 0x81440000 mapped to 0xD10C0000 device: 10,188 using IRQ#8
Serial: 8250/16550 driver $Revision: 1.90 $ 4 ports, IRQ sharing enabled
ttyS0 at MMIO 0x83e01003 (irq = 9) is a 16550A
ttyS0 at MMIO 0x83e01003 (irq = 9) is a 16550A
io scheduler noop registered
io scheduler anticipatory registered
io scheduler deadline registered
io scheduler cfq registered
RAMDISK driver initialized: 16 RAM disks of 65536K size 1024 blocksize
xilinx_sysace: Could not lock (XST_DEVICE_BUSY), giving up
XLLTemac: using DMA mode.
XLLTemac: DCR address: 0x80
XLLTemac: buffer descriptor size: 32768 (0x8000)
XLLTemac: (buffer_descriptor_init) phy: 0x4c8000, virt: 0xc04c8000, size: 0x8000
XLLTemac: PHY detected at address 7.
eth0: Xilinx TEMAC #0 at 0xD10E0000 mapped to 0xD10E0000, irq=5
EMC Flash on Xilinx board: Found 1 x16 devices at 0x0 in 16-bit bank
Intel/Sharp Extended Query Table at 0x010A
cfi_cmdset_0001: Suspend erase on write disabled.
Using buffer write method
Registering a 32MB EMC Flash at 0xFC000000
EMC Flash MTD driver: Configuration of partitions is 0-25:26-45:46-125:126-254
Creating 4 MTD partitions on "EMC Flash on Xilinx board":
0x00000000-0x00340000 : "EMC Flash on Xilinx board"
0x00340000-0x005c0000 : "EMC Flash on Xilinx board"
0x005c0000-0x00fc0000 : "EMC Flash on Xilinx board"
0x00fc0000-0x01fe0000 : "EMC Flash on Xilinx board"
EMC Flash MTD driver: Configured 4 partitions
xilinx_iic.0 #0 at 0x81600000 mapped to 0xD3120000, irq=6
NET: Registered protocol family 2
IP route cache hash table entries: 4096 (order: 2, 16384 bytes)
TCP established hash table entries: 16384 (order: 5, 131072 bytes)
TCP bind hash table entries: 16384 (order: 4, 65536 bytes)
TCP: Hash tables configured (established 16384 bind 16384)
TCP reno registered
TCP bic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
eth0: XLLTemac: Options: 0x3f8
eth0: XLLTemac: allocating interrupt 11 for dma mode tx.
eth0: XLLTemac: allocating interrupt 10 for dma mode rx.
eth0: XLLTemac: speed set to 100Mb/s
eth0: XLLTemac: Send Threshold = 32, Receive Threshold = 2
eth0: XLLTemac: Send Wait bound = 255, Receive Wait bound = 255
IP-Config: Incomplete network configuration information.
RAMDISK: Compressed image found at block 17440
Freeing BlueCat RFS memory: 6356k freed
VFS: Mounted root (ext2 filesystem).
Freeing unused kernel memory: 84k init
INIT: version 2.85 booting
eth0: XLLTemac: Options: 0x7f8
eth0: XLLTemac: allocating interrupt 11 for dma mode tx.
eth0: XLLTemac: allocating interrupt 10 for dma mode rx.
eth0: XLLTemac: speed set to 100Mb/s
eth0: XLLTemac: Send Threshold = 32, Receive Threshold = 2
eth0: XLLTemac: Send Wait bound = 255, Receive Wait bound = 255
Starting Apache HTTP server
Running the DHCP client
INIT: Entering runlevel: 1
myhostname login: root
```

UG511_03_01_052109

Figure 3-1: BlueCat Linux Boot Output - Ramdisk

- 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. Connect the Platform USB cable or the Parallel IV JTAG cable between the host computer and the Virtex-5 FXT ML507 development board.
2. Connect the serial cable between the host computer and the RS232 port on the Virtex-5 FXT ML507 development board.
3. Apply power to the Virtex-5 FXT ML507 development board.
4. Start a HyperTerminal (or similar) session on the host computer. 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**.
5. Open the reference system project in XPS.
6. Implement the hardware design and create the hardware bitstream by selecting **Hardware** → **Generate Bitstream** in XPS.
7. Download the bitstream to the board by selecting **Device Configuration** → **Download Bitstream** in XPS.
8. Select **Debug** → **Launch XMD...** to launch an XMD command window.
9. In XMD, download the BlueCat Linux kernel image into DDR2 memory using one of the following commands, depending on the system.

PPC440 System:

```
XMD% dow -data bclinux_images/v5fxt_dev1_kit_demo.kdi
0x03000000
```

MicroBlaze System:

```
XMD% dow -data bclinux_images/v5fxt_dev1_kit_demo.kdi
0x90000000
```

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

10. To start the kernel image running and boot BlueCat Linux, use one of the following XMD commands, depending on the system.

PPC440 System:

```
XMD% con 0x03007000
```

MicroBlaze System:

```
XMD% con 0x90000000
```

- a. After BlueCat Linux boots, the HyperTerminal output will be as shown in [Figure 3-1](#).
- 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 image, the hardware bitstream must be programmed to the Virtex-5 FXT 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 system root directory, follow these steps:

1. Connect the Platform USB cable or the Parallel IV JTAG cable between the host computer and the Virtex-5 FXT ML507 development board.
2. Connect the serial cable between the host computer and the RS232 port on the Virtex-5 FXT ML507 development board.
3. Apply power to the Virtex-5 FXT ML507 development board.
4. Start a HyperTerminal (or similar) session on the host computer. 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**.
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 ug511.cmd
```

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

```
$ xmd -opt ug511.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 46-125 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 3-2](#).
 - a. Enter 3 at the Main Menu.
 - b. In the Flash Erase Menu, enter 2.
 - c. Enter 46 as the starting block.

- d. Enter 125 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 (0-255) to erase: 46
Enter ending block (46-255) to erase: 125
Erasing blocks 46-125 of Flash...
.....
Erased the Flash memory contents successfully
```

UG511_03_11_121208

Figure 3-2: Erase Blocks 46-125 of the Flash

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

```
XMD% stop
```

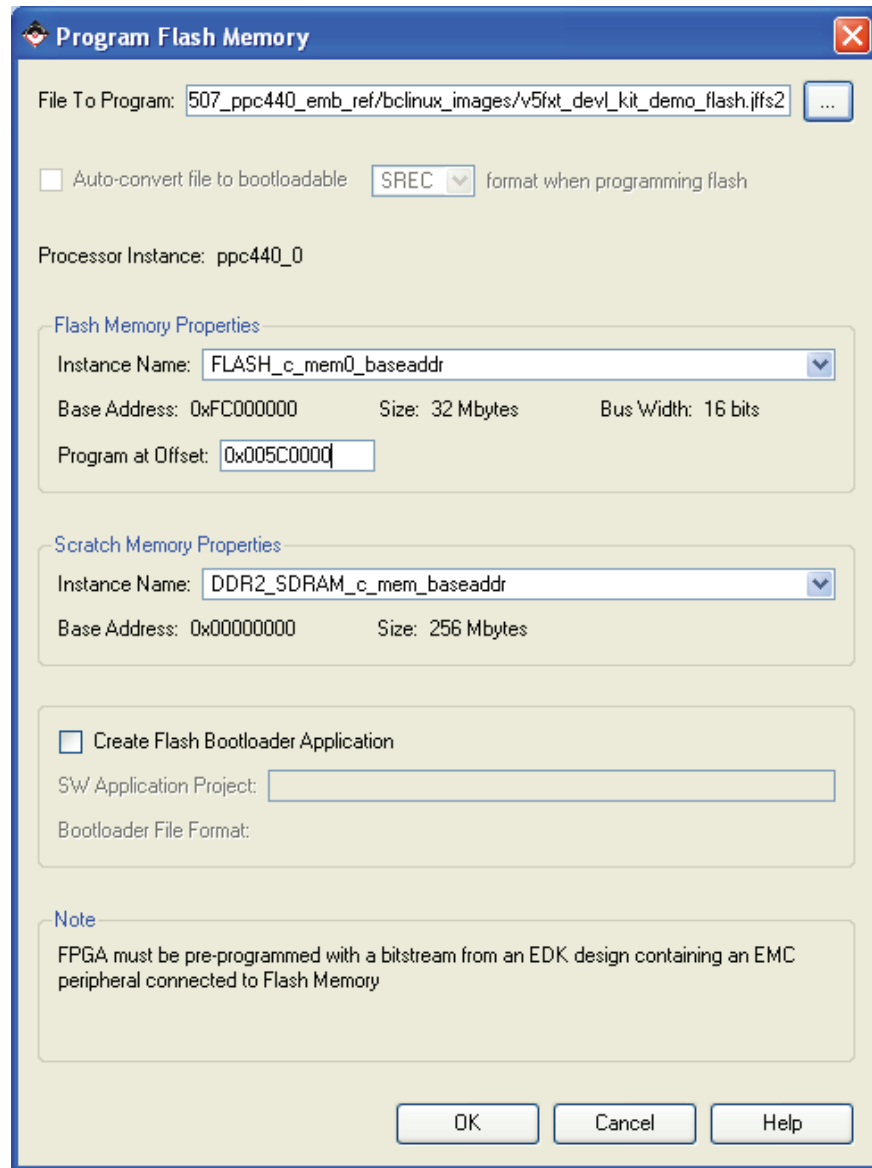
```
XMD% rst
```

```
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/v5fxt_dev1_kit_demo_flash.jffs2. Enter the offset to be

0x005C0000. The external DDR2 memory is set as the Scratch Memory. The Program Flash Memory settings are shown in [Figure 3-3](#).



Program Flash Memory

File To Program: 507_ppc440_emb_ref/bclinux_images/v5fxt_devl_kit_demo_flash.jffs2 ...

☐ Auto-convert file to bootloadable SREC format when programming flash

Processor Instance: ppc440_0

Flash Memory Properties

Instance Name: FLASH_c_mem0_baseaddr

Base Address: 0xFC000000 Size: 32 Mbytes Bus Width: 16 bits

Program at Offset: 0x005C0000

Scratch Memory Properties

Instance Name: DDR2_SDRAM_c_mem_baseaddr

Base Address: 0x00000000 Size: 256 Mbytes

☐ Create Flash Bootloader Application

SW Application Project:

Bootloader File Format:

Note

FPGA must be pre-programmed with a bitstream from an EDK design containing an EMC peripheral connected to Flash Memory

OK Cancel Help

UG511_03_12_121208

Figure 3-3: 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 ug511.opt
```

16. Download the BlueCat Linux kernel image that uses the JFFS2 filesystem into DDR2 memory using one of the following commands, depending on the system.

PPC440 System:

```
XMD% dow -data  
../bclinux_images/v5fxt_dev1_kit_demo_flash.kdi 0x03000000
```

MicroBlaze System:

```
XMD% dow -data  
../bclinux_images/v5fxt_dev1_kit_demo_flash.kdi 0x90000000
```

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

17. To start the kernel image running and boot BlueCat Linux, use one of the following XMD commands, depending on the system.

PPC440 System:

```
XMD% con 0x03007000
```

MicroBlaze System:

```
XMD% con 0x90000000
```

- a. After BlueCat Linux boots, the HyperTerminal output will be similar to the output shown in [Figure 3-4](#).

```
Linux version 2.6.13.4 () (gcc version 3.4.3) #1 Thu Apr 23 11:49:09 EDT 2009
Xilinx ML507 Reference System (Virtex-5 FXT)
Built 1 zonelists
Kernel command line: root=/dev/nfs rw ip=on rootfstype=jffs2 noinitrd rw xilinx_
emc_part_conf=0-25:26-45:46-125:126-254 hda=bswap hdb=bswap hdc=bswap hdd=bswap
root=1f03
Xilinx INTC #0 at 0x81800000 mapped to 0xFDFFE000
PID hash table entries: 2048 (order: 11, 32768 bytes)
Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
Memory: 257280k available (1724k kernel code, 464k data, 84k init, 0k highmem)
Mount-cache hash table entries: 512
NET: Registered protocol family 16
JFFS2 version 2.2. (NAND) (C) 2001-2003 Red Hat, Inc.
xgpic00 #0 at 0x81460000 mapped to 0xD1060000 device: 10,185 using IRQ#7
xgpic01 #1 at 0x81400000 mapped to 0xD1080000 device: 10,186 not using IRQ
xgpic02 #2 at 0x81420000 mapped to 0xD10A0000 device: 10,187 not using IRQ
xgpic03 #3 at 0x81440000 mapped to 0xD10C0000 device: 10,188 using IRQ#8
Serial: 8250/16550 driver $Revision: 1.90 $ 4 ports, IRQ sharing enabled
ttyS0 at MMIO 0x83e01003 (irq = 9) is a 16550A
ttyS0 at MMIO 0x83e01003 (irq = 9) is a 16550A
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
xilinx_sysace: Could not lock (XST_DEVICE_BUSY), giving up
XLlTemac: using DMA mode.
XLlTemac: DCR address: 0x80
XLlTemac: buffer descriptor size: 32768 (0x8000)
XLlTemac: (buffer_descriptor_init) phy: 0x508000, virt: 0xc0508000, size: 0x8000
XLlTemac: PHY detected at address 7.
eth0: Xilinx TEMAC #0 at 0xD10E0000 mapped to 0xD10E0000, irq=5
EMC Flash on Xilinx board: Found 1 x16 devices at 0x0 in 16-bit bank
Intel/Sharp Extended Query Table at 0x010A
cfi_cmdset_0001: Suspend erase on write disabled.
Using buffer write method
Registering a 32MB EMC Flash at 0xFC000000
EMC Flash MTD driver: Configuration of partitions is 0-25:26-45:46-125:126-254
Creating 4 MTD partitions on "EMC Flash on Xilinx board":
0x00000000-0x00340000 : "EMC Flash on Xilinx board"
0x00340000-0x005c0000 : "EMC Flash on Xilinx board"
0x005c0000-0x00fc0000 : "EMC Flash on Xilinx board"
0x00fc0000-0x01fe0000 : "EMC Flash on Xilinx board"
EMC Flash MTD driver: Configured 4 partitions
xilinx_iic.0 #0 at 0x81600000 mapped to 0xD3120000, irq=6
NET: Registered protocol family 2
IP route cache hash table entries: 4096 (order: 2, 16384 bytes)
TCP established hash table entries: 16384 (order: 5, 131072 bytes)
TCP bind hash table entries: 16384 (order: 4, 65536 bytes)
TCP: Hash tables configured (established 16384 bind 16384)
TCP reno registered
TCP bic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
eth0: XLlTemac: Options: 0x3f8
eth0: XLlTemac: allocating interrupt 11 for dma mode tx.
eth0: XLlTemac: allocating interrupt 10 for dma mode rx.
eth0: XLlTemac: speed set to 100Mb/s
eth0: XLlTemac: Send Threshold = 32, Receive Threshold = 2
eth0: XLlTemac: Send Wait bound = 255, Receive Wait bound = 255
IP-Config: Incomplete network configuration information.
RAMDISK: Couldn't find valid RAM disk image starting at 0.
VFS: Mounted root (jffs2 filesystem).
Freeing unused kernel memory: 84k init
INIT: version 2.85 booting
eth0: XLlTemac: Options: 0x7f8
eth0: XLlTemac: allocating interrupt 11 for dma mode tx.
eth0: XLlTemac: allocating interrupt 10 for dma mode rx.
eth0: XLlTemac: speed set to 100Mb/s
eth0: XLlTemac: Send Threshold = 32, Receive Threshold = 2
eth0: XLlTemac: Send Wait bound = 255, Receive Wait bound = 255
Starting Apache HTTP server
Running the DHCP client
INIT: Entering runlevel: 1
myhostname login: root
```

UG511_03_13_052109

Figure 3-4: BlueCat Linux Boot Output - JFFS2

- 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. Connect the Platform USB cable or the Parallel IV JTAG cable between the host computer and the Virtex-5 FXT ML507 development board.
2. Connect the serial cable between the host computer and the RS232 port on the Virtex-5 FXT ML507 development board.
3. Apply power to the Virtex-5 FXT ML507 development board.
4. Start a HyperTerminal (or similar) session on the host computer. 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**.
5. Open the reference system project in XPS.
6. Implement the hardware design and create the hardware bitstream by selecting **Hardware** → **Generate Bitstream** in XPS.
7. Download the bitstream to the board by selecting **Device Configuration** → **Download Bitstream** in XPS.
8. Right click the FlashRWE software application project and select **Build Project** to create the executable file.
9. Select **Debug** → **Launch XMD...** to launch an XMD command window.
10. Download the FlashRWE software application into BRAM using the following command:

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

11. 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.

12. With the FlashRWE program, erase blocks 46-125 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 3-5](#).
 - a. Enter 3 at the Main Menu.
 - b. In the Flash Erase Menu, enter 2.
 - c. Enter 46 as the starting block.
 - d. Enter 125 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 (0-255) to erase: 46
Enter ending block (46-255) to erase: 125
Erasing blocks 46-125 of Flash...
.....
Erased the Flash memory contents successfully
  
```

UG511_03_14_121208

Figure 3-5: Erase Blocks 46-125 of the Flash

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

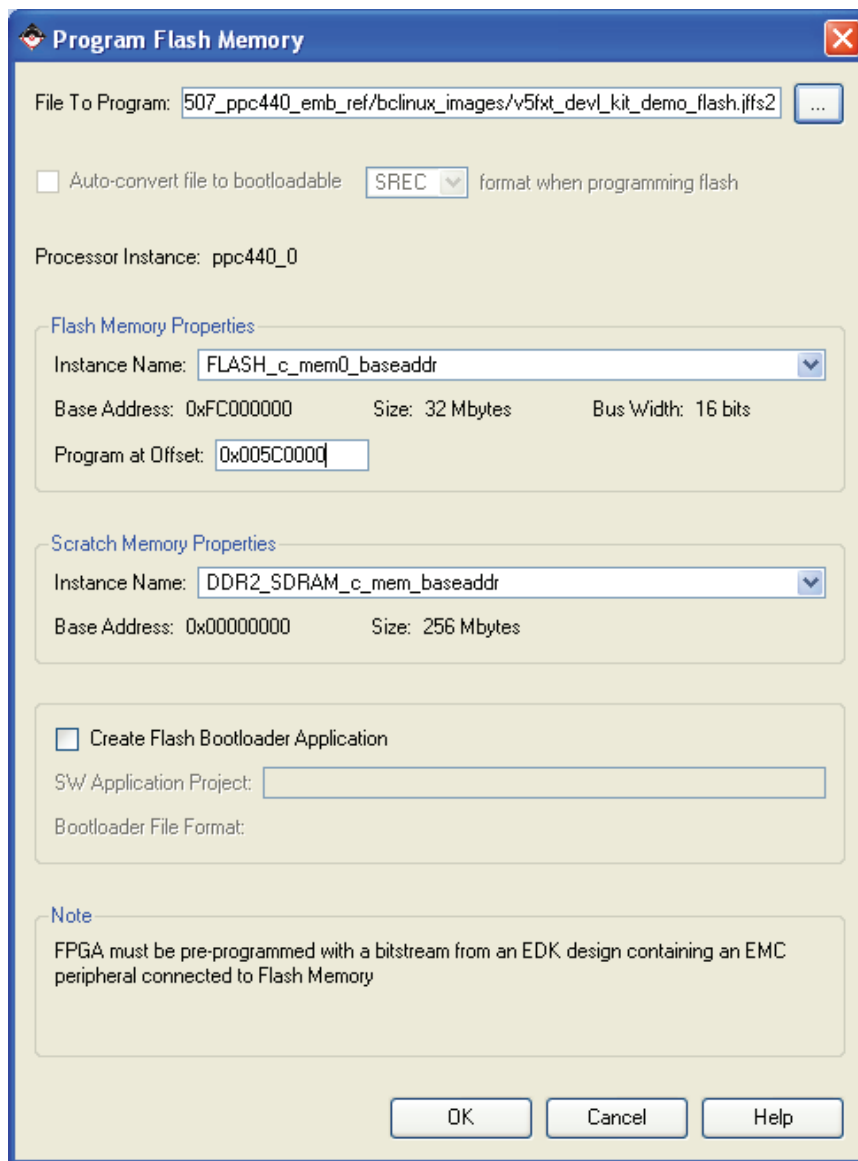

```

XMD% stop

XMD% rst

XMD% exit
      
```
14. In XPS, select **Device Configuration** → **Program Flash Memory**.
15. In the Program Flash Memory dialog box, choose the file to program to be `/bclinux_images/v5fxt_dev1_kit_demo_flash.jffs2`. Enter the offset to be

0x005C0000. The external DDR2 memory is set as the Scratch Memory. The Program Flash Memory settings are shown in Figure 3-6.



UG511_03_15_121208

Figure 3-6: Program Flash Memory Box for the JFFS2 File System

16. Select **Debug** → **Launch XMD...** to launch an XMD command window.
17. Download the BlueCat Linux kernel image that uses the JFFS2 filesystem into DDR2 memory using one of the following commands, depending on the system.

PPC440 System:

```
XMD% dow -data
../bclinux_images/v5fxt_dev1_kit_demo_flash.kdi 0x03000000
```

MicroBlaze System:

```
XMD% dow -data
../bclinux_images/v5fxt_dev1_kit_demo_flash.kdi 0x90000000
```


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

18. To start the kernel image running and boot BlueCat Linux, use one of the following XMD commands, depending on the system.

PPC440 Processor System:

```
XMD% con 0x03007000
```

MicroBlaze Processor System:

```
XMD% con 0x90000000
```

- a. After BlueCat Linux boots, the HyperTerminal output will be as shown in [Figure 3-4](#).
- 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.

The BlueCat Linux kernel images were built with networking support enabled, therefore the images support 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 timeout 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 3-7](#). In the figure, the board IP address is 192.168.0.126. The board IP address can be manually set or changed by issuing the command **ifconfig eth0 IP_address**.

```
eth0      Link encap:Ethernet  HWaddr 00:01:00:00:FE:44
          inet addr:192.168.0.126  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MTU:1500 Metric:1
          RX packets:83 errors:0 dropped:0 overruns:0 frame:0
          TX packets:83 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:22140 (21.6 KiB)  TX bytes:21486 (20.9 KiB)
          Interrupt:5

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)
```

UG511_03_02_070108

Figure 3-7: Ethernet Configuration Settings

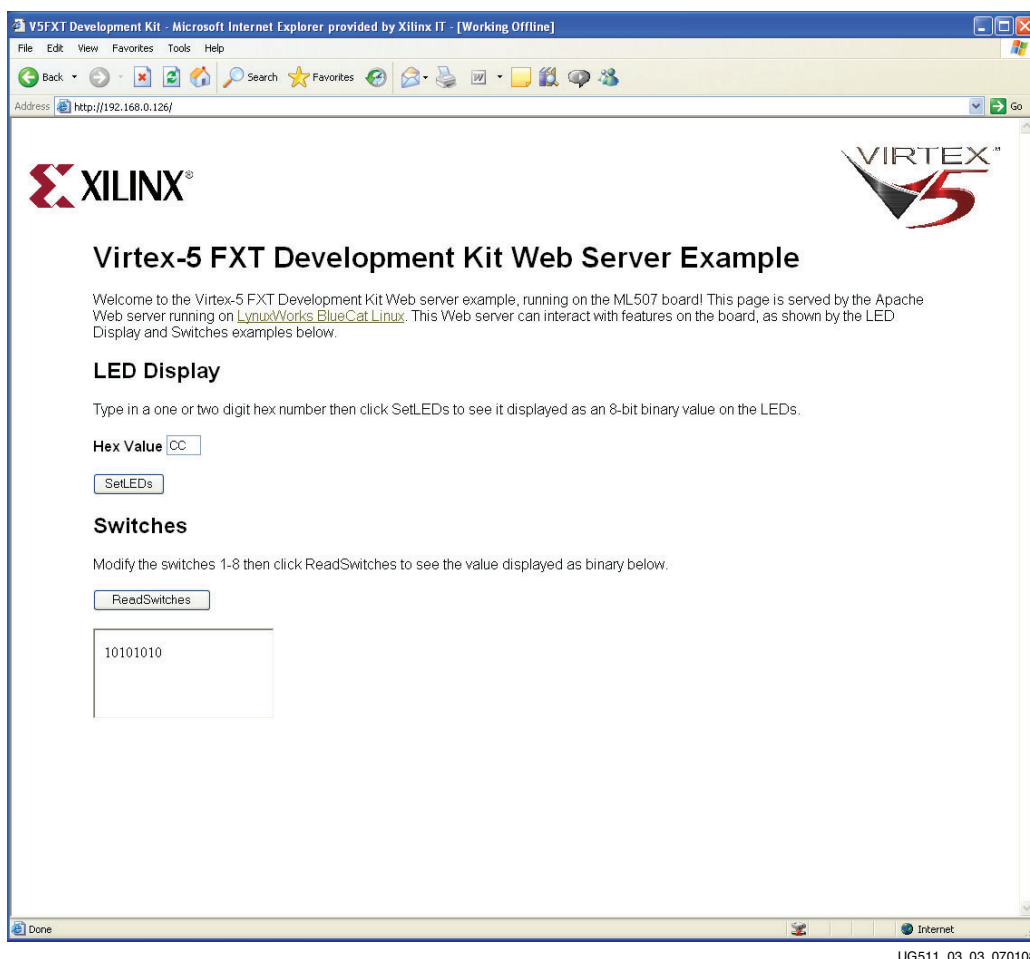
To ping a remote computer from the development board, the command string, **ping -c 4 remote_computer_IP_address**, is used to ping the remote computer 4 times.

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 3-8. 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.



UG511_03_03_070108

Figure 3-8: 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.

Building the BlueCat Linux kernel image for the PowerPC 440 processor system requires the BlueCat Linux core components for the PPC and the ML507 Board Support Package (BSP). For more information on the ML507 BSP, see the *BlueCat Linux Board Support Guide for Xilinx ML507 Virtex-5 FXT Boards*.

Building the BlueCat Linux kernel image for the MicroBlaze processor system requires the BlueCat Linux core components for the MicroBlaze processor and the Spartan-3E BSP. 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*, the *BlueCat Linux Board Support Guide for Xilinx ML507 Virtex-5 FXT Boards*, 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 and the 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* referenced 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 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 = ml507 for the PPC440 processor system or bsp = sp3e for the MicroBlaze processor system.

3. After the BSP is installed, support for it must be activated in the bash shell. To activate the 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 Demo Directories

BlueCat Linux demo directories are provided with the reference systems 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 demos in the BlueCat Linux development environment. In each EDK project, there are two zipped demo directories. The v5fxt_dev1_kit_demo.zip file is the demo directory to recreate the BlueCat Linux image that boots with a ramdisk file system. The

v5fxt_dev1_kit_demo_flash.zip file is the demo directory to recreate the BlueCat Linux image that boots with a JFFS2 file system.

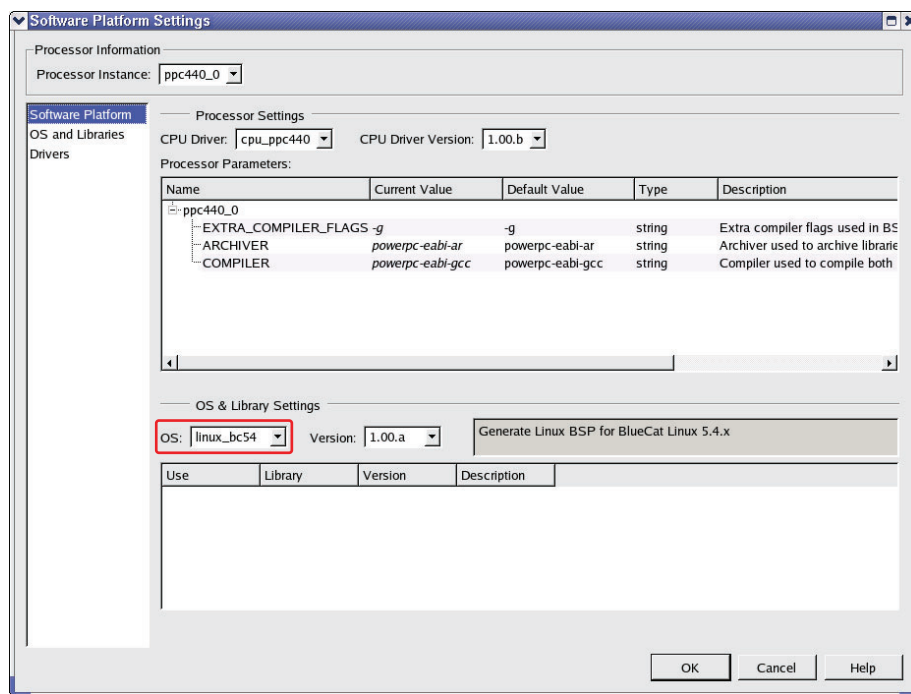
Getting the MLD File Set

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

Generating the BSP

With the use of the BlueCat Linux MLD, XPS 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 3-9.



UG511_03_04_070108

Figure 3-9: 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_install_point>/usr/src/linux

KERNEL_CONFIG:

Ramdisk file system:

```
<BlueCat_Linux_install_point>/demo/v5fxt_devl_kit_demo/v5fxt_devl_kit_demo.config
```

Flash file system:

```
<BlueCat_Linux_install_point>/demo/v5fxt_devl_kit_demo_flash/v5fxt_devl_kit_demo_flash.config
```

An example showing these fields for the ramdisk file system is in [Figure 3-10](#).

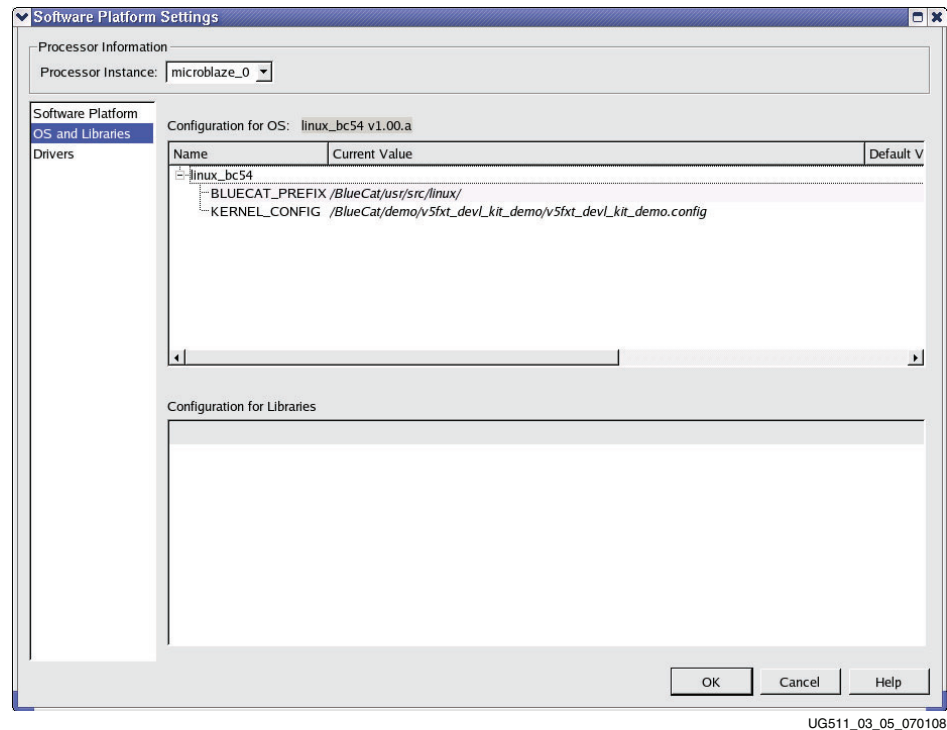


Figure 3-10: Set the BlueCat Linux Paths

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 these steps:

1. Two patches are provided with the reference systems. The `cfi_patch` allows the Flash memory to be used to hold a file system without chip errors. The `xlltemac_patch` makes the xlltemac driver compatible with v2.00.a of the XPS LL TEMAC core. The patches are located in the `/bclinux_demo` directory in the project. The patch files must be copied to the BlueCat Linux installation location, `$BLUECAT_PREFIX`, then the patches can be applied.

```
BlueCat:$ cd $BLUECAT_PREFIX
```

```
BlueCat:$ cp <project root directory>/bclinux_demo/cfi_patch .
```

- ```
BlueCat:$ cp <project root
directory/bclinux_demo/xlltemac_patch .

BlueCat:$ patch -p0 < cfi_patch

BlueCat:$ patch -p0 < xlltemac_patch
```
2. To force all kernel components to rebuild, clean the kernel tree 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/v5fxt_dev1_kit_demo
```

Flash file system:

```
BlueCat:$ cd
$BLUECAT_PREFIX/demo/v5fxt_dev1_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, 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/v5fxt_dev1_kit_demo/v5fxt_dev1_kit_demo
.kdi
```

Flash file system:

```
$BLUECAT_PREFIX/demo/v5fxt_dev1_kit_demo_flash/v5fxt_dev1_ki
t_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 DDR

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, root filesystem, 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. This section details the steps to program the BlueCat Linux image into parallel Flash. Flash files that have already been generated are provided, or the user can create new files for programming the Flash.

After programming all the files into the Flash memory as detailed in the steps in this section, the Flash memory will have the address map shown in [Table 3-1](#). The files do not necessarily take the entire space assigned. Each file starts on a block boundary so that each can be erased individually without affecting the other files in the device.

**Note:** Before starting these steps, make sure that the standalone OS is chosen under Software Platform Settings. If changing the OS to standalone, make sure to set stdout and stdin in the OS and Libraries settings.

**Table 3-1: Parallel Flash Address Mapping**

| File                                  | Start Address Offset | End Address Offset |
|---------------------------------------|----------------------|--------------------|
| Bootloader/System Configuration (BIN) | 0x00000000           | 0x0033FFFF         |
| BlueCat Linux Image (KDI)             | 0x00340000           | 0x005BFFFF         |
| Flash File System (JFFS2)             | 0x005C0000           | 0x00FBFFFF         |
| Unused Space                          | 0x00FC0000           | 0x01FFFFFF         |

## 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.

### Programming the KDI File

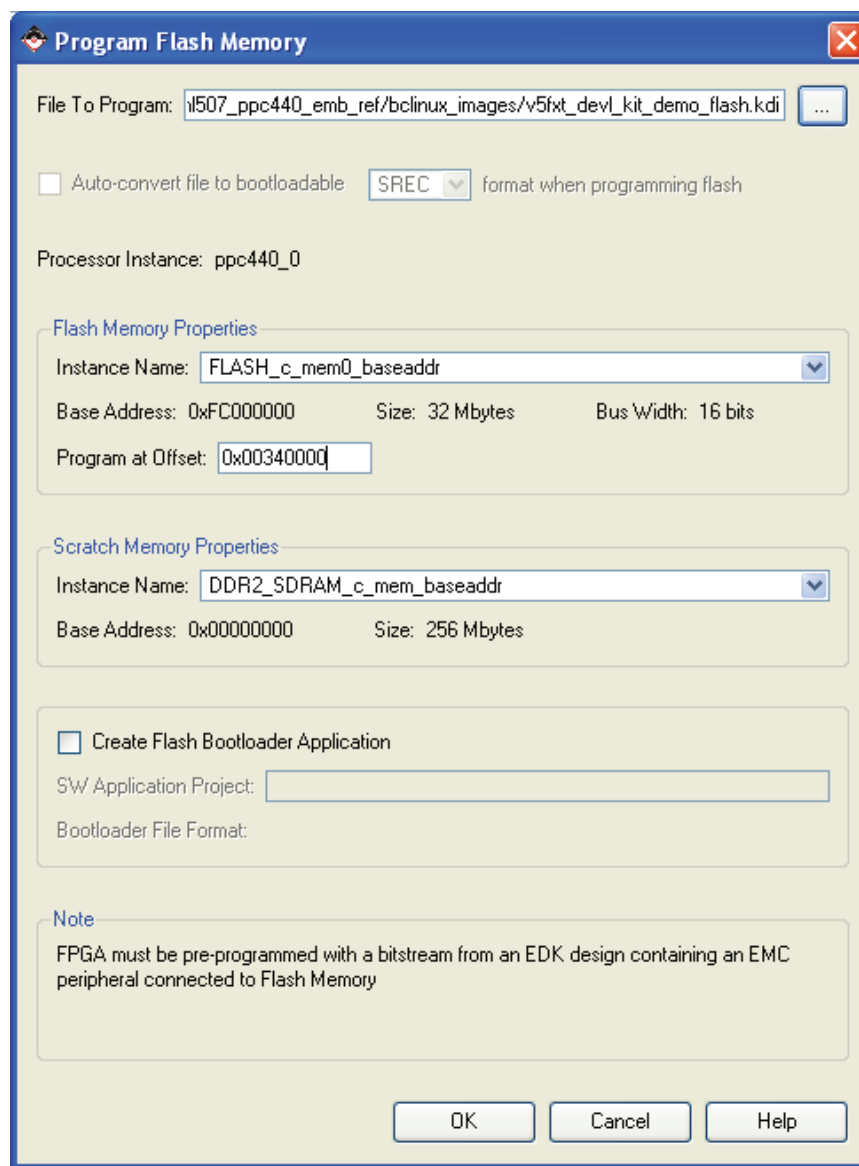
1. Open the reference system project in XPS.
2. Connect the USB or Parallel IV programming cable, the serial cable, and the power supply to the ML507 V5FXT development board. Power on the board.
3. The target board must be configured with the project bitstream before XPS can program the Flash memory. In an EDK shell, change directories to the `ready_for_download` directory. Use `iMPACT` to download the bitstream by using the following command:  

```
$ impact -batch ug511.cmd
```
4. In XPS, select **Device Configuration** → **Program Flash Memory**.



- In the Program Flash Memory dialog box, choose the file to program to be `/bclinux_images/v5fxt_dev1_kit_demo_flash.kdi` under the project root directory. Change the program offset to `0x00340000`, which will allow enough room for the bootloader and will not overwrite the location that BlueCat Linux expects the Flash file system. The external DDR2 memory is set as the Scratch Memory. The Program Flash Memory settings are shown in Figure 3-11.

**Note:** The bootloader provided in the `bootloader_bclinux` directory assumes the image has been programmed at an offset of `0x00340000`.



UG511\_03\_16\_121208

Figure 3-11: Program Flash Memory Dialog Box for BlueCat Linux Image

- Click the OK button. This will program the flash memory with the BlueCat Linux image.

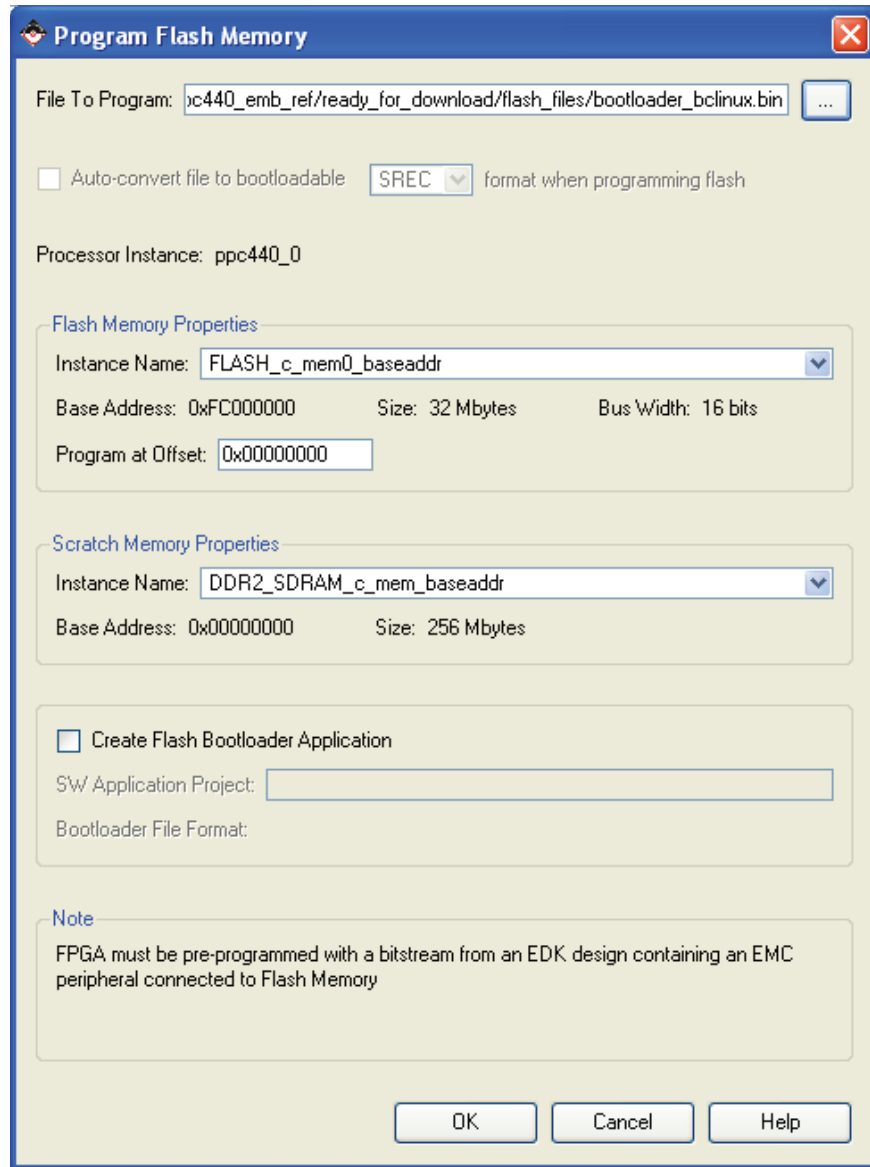
**Note:** The Program Flash Memory application will take a long time as the application must program the image of size approximately 2 MB.



## Programming the BIN File

The next step is to program the BIN file for the system. The BIN file is used to program the Flash with the system bitstream and bootloader loaded in the bitstream.

1. From the XPS menu, select **Device Configuration** → **Program Flash Memory**.
2. In the Program Flash Memory dialog box, choose the file to program to be `bootloader_bclinux.bin` under the `ready_for_download/flash_files` directory of the project. Change the program offset to `0x00000000`. The external DDR2 memory is set as the Scratch Memory. The Program Flash Memory settings are shown in Figure 3-12..



UG511\_03\_17\_121208

**Figure 3-12: Program Flash Memory Box for the Bootloader and Bitstream BIN File**

3. Click the **OK** button. This will program the flash memory with the bitstream and bootloader loaded in the bitstream.

**Note:** The Program Flash Memory application will take a long time as the application must program the BIN file of size approximately 3 MB.

## Programming the Root File System

1. In an EDK shell, change directories to the `ready_for_download` directory.
2. Invoke XMD and connect to the processor by the following command:
3. Download the FlashRWE software application into BRAM using the following command:

```
$ xmd -opt ug511.opt
```

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

4. 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.

5. With the FlashRWE program, erase blocks 46-125 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 3-13.
  - a. Enter 3 at the Main Menu.
  - b. In the Flash Erase Menu, enter 2.
  - c. Enter 46 as the starting block.
  - d. Enter 125 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 (0-255) to erase: 46
Enter ending block (46-255) to erase: 125
Erasing blocks 46-125 of Flash...
.....
Erased the Flash memory contents successfully
```

UG511\_03\_14\_121208

Figure 3-13: Erase Blocks 46-125 of the Flash

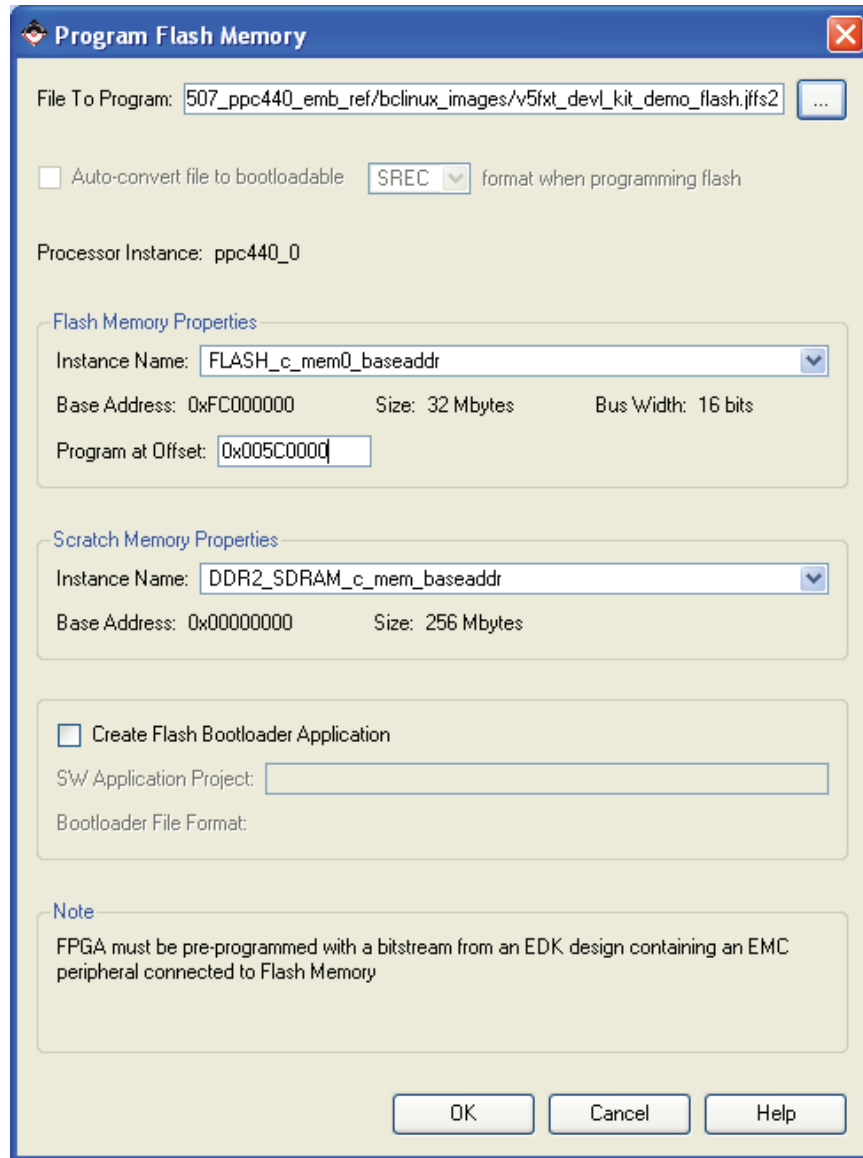
6. In XMD, stop and reset the processor. Then, exit XMD.

```
XMD% stop
```

```
XMD% rst
```

```
XMD% exit
```

7. In XPS, select **Device Configuration** → **Program Flash Memory**.
8. In the Program Flash Memory dialog box, choose the file to program to be <project root directory>/bclinux\_images/v5fxt\_dev1\_kit\_demo\_flash.jffs2. Enter the offset to be 0x005C0000. The external DDR2 memory is set as the Scratch Memory. The Program Flash Memory settings are shown in Figure 3-14.



UG511\_03\_15\_121208

**Figure 3-14: Program Flash Memory Box for the JFFS2 File System**

9. Click the **OK** button. This will program the flash memory with the JFFS2 file.

**Note:** The Program Flash Memory application will take a long time as the application must program the JFFS2 file of size approximately 7 MB.

## Running the Design

1. Once the Flash is programmed with all the files documented in the previous steps, change the configuration address / mode DIP switches to **00001001**.

2. Press the **Prog** button. The design takes about 10 seconds to finish loading and begin to run.

## 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.

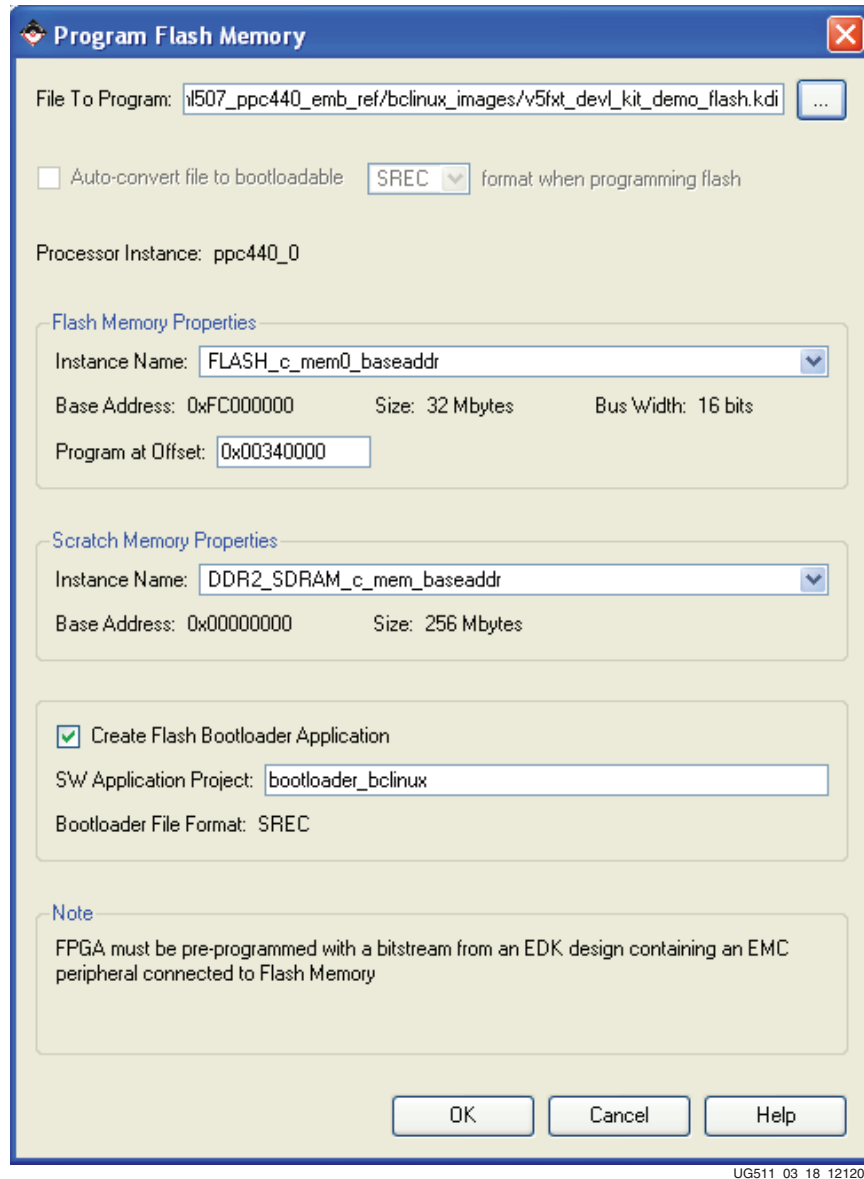
### Programming the KDI File and Creating the Bootloader

1. Open the reference system project in XPS.
2. Connect the USB or Parallel IV programming cable, the serial cable, and the power supply to the ML507 V5FXT development board. Power on the board.
3. The target board must be configured with the project bitstream before XPS can program the SREC file into the flash memory. Select **Device Configuration** → **Download Bitstream** in XPS.
4. In XPS, select **Device Configuration** → **Program Flash Memory**.

In the Program Flash Memory dialog box, choose the file to program to be `/bclinux_images/v5fxt_dev1_kit_demo_flash.kdi` under the project root directory. Change the program offset to `0x00340000`, which will allow enough room for the bootloader and will not overwrite the location that BlueCat Linux expects the Flash file system.

The external DDR2 memory is set as the Scratch Memory. A bootloader is created by clicking the *Create Flash Bootloader Application* check box in the Program Flash Memory dialog box. The Program Flash Memory settings are shown in Figure 3-15.

**Note:** The bootloader provided in the `bootloader_bclinux` directory assumes the image has been programmed at an offset of `0x00340000`.



UG511\_03\_18\_121208

Figure 3-15: Program Flash Memory Dialog Box for BlueCat Linux Image

- Click the *OK* button. This will program the flash memory with the BlueCat Linux image and will create a bootloader software application project.

**Note:** The Program Flash Memory application will take a long time as the application must program the image of size approximately 2 MB.

- After creating the bootloader files, add the following lines, shown in Figure 3-16, in the `bootloader.c` file:

```
#include "xparameters.h"
#include "xuartns550_1.h"
```

7. It is strongly suggested to disable the bootloader from displaying its progress. This is done by commenting out the following line in the `bootloader.c` generated file:

```
#define VERBOSE
```

The line to comment out to allow non-verbose bootloading is shown in [Figure 3-16](#).

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "blconfig.h"
#include "portab.h"
#include "errors.h"
#include "srec.h"
#include "xparameters.h"
#include "xuartns550_1.h"

/* Defines */
#define CR 13

/* Comment the following line, if you want a smaller and faster bootloader which will be silent */
/* #define VERBOSE */
```

UG511\_03\_19\_121208

**Figure 3-16: Code to Select Non-Verbose Bootloading**

8. The bootloader must be modified to copy the KDI image from Flash into DDR2 for the BlueCat Linux demonstration to fully operate. This is done by adding lines of code to the `bootloader.c` file that the EDK generates. To modify the newly created bootloader, add the following pieces of code:
  - a. Code is required to define the location in Flash where the KDI image resides and the location in DDR to put the KDI image. This code is shown in [Figure 3-17](#). The `KDI_FLASH_LOC` parameter should be set to the location in Flash memory of where the KDI image will be placed. The `KDI_DDR_LOC` parameter should be set to the location in DDR2 memory where the KDI image is to be copied. The `KDI_LENGTH` parameter should be set to the length of the KDI image in bytes.

**Note:** Figure 3-17 shows the code for the PPC440 system. Some of the parameters have different values for the MicroBlaze processor system and can be seen in the example code provided for the MicroBlaze processor system.

```
/* Add for copying the contents of flash to DDR */
#define KDI_DDR_LOC XPAR_DDR2_SDRAM_MEM_BASEADDR + 0x03000000 //Location to put KDI into the DDR
#define KDI_FLASH_LOC XPAR_FLASH_MEM0_BASEADDR + 0x00340000 //Location of the KDI in FLASH
#define KDI_LENGTH 0x10E000 //Size (in bytes) of the KDI with the Flash file system

/* Declarations for copying the contents from flash to DDR */
uint8_t * kdi_ddr_ptr;
uint8_t * kdi_flash_ptr;
static uint32_t ddr_loc = KDI_DDR_LOC;
static uint32_t flash_loc = KDI_FLASH_LOC;
```

UG511\_03\_20\_121208

**Figure 3-17: BlueCat Linux Bootloader Code Definitions and Declarations**

- b. Comment the SREC function `load_exec` in the source file as the KDI image is loaded as it is to the FLASH memory and so these functions are not required. Code is also required for the bootloader to copy the KDI image from flash into DDR2 when the bootloader runs. This code is shown in [Figure 3-18](#). After modifying the C file, be sure to recompile the software application.

**Note:** Figure 3-18 shows the code for the PPC440 system. The MicroBlaze processor system has a different address to execute from and therefore the code is slightly different. The

bootloader code for the MicroBlaze processor can be found in the MicroBlaze processor reference system.

```

152 /*
153 srinfo.sr_data = sr_data_buf;
154
155 while (!done) {
156 if ((ret = flash_get_srec_line (sr_buf)) != 0)
157 return ret;
158
159 if ((ret = decode_srec_line (sr_buf, &srinfo)) != 0)
160 return ret;
161
162 #ifdef VERBOSE
163 display_progress (srec_line);
164 #endif
165
166 switch (srinfo.type) {
167 case SREC_TYPE_0:
168 break;
169 case SREC_TYPE_1:
170 case SREC_TYPE_2:
171 case SREC_TYPE_3:
172 memcpy ((void*)srinfo.addr, (void*)srinfo.sr_data, srinfo.dlen);
173 break;
174 case SREC_TYPE_5:
175 break;
176 case SREC_TYPE_7:
177 case SREC_TYPE_8:
178 case SREC_TYPE_9:
179 laddr = (void (*)())srinfo.addr;
180 done = 1;
181 ret = 0;
182 break;
183 }
184 }
185
186 /* Copy the KDI from FLASH into the DDR */
187 kdi_ddr_ptr = (uint8_t*)ddr_loc;
188 kdi_flash_ptr = (uint8_t*)flash_loc;
189 memcpy (kdi_ddr_ptr, kdi_flash_ptr, KDI_LENGTH);
190
191 laddr = (void*)(ddr_loc+0x7000);
192
193 #ifdef VERBOSE
194 print ("\r\nExecuting program starting at address: ");
195 putnum ((uint32_t)laddr);
196 print ("\r\n");
197 #endif
198
199 (*laddr)();
200
201

```

UG511\_03\_09\_070108

**Figure 3-18: BlueCat Linux Bootloader Code to Copy the KDI Image**

9. In XPS, compile a bitstream download.bit, that includes the system configuration and the bootloader\_bclinux application. This is done by marking the bootloader application to **Initialize BRAM's**, then selecting **Device Configuration → Update Bitstream** in XPS.

## Creating and Programming the BIN File

Once the bootloader application is created, the next step is to create the BIN file for the system. The BIN file is used to program the Flash with the system bitstream and bootloader loaded in the bitstream.

1. In XPS, open an EDK shell.
2. Create a BIN file to program the Flash device by issuing the following command in the project root directory:
 

```
$ promgen -w -p bin -o bootloader_bclinux.bin -u 0 implementation/download.bit
```
3. From the XPS menu, select **Device Configuration → Program Flash Memory**.

4. In the Program Flash Memory dialog box, choose the file to program to be <project root directory>/bootloader\_bclinux.bin. Change the program offset to 0x00000000. The external DDR2 memory is set as the Scratch Memory. The Program Flash Memory settings are shown in Figure 3-19.

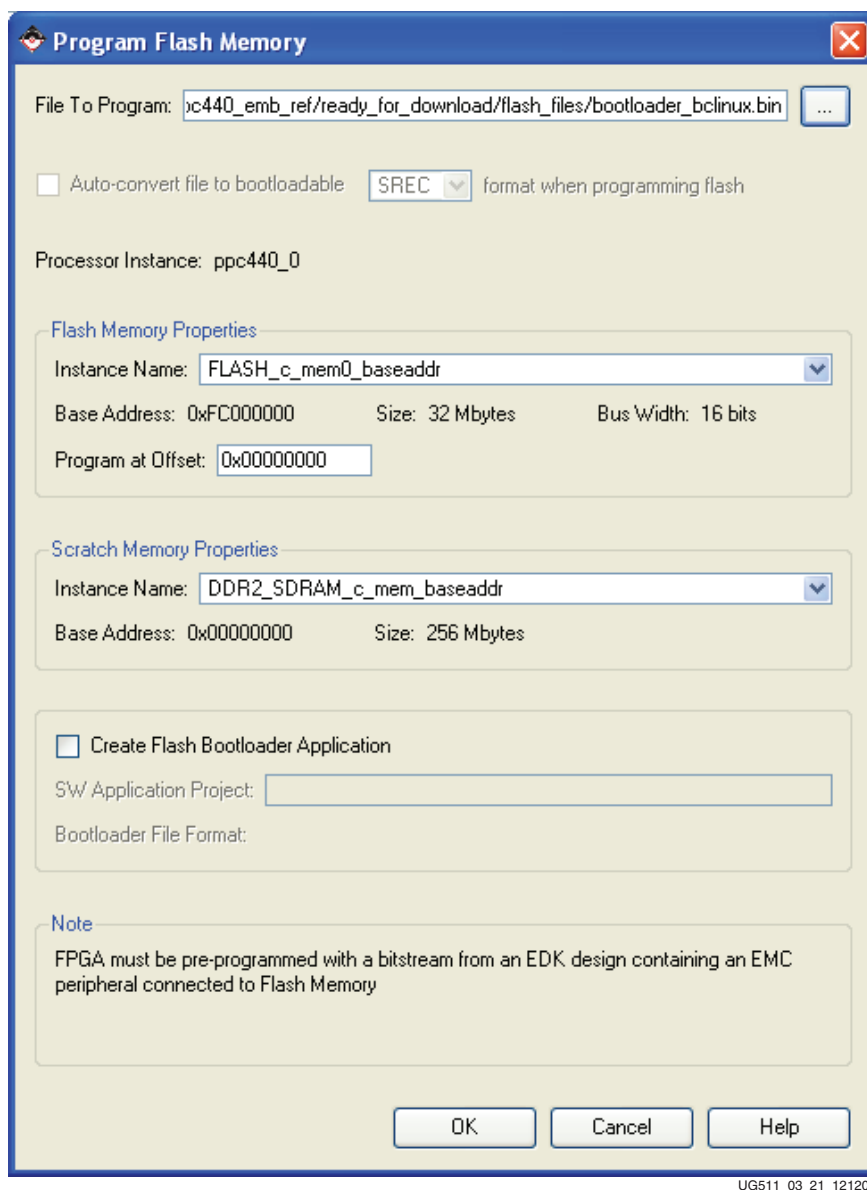


Figure 3-19: Program Flash Memory Box for the Bootloader and Bitstream BIN File

5. Click the OK button. This will program the flash memory with the bitstream and bootloader loaded in the bitstream.

**Note:** The Program Flash Memory application will take a long time as the application must program the BIN file of size approximately 3 MB.

## Programming the Root File System

1. In an EDK shell, change directories to the ready\_for\_download directory.
2. Invoke XMD and connect to the processor by the following command:



```
$ xmd -opt ug511.opt
```

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

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

4. 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.

5. With the FlashRWE program, erase blocks 46-125 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 3-20](#).
  - a. Enter 3 at the Main Menu.
  - b. In the Flash Erase Menu, enter 2.
  - c. Enter 46 as the starting block.
  - d. Enter 125 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 (0-255) to erase: 46
Enter ending block (46-255) to erase: 125
Erasing blocks 46-125 of Flash...
.....
Erased the Flash memory contents successfully
```

UG511\_03\_22\_121208

**Figure 3-20: Erase Blocks 46-125 of the Flash**

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

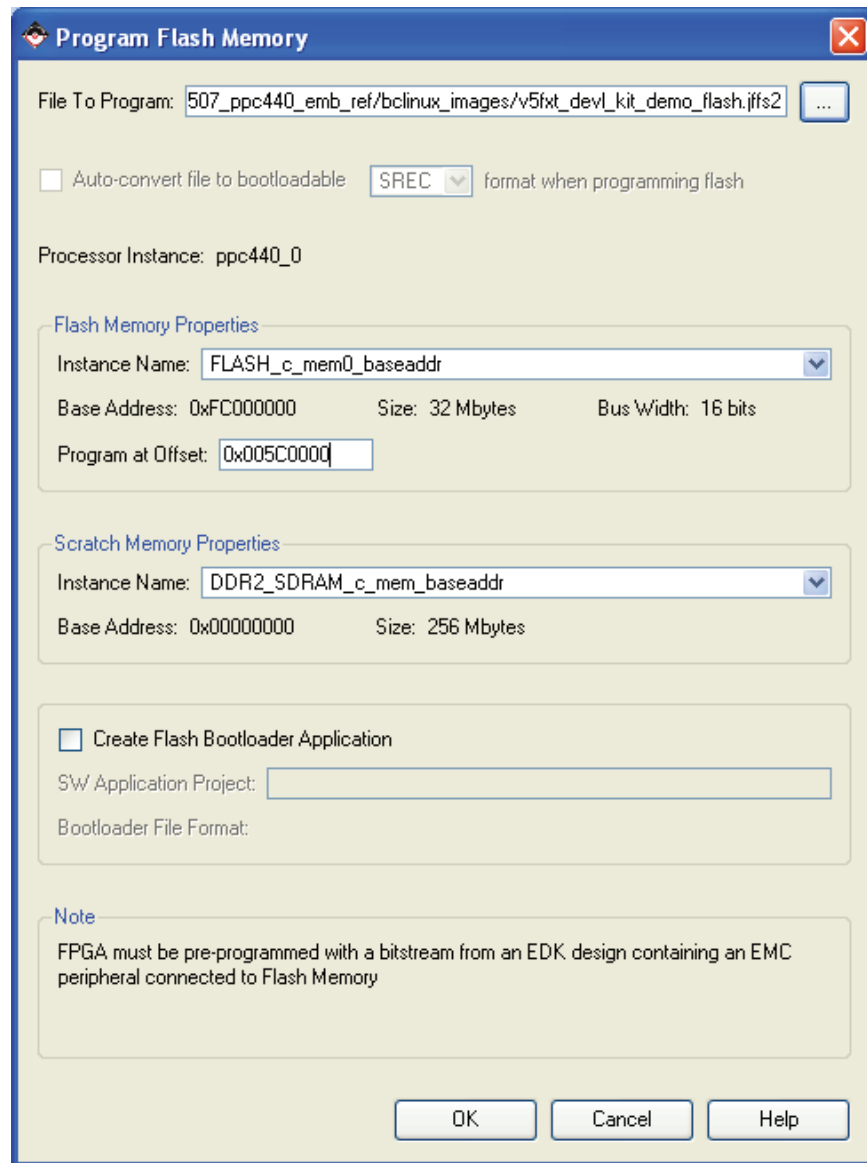
```
XMD% stop
```

```
XMD% rst
```

```
XMD% exit
```

7. In XPS, select **Device Configuration** → **Program Flash Memory**.
8. In the Program Flash Memory dialog box, choose the file to program to be `/bclinux_images/v5fxt_dev1_kit_demo_flash.jffs2`. Enter the offset to be

0x005C0000. The external DDR2 memory is set as the Scratch Memory. The Program Flash Memory settings are shown in Figure 3-21.



UG511\_03\_23\_121208

Figure 3-21: Program Flash Memory Box for the JFFS2 File System

9. Click the **OK** button. This will program the flash memory with the JFFS2 file.

**Note:** The Program Flash Memory application will take a long time as the application must program the JFFS2 file of size approximately 7 MB.

## Running the Design

1. Once the Flash is programmed with all the files documented in the previous steps, change the configuration address / mode DIP switches to **00001001**.
2. Press the **Prog** button. The design takes about 10 seconds to finish loading and begin to run.

# FlashRWE Software Application

---

## Introduction

The FlashRWE software application 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. The FlashRWE application uses the xilflash library. The FlashRWE software application can run on either the MicroBlaze processor or the PowerPC 440 processor reference system.

To download and run the FlashRWE software application:

- Download the executable file directly into BRAM using a debugger, such as XMD, which is provided as part of the EDK tools. The process 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

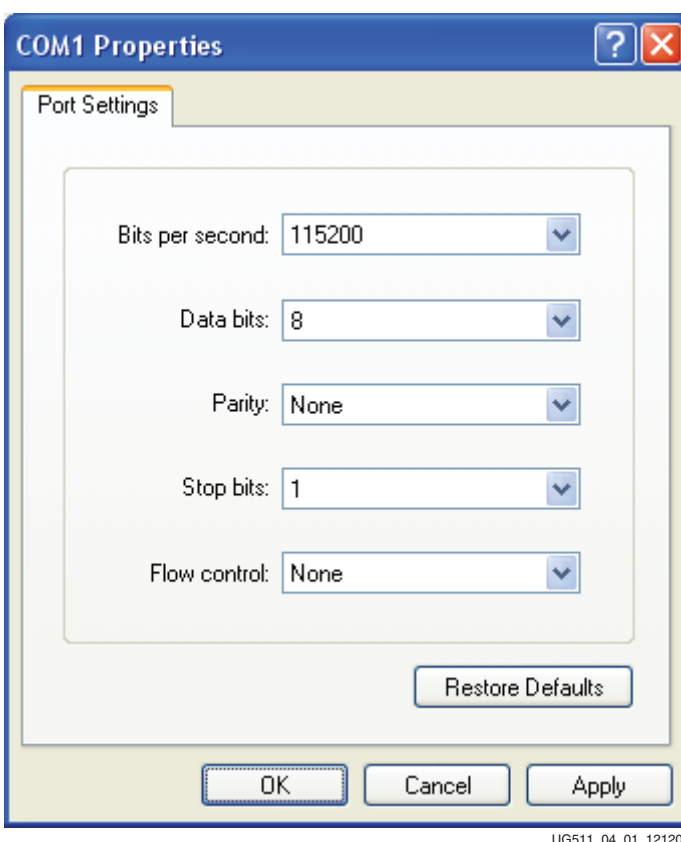
To execute the FlashRWE software application, program the hardware bitstream to the Virtex-5 FX device and load the FlashRWE software application into BRAM. Program the bitstream by downloading the pre-built bitstream from the `ready_for_download` directory or generate and download it from XPS. Similarly, 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 in 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 Virtex-5 FX70T ML507 development board.
2. Connect the serial cable between the host computer and the RS232 port on the Virtex-5 FX70T ML507 development board.
3. Apply power to the Virtex-5 FX70T ML507 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**.



UG511\_04\_01\_121208

Figure 4-1: 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 in either the MicroBlaze processor or the PowerPC processor reference system.
7. Use iMPACT to download the bitstream by using the following command:  

```
$ impact -batch ug511.cmd
```
8. Invoke XMD and connect to the processor by using the following command:  

```
$ xmd -opt ug511.opt
```
9. Download the FlashRWE software application into BRAM by using the following command:  

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

10. Start the FlashRWE software application running by using 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 55](#).
  - 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 MicroBlaze processor or the PowerPC 440 processor 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 file 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 55](#).
  - 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.

```

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

```

UG511\_04\_02\_121208

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                                                                                               | Parameters Entered by User                                       | 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                                  | <a href="#">“Read Bytes of Flash,” page 57</a>                 |
|                  | 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                                                             | <a href="#">“Check if Flash is Empty,” page 57</a>             |
|                  | Exit to Main Menu                   | Exit the submenu and return to the main menu                                                              | None                                                             | N/A                                                            |
| Flash Write      | Write Incrementing Numbers to Flash | Write incrementing numbers, 0x0-0xFF to bytes in the Flash.                                               | Address offset, Number of bytes                                  | <a href="#">“Write Incrementing Numbers to Flash,” page 59</a> |
|                  | Write Bytes to Flash                | Write bytes of data to Flash                                                                              | Address offset, Bytes of data, Number of times to write the data | <a href="#">“Write Bytes to Flash,” page 60</a>                |
|                  | 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                                  | <a href="#">“Erase Bytes of Flash,” page 61</a>                |
|                  | Erase Blocks of Flash               | Erases a single block or a range of consecutive blocks of Flash                                           | Starting block, Ending block                                     | <a href="#">“Erase Blocks of Flash,” page 62</a>               |
|                  | Erase the Entire Flash              | Erases the entire Flash device                                                                            | None                                                             | <a href="#">“Erase the Entire Flash,” page 63</a>              |
|                  | 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, to check if the Flash is empty, or to exit back to the Main Menu. The Flash Read Menu is shown in Figure 4-3.

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

UG511\_04\_03\_121208

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, 32 bytes of data is read starting from the address 0xFCCC0000. The example output was created while the software application was running on the PPC440 hardware system. The output from the application running on the MicroBlaze processor hardware system will be the same except the addressing will be offset from 0x8C000000 instead of 0xFC000000.

```
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 (0-1FFFFFF): CC0000
Enter number of bytes to read (0-128): 32

0xFCCC0000: 00 01 02 03 04 05 06 07
0xFCCC0008: 08 09 0A 0B 0C 0D 0E 0F
0xFCCC0010: 10 11 12 13 FF FF FF FF
0xFCCC0018: FF FF FF FF FF FF FF FF
```

UG511\_04\_04\_121208

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 8 seconds to run for the PPC440 system and 28 seconds to run for the MicroBlaze system.

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. The example output was created while the software application was running on the PPC440 hardware system. The output from the application running on the MicroBlaze processor hardware system will be the same except the addressing will be offset from 0x8C000000 instead of 0xFC000000.

```
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
```

UG511\_04\_05\_121208

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 0xFC100000; Data is 0
Flash is not erased at 0xFC100001; Data is 1
Flash is not erased at 0xFC100002; Data is 2
.....
Flash is not erased at 0xFCCC0000; Data is 0
Flash is not erased at 0xFCCC0001; Data is 1
Flash is not erased at 0xFCCC0002; Data is 2
Flash is not erased at 0xFCCC0003; Data is 3
Flash is not erased at 0xFCCC0004; Data is 4
Flash is not erased at 0xFCCC0005; Data is 5
Flash is not erased at 0xFCCC0006; Data is 6
Flash is not erased at 0xFCCC0007; Data is 7
Flash is not erased at 0xFCCC0008; Data is 8
Flash is not erased at 0xFCCC0009; Data is 9
Flash is not erased at 0xFCCC000A; Data is A
Flash is not erased at 0xFCCC000B; Data is B
Flash is not erased at 0xFCCC000C; Data is C
Flash is not erased at 0xFCCC000D; Data is D
Flash is not erased at 0xFCCC000E; Data is E
Flash is not erased at 0xFCCC000F; Data is F
Flash is not erased at 0xFCCC0010; Data is 10
Flash is not erased at 0xFCCC0011; Data is 11
Flash is not erased at 0xFCCC0012; Data is 12
Flash is not erased at 0xFCCC0013; Data is 13
.....
Flash is not erased at 0xFDC00000; Data is FE
Flash is not erased at 0xFDC00001; Data is ED
.....
Flash is not completely empty
31 total bytes of Flash were found to not be empty
The first 25 locations of Flash that were not empty were printed out
```

UG511\_04\_06\_121208

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:
```

UG511\_04\_07\_121208

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 for which the user must supply values:

- 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 an example output from the function when the bytes to write are already empty. In Figure 4-8, 64 bytes of data is being written to address 0xFC100000. Figure 4-9 shows an example output from the function when the bytes to write have previously stored data. In Figure 4-9, 20 bytes of data were inputted to be written to address 0xFDA00000, but 6 bytes of Flash were found to have previous data. To proceed with the write, the user must allow the entire block (0xFDA00000-0xFDA1FFFF) to be erased. The example output was created while the software application was running on the PowerPC 440 processor hardware system. The output from the application running on the MicroBlaze processor hardware system will be the same, except that the addressing will be offset from 0x8C000000 instead of 0xFC000000

```
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 (0-1FFFFFF): 100000
Enter number of bytes to write (0-2048): 64

Completed writing incrementing numbers to Flash
```

UG511\_04\_08\_121208

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 (0-1FFFFFF): 1A00000
Enter number of bytes to write (0-2048): 20
In the desired write location, 6 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
```

UG511\_04\_09\_121208

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 for which the user must supply values:

- 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 0x21436587 is being written 3 times starting at address 0xFDA00000. 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 0xFEDCBA was input to be written 2 times to the address 0xFC100000, but 6 bytes of Flash were found to have previous data. To proceed with the write, the user must allow the entire block (0xFC100000-0xFC11FFFF) to be erased. The example output was created while the software application was running on the PowerPC 440 processor hardware system. The output from the application running on the MicroBlaze processor hardware system will be the same except that the addressing will be offset from 0x8C000000 instead of 0xFC000000.

```
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 (0-1FFFFFF): 1A00000
Enter 0-16 bytes of data to write: 21436587
Enter number of times to write the data (0-32): 3
Completed writing bytes of data to Flash
```

UG511\_04\_10\_121208

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 (0-1FFFFFF): 100000
Enter 0-16 bytes of data to write: FEDCBA
Enter number of times to write the data (0-32): 2
In the desired write location, 6 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
```

UG511\_04\_11\_121208

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:
```

UG511\_04\_12\_121208

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 in which the specified bytes reside.

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

- 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.

An example input and output from this function is shown in [Figure 4-13](#). In the figure, it is specified to erase 2048 bytes at address 0xFC010000. However, because of Flash memory characteristics, the entire block in which these bytes reside, will be erased, namely the address range 0xFC000000-0xFC01FFFF. The example output was created while the software application was running on the PPC440 hardware system. The output from the application running on the MicroBlaze processor hardware system will be the same except that the addressing will be offset from 0x8C000000 instead of from 0xFC000000.

```
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 (0-1FFFFFF): 10000
Enter number of bytes to erase (0-393216): 2048
Erased the Flash memory contents successfully
```

UG511\_04\_13\_121308

**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 (256 blocks) is specified to be erased, this function will take approximately 4 minutes to complete.

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

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

An example input and output from this function is shown in [Figure 4-14](#). In the figure, blocks 35-104 have been erased, which corresponds to the address range 0xFC4600000-0xFCD1FFFF. The example output was created while the software application was running on the PowerPC 440 processor hardware system. The output from the application running on the MicroBlaze processor hardware system will be the same except the addressing will be offset from 0x8C000000 instead of 0xFC000000.

```
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 (0-255) to erase: 35
Enter ending block (35-255) to erase: 104
Erasing blocks 35-104 of Flash...
.....
Erased the Flash memory contents successfully
```

UG511\_04\_14\_121208

**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 4 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
```

UG511\_04\_15\_121208

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

