XAPP1106 (v1.2) January 27, 2009

Using and Creating Flash Files for the MicroBlaze Development Kit - Spartan-3A DSP 1800A Starter Platform

Author: Sundararajan Ananthakrishnan/Casey Cain

Abstract

This application note describes the files for programming the serial Flash memory and the StrataFlash memory for the MicroBlaze™ Development Kit - Spartan®-3A DSP 1800A Starter Platform. The reference system is created to execute the HelloWorld software application to run from the Serial Flash using SPI configuration mode and the BlueCat Linux image from the StrataFlash in BPI configuration mode. All files needed to run the BlueCat Linux reference system from the StrataFlash memory and the HelloWorld application from the Serial Flash are provided for the user with the Development Kit. This application note describes how to use the provided files, as well as create new files, to run the reference system successfully from the Flash memories.

Introduction

The SP3ADSP1800A board supports a variety of FPGA configuration options. One of the options is to program the on-board Intel StrataFlash parallel NOR Flash PROM, then configure the FPGA from the image stored in the Flash PROM using the BPI configuration mode. The other option is to program the on-board Serial Flash, then configure the FPGA from the image stored in the Serial Flash PROM using the SPI configuration mode.

Hardware and Software Requirements

The hardware and software requirements are:

- Xilinx Spartan-3A DSP 1800A Starter Platform
- Xilinx Parallel Cable IV (PC4) or Platform Cable USB with flyleads
- RS232 serial cable and serial communications utility, such as HyperTerminal
- Xilinx Platform Studio 10.1.3
- Xilinx Integrated Software Environment (ISE®) 10.1.3

Configuration Using the Intel StrataFlash

To boot the BlueCat Linux image from parallel Flash, the Linux image, the root file system, and a bootloader application must be programmed into Flash. The bootloader application copies the Linux image from Flash to DDR2 memory and boots BlueCat Linux.

The files required to run the BlueCat Linux demonstration out of the StrataFlash memory are provided with the Development Kit, and are documented by UG486.

The steps for programming the BlueCat Linux image into parallel Flash are detailed in the subsequent sections.

After programming all the files into the Flash memory as directed, the Flash memory will have the address map shown in Table 1. The files do not neccessarily 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 1: **Parallel Flash Address Mapping**

File	Start Address Offset	End Address Offset
Bootloader/System Configuration (BIN)	0x00000000	0x0000FFFF
BlueCat Linux Image (KDI)	0x00100000	0x0029FFFF
Flash File System (JFFS2)	0x002A0000	0x00C7FFFF
Unused Space	0x00C80000	0x00FFFFFF

Programming the Parallel Flash with the Provided BlueCat Linux Files

This section includes details for using the Flash files provided in the reference system to boot the BlueCat Linux demonstration out of the StrataFlash.

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 BlueCat Linux KDI Image into Flash

To run the BlueCat Linux demonstration properly, the KDI image must be loaded into the StrataFlash. To load the KDI image into the StrataFlash memory, follow these steps:

1. Unzip the contents of the reference system into a directory.
2. Open the reference system project in XPS.
3. Connect the USB or Parallel IV programming cable, the serial cable, and the power supply to the S3ADSP1800A development board. Power on the board.
4. Select **Project** → **Launch EDK Shell** In XPS to launch an EDK shell**.**
5. Change directories to the `ready_for_download` directory in the EDK shell.
6. Use iMPACT to download the bitstream by using the following command:

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

7. Select **Device Configuration** → **Program Flash Memory** in XPS.

8. In the Program Flash Memory window, choose the file that is programmed to be the provided KDI image file, `/bclinux_images/s3adsp_devl_kit_demo_flash.kdi` under the project root directory. Change the program offset to `0x00100000`. This program offset is the address where the provided BlueCat Linux bootloader is programmed to find the KDI image. The Program Flash Memory settings for the KDI image are shown in Figure 1.

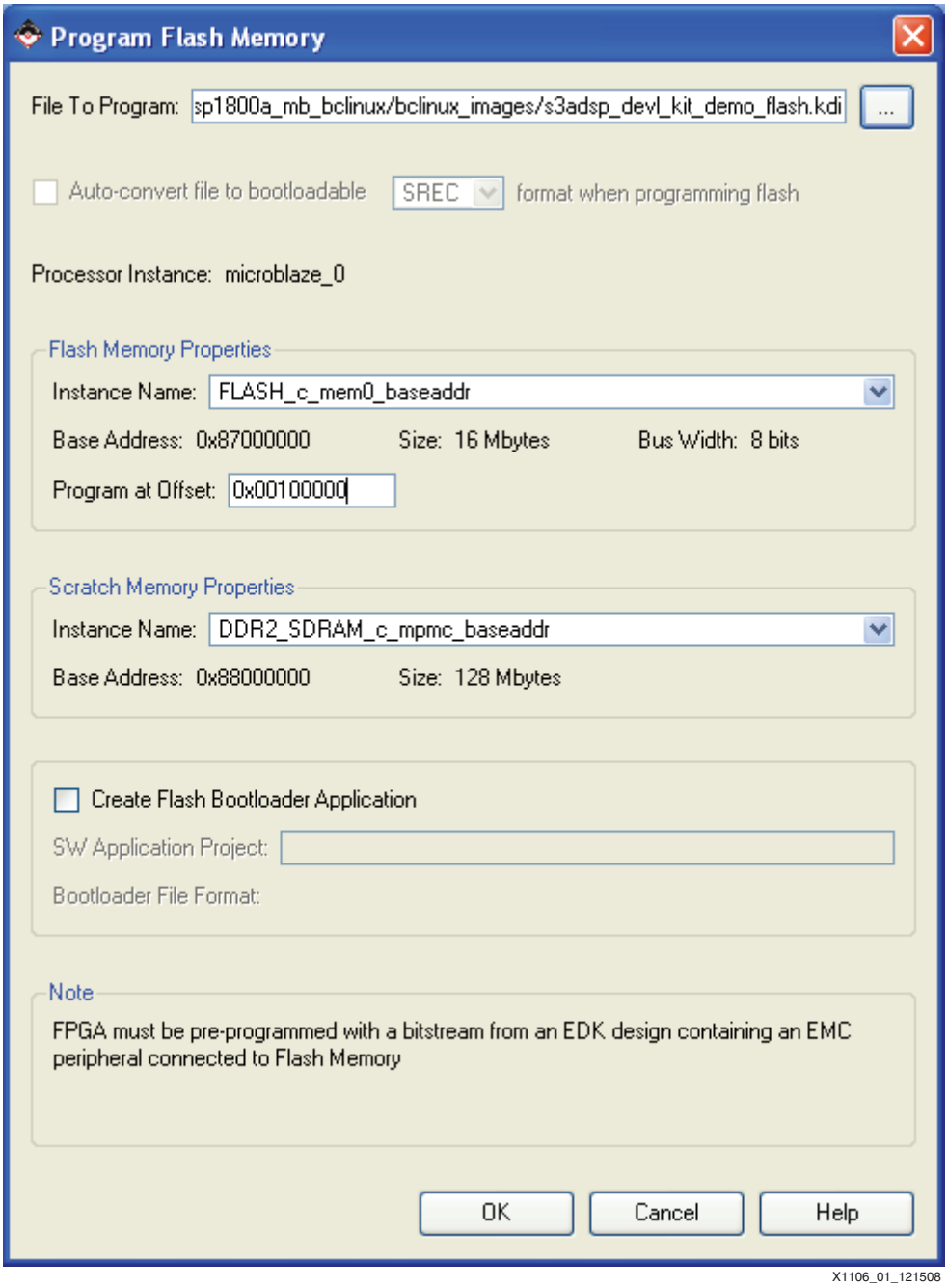


Figure 1: **Program Flash Memory Dialog Box for the Pre-built BlueCat Linux Image**

9. Click **OK** to program the Flash memory with the BlueCat Linux image.

Note: The Program Flash Memory application will take a long time as the application must erase the flash contents and program the approximately 1.5 MB image.

Programming the Bitstream and Bootloader BIN File

The BIN file is used to program the Flash with the system bitstream and bootloader loaded in the bitstream. Follow the subsequent steps to program the BIN file for the system.

1. Select **Device Configuration** → **Program Flash Memory** in XPS.
2. With the XPS Flash programmer, program the provided binary bootloader bitstream `\ready_for_download\Flash_files\bootloader_bclinux.bin` to offset `0x00000000` of the Flash Memory. The Program Flash Memory settings for the BIN file are shown in Figure 2.

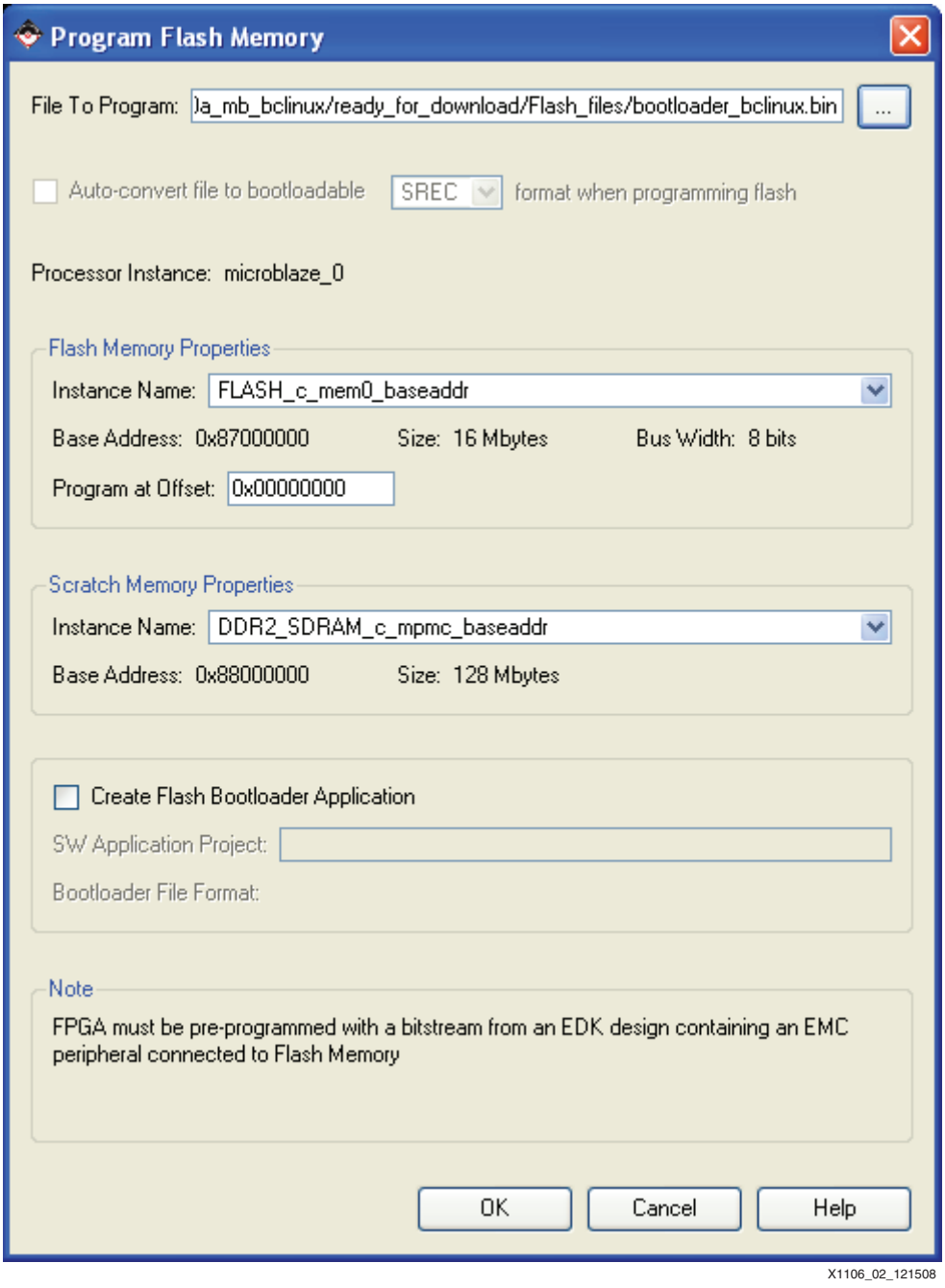


Figure 2: **Program Flash Memory Dialog Box for the Pre-built Binary Bootload Bitstream**

3. Click the **OK** to 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 approximately 1 MB BIN file.

Programming the Root File System

The Linux image that was downloaded into Flash in the previous steps is built to boot with a JFFS2 root file system which allows persistent storage in the Flash memory. For the Linux image to boot, program the root file system into the parallel Flash as described in the subsequent section.

1. In an EDK shell, invoke XMD and connect to the processor by using the following command:

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

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

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

3. 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.

4. With the FlashRWE program, erase blocks 21-99 of Flash. These blocks are the location where the BlueCat Linux kernel image will expect the JFFS2 root file system. To erase the blocks, perform the following steps. The steps appear also in Figure 3.

   a. Enter 3 at the Main Menu.

   b. In the Flash Erase Menu, enter 2.

   c. Enter 21 as the starting block.

   d. Enter 99 as the ending block.

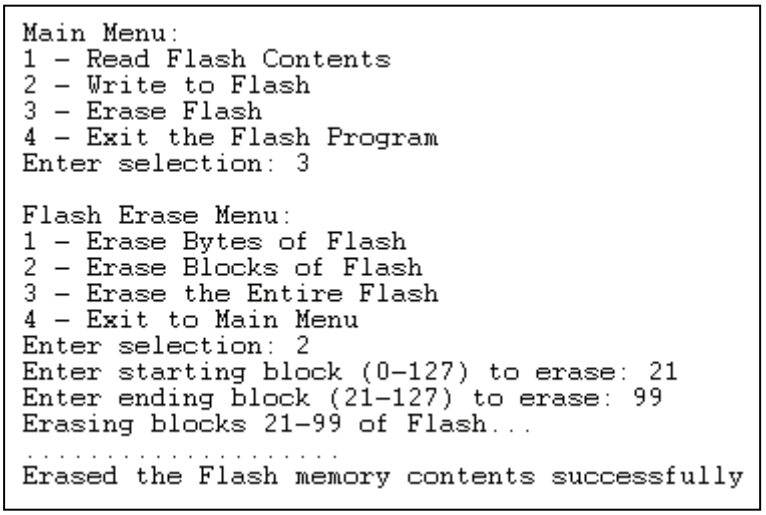


Figure 3: **Erase Blocks 21-99 of Flash**

5. In XMD, stop and reset the processor, then exit XMD.

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

6. Select **Device Configuration** → **Program Flash Memory** in XPS.

7. In the Program Flash Memory dialog box, choose the file to program to be `/bclinux_images/s3adsp_devl_kit_demo_flash.jffs2`. Enter the offset to `0x002A0000`. The external DDR2 memory is set as the Scratch Memory. The Program Flash Memory settings are shown in Figure 4.

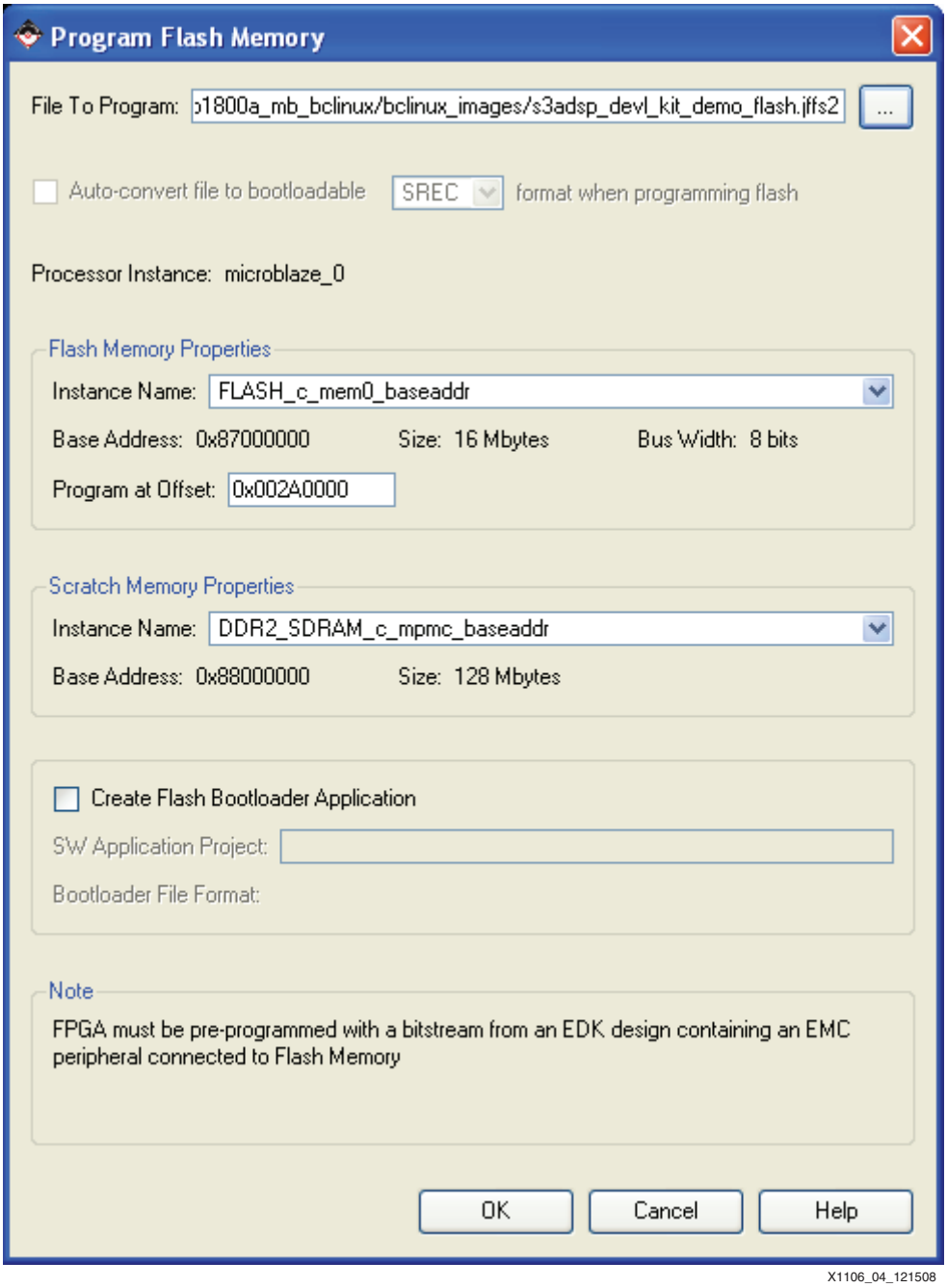


Figure 4: **Program Flash Memory Dialog Box for the Pre-built JFFS2 Root File System**

8. Click the **OK**. 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 approximately 7 MB JFFS2 file.

Running the Design

1. The BlueCat linux image will be loaded into the StrataFlash memory and the BlueCat Linux demonstration will be ready to be run in BPI mode. Proceed to the section, "Running the BlueCat Linux Demonstration out of StrataFlash" section on page 14, to run the demonstration.

Creating and Programming New Parallel Flash Files for BlueCat Linux

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

Instead of using the existing bootloader file from the `/ready_for_download/Flash_files` folder under the root directory, new bootloader files can be created using the EDK tools.

To load a new BlueCat Linux image KDI file into the StrataFlash and to create new bootloader files, follow these steps:

1. Open the reference system project in XPS.
2. Connect the USB programming cable, the serial cable, and the power supply to the SP3ADSP1800A starter platform. Apply power to the board.
3. The target board must be configured with the project bitstream before XPS can program the KDI file into the Flash memory. Select **Device Configuration** → **Download Bitstream** in XPS.
4. Select **Device Configuration** → **Program Flash Memory** in XPS.

5. In the Program Flash Memory window, choose the file that is to programmed to be the provided KDI image file, `/bclinux_images/s3adsp_devl_kit_demo_flash.kdi` under the project root directory. Change the program offset to `0x00100000`. 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 for the KDI image are shown in Figure 5.

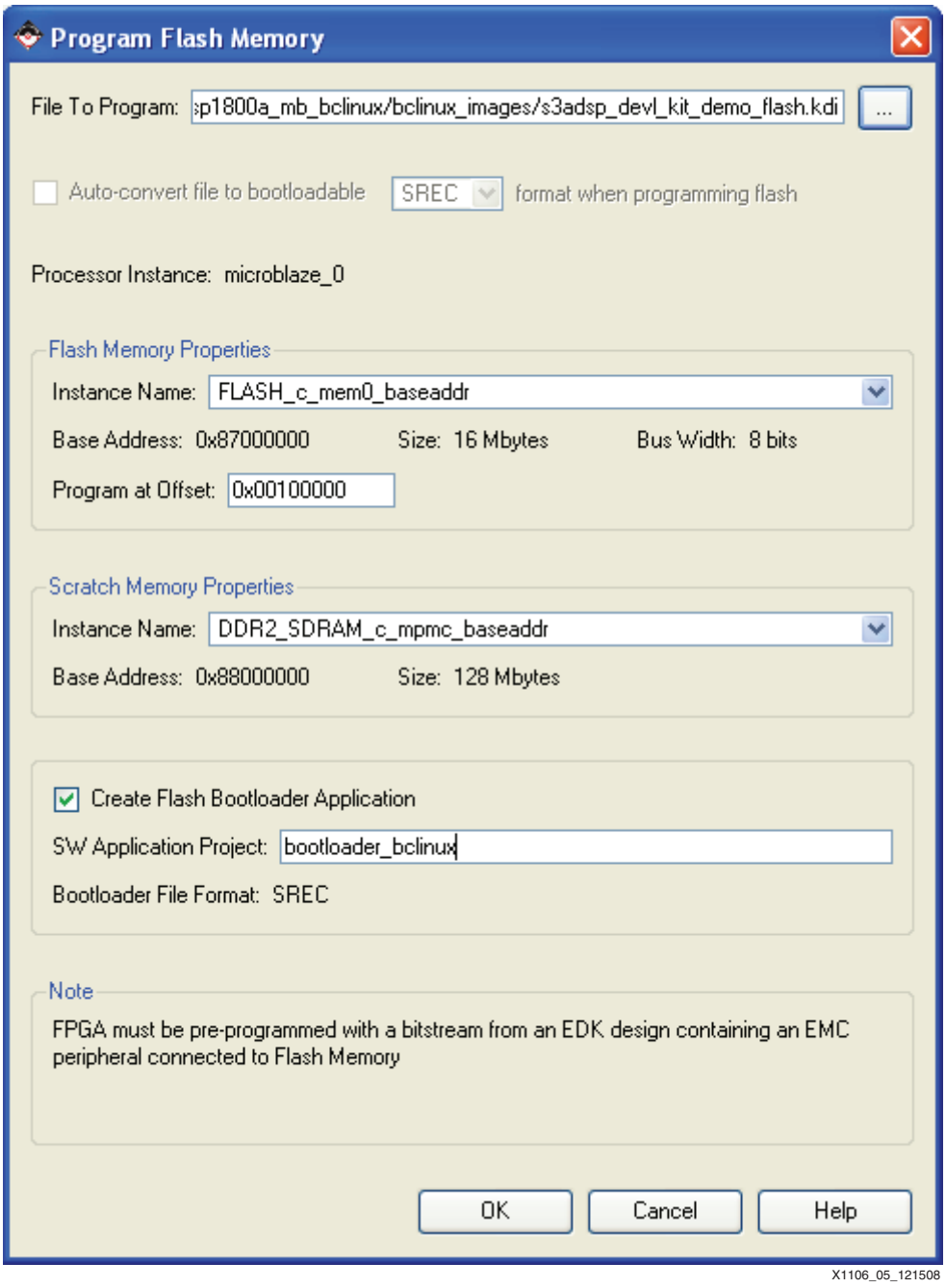


Figure 5: **Program Flash Memory for KDI File with Bootloader Creation**

6. Click **OK** to program the flash memory with the BlueCat Linux image and to create a bootloader software application project.

Note: The Program Flash Memory application will take a long time as the application must program the approximately 1.5 MB image file.

7. After creating the bootloader files, add the following line in the `bootloader.c` file to get it to compile:

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

8. It is strongly suggested that bootloader be disabled from displaying its progress by commenting out the following line in the generated file, `bootloader.c`.

   ```
   #define VERBOSE
   ```

   The line to comment out to allow non-verbose bootloading is shown in Figure 6.

```
40
41 #include "xparameters.h"
42
43 /* Defines */
44 #define CR       13
45
46 /* Comment the following line, if you want a smaller and faster bootloader which will be silent */
47 /* #define VERBOSE */
48
```

X1106_06_121508

Figure 6: **Code to Select Non-Verbose Bootloading**

9. The bootloader must be modified to copy the KDI image from flash into DDR for the BlueCat Linux demonstration to operate fully. 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 pieces of code shown in Figure 7.

   a. Code is required to define the location in flash where the KDI image resides and the location in DDR2 of the KDI image. Set the KDI_FLASH_LOC parameter to the location in flash memory of where the KDI image will be placed. Set the KDI_DDR_LOC parameter to the location in DDR memory where the KDI image is to be copied. Set the KDI_LENGTH parameter to the length of the KDI image in bytes.

```
49
50 /* Add for copying the contents of flash to DDR */
51 #define KDI_DDR_LOC XPAR_DDR2_SDRAM_MPMC_BASEADDR   //Location to put KDI into the DDR
52 #define KDI_FLASH_LOC XPAR_FLASH_MEM0_BASEADDR + 0x00100000     //Location of the KDI in FLASH
53 #define KDI_LENGTH 0x0182000            //Size (in bytes) of the KDI
54
55
56 /* Declarations */
57 static void display_progress (uint32_t lines);
58 static int8_t load_exec ();
59 static int8_t flash_get_srec_line (uint8_t *buf);
60 extern int srec_line;
61
62 /* Declarations for copying the contents from flash to DDR */
63 uint8_t * kdi_ddr_ptr;
64 uint8_t * kdi_flash_ptr;
65 static uint32_t ddr_loc = KDI_DDR_LOC;
66 static uint32_t flash_loc = KDI_FLASH_LOC;
67
```

X1106_06_012709

Figure 7: **BlueCat Linux Bootloader Code Definitions and Declarations**

b. Comment the SREC function **load_exec** in the source file. Because KDI image is loaded as is to the FLASH memory, these functions are not required. Code is also required for the bootloader to copy the KDI image from flash into DDR when the bootloader runs. This code is shown in Figure 8. After modifying the C file, be sure to recompile the software application.

```
/*
    srinfo.sr_data = sr_data_buf;

    while (!done) {
        if ((ret = flash_get_srec_line (sr_buf)) != 0)
            return ret;

        if ((ret = decode_srec_line (sr_buf, &srinfo)) != 0)
            return ret;

#ifdef VERBOSE
        display_progress (srec_line);
#endif
        switch (srinfo.type) {
            case SREC_TYPE_0:
                break;
            case SREC_TYPE_1:
            case SREC_TYPE_2:
            case SREC_TYPE_3:
                memcpy ((void*)srinfo.addr, (void*)srinfo.sr_data, srinfo.dlen);
                break;
            case SREC_TYPE_5:
                break;
            case SREC_TYPE_7:
            case SREC_TYPE_8:
            case SREC_TYPE_9:
                laddr = (void*)srinfo.addr;
                done = 1;
                ret = 0;
                break;
        }
    }
*/
/* Copy the KDI from FLASH into the DDR */
    kdi_ddr_ptr = (uint8_t*)ddr_loc;
    kdi_flash_ptr = (uint8_t*)flash_loc;
    memcpy (kdi_ddr_ptr, kdi_flash_ptr, KDI_LENGTH);


#ifdef VERBOSE
    print ("\r\nExecuting program starting at address: ");
    putnum ((uint32_t)ddr_loc);
    print ("\r\n");
#endif

laddr = (void*)ddr_loc;
    (*laddr)();
```

X1106_08_121508

Figure 8: **BlueCat Linux Bootloader Code to Copy the KDI Image**

10. 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 BRAMs** and then by selecting **Device Configuration** → **Update Bitstream** in XPS.
11. In the `FLASH_BURN` directory, run the `make_bpi_up.bat` file to create a binary version of the bootloader bitstream.

Note: The `make_bpi_up.bat` file is currently set to use the bitstream in `<project root directory>/implementation/download.bit`. The batch file can be edited to use a different bitstream.

12. With the XPS Flash programmer, program in the binary bootloader bitstream, `<project root directory>/bclinux_images/bootloader_bclinux.bin`, to offset `0x00000000` of the Flash Memory as shown in Figure 9.

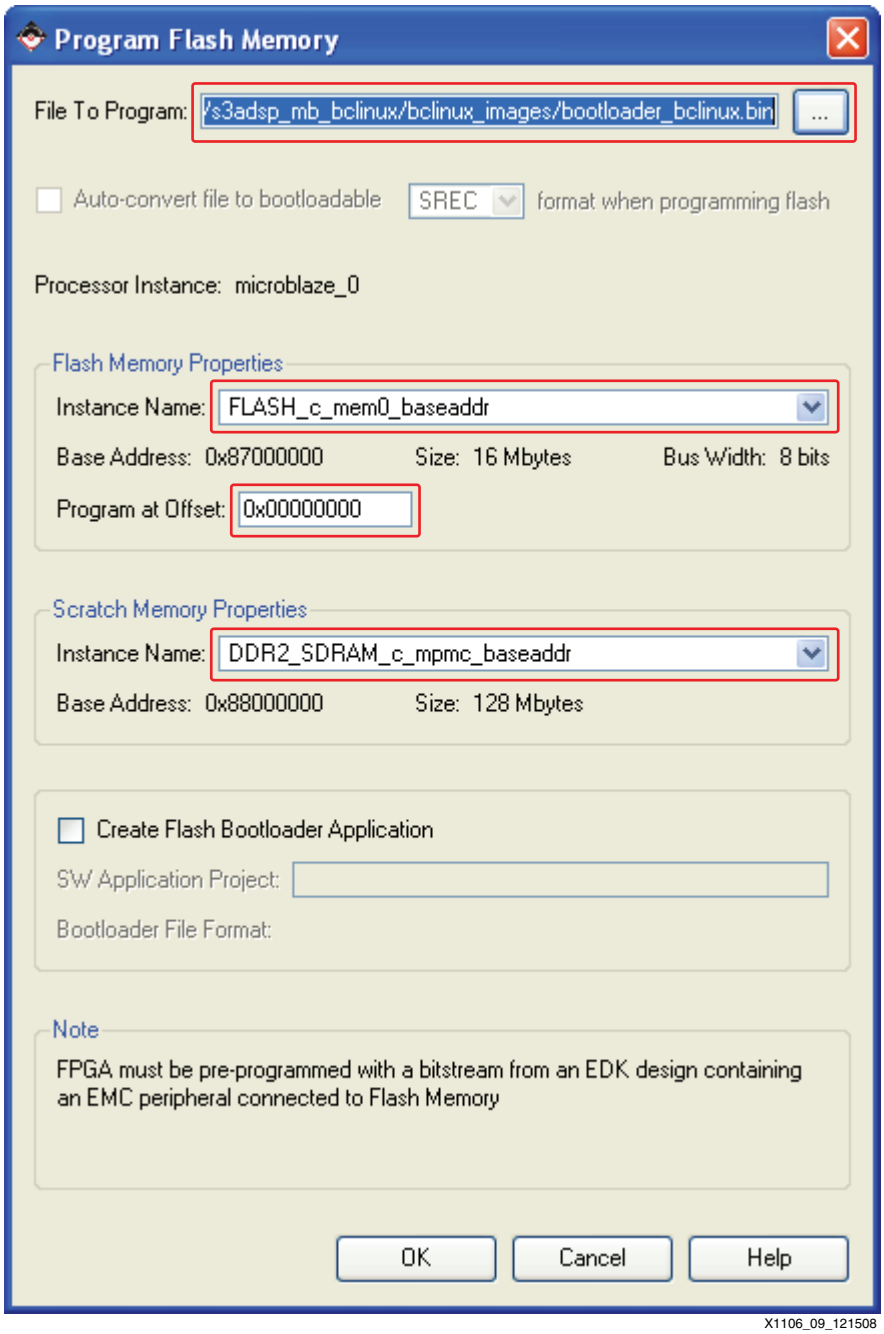


Figure 9: **Program Flash Memory Window for the Binary Bootload Bitstream**

Programming the Root File System

The Linux image that was downloaded into Flash in the previous steps is built to boot with a JFFS2 root file system which allows persistent storage in the Flash memory. For the Linux image to boot, the root file system must be programmed into the parallel Flash, as detailed in this section.

1. In an EDK shell, invoke XMD and connect to the processor by using the following command:

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

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

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

3. Start the FlashRWE software application running by using the following XMD command:

   ```
   XMD% run
   ```

   After the FlashRWE software application runs, the HyperTerminal displays the Main Menu.

4. Erase blocks 21-99 of Flash with the FlashRWE program. These blocks are the location where the BlueCat Linux kernel image expects the JFFS2 root file system. To erase the blocks, perform the following steps. The steps are shown also in Figure 10.

   a. Enter 3 at the Main Menu.

   b. In the Flash Erase Menu, enter 2.

   c. Enter 21 as the starting block.

   d. Enter 99 as the ending block.

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

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

X1106_10_121508

Figure 10: **Erase Blocks 21-99 of Flash**

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

   ```
   XMD% stop
   ```

   ```
   XMD% rst
   ```

   ```
   XMD% exit
   ```

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

7. In the Program Flash Memory dialog box, choose the file, `/bclinux_images/s3adsp_devl_kit_demo_flash.jffs2`, as the file to be programmed. Enter the offset to `0x002A0000`. The external DDR2 memory is set as the Scratch Memory. The Program Flash Memory settings are shown in Figure 11.

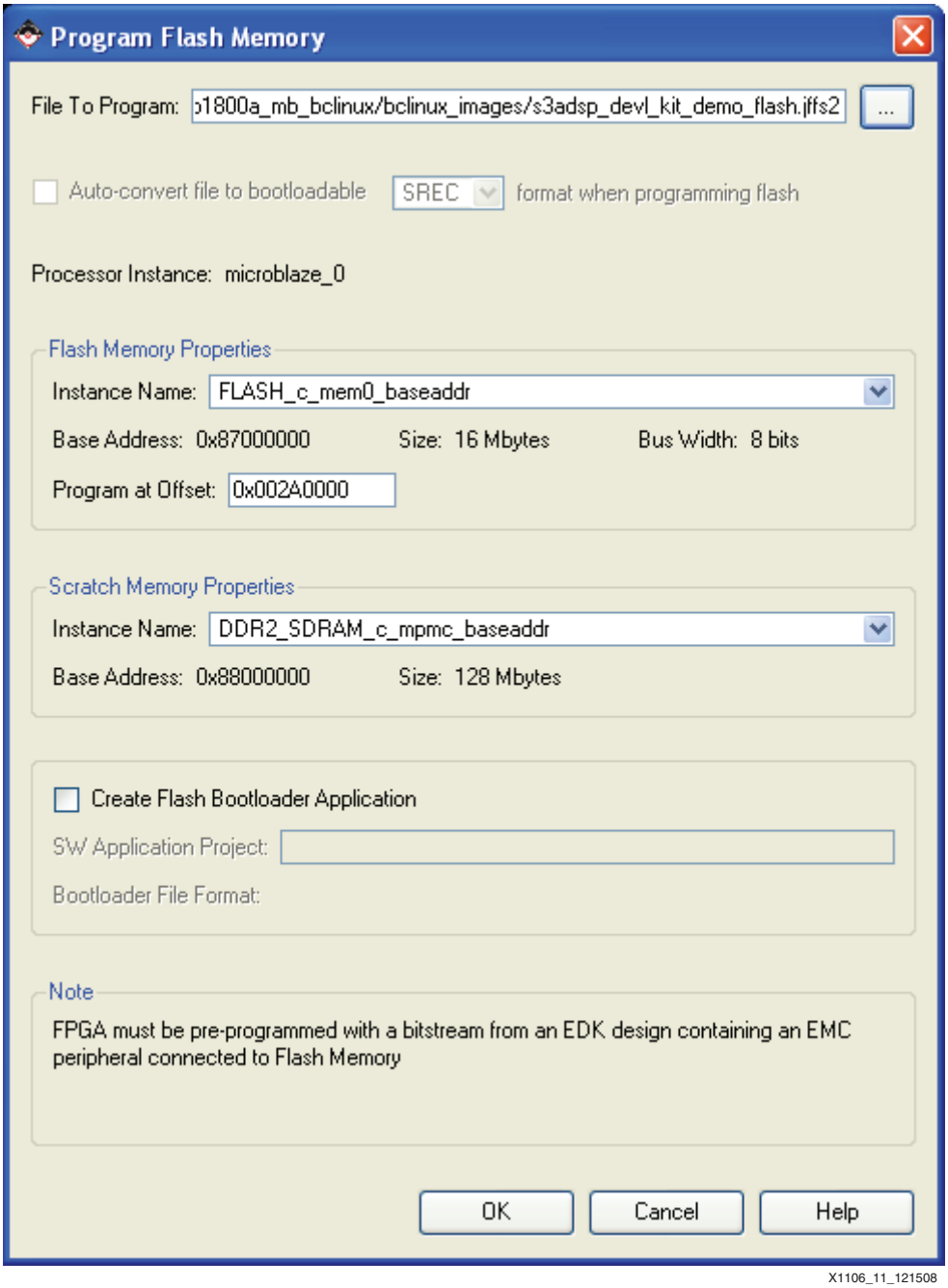


Figure 11: **Program Flash Memory Dialog Box for the Pre-built JFFS2 Root File System**

8. Click the **OK** to program the flash memory with the JFFS2 file.

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

Running the Design

1. The BlueCat linux image will be loaded into the StrataFlash memory and the BlueCat Linux demonstration will be ready to be run in BPI mode. Proceed to the section, "Running the BlueCat Linux Demonstration out of StrataFlash" section on page 14, to run the demonstration.

Running the BlueCat Linux Demonstration out of StrataFlash

After the files are loaded into the StrataFlash memory as detailed in the previous sections, the BlueCat Linux demonstration can be run out of flash. The subsequent section provides the instructions for running the BlueCat Linux demonstration.

Preparing to Run the BlueCat Linux Demonstration out of StrataFlash

Ensure that the Spartan-3A DSP 1800A development board is set up properly with the serial cable and power supply connected. Set up aserial terminal program, such as HyperTerminal, to view the output of the demonstrations. Set the serial terminal program as shown in Figure 12: Baud Rate of **115200**, Data Bits to **8**, Parity to **None**, and Flow Control to **None**.

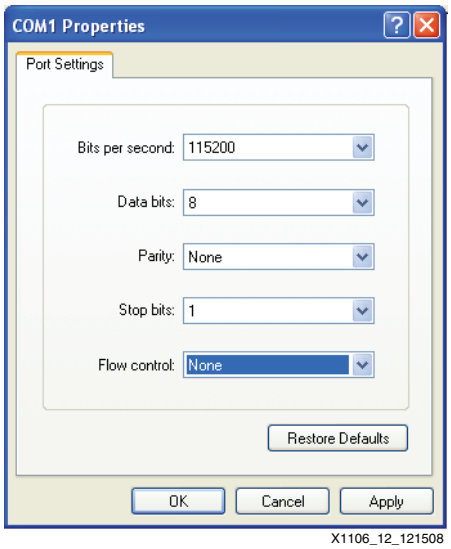


Figure 12: **HyperTerminal Settings for the Demonstrations**

Running the BlueCat Linux Demonstration in BPI Mode

To run the files out of the StrataFlash in BPI mode, set the JP9 configuration jumpers to BPI mode, for example, M0 and M2 closed. Once the configuration jumpers are set properly, apply power to the development board or press the PROG button to start the BlueCat Linux demonstration.

After BlueCat Linux boots, the HyperTerminal output will be as shown in Figure 13. Once the BlueCat Linux kernel boots up, log into the BlueCat Linux by using the user name **root.**

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

myhostname login: root
```

X1106_13_121508

Figure 13: **BlueCat Linux Boot Output**

Configuration Using the Serial Flash

Similar to the traditional configuration memories, SPI serial flash memories must be loaded with the configuration data. SPI serial flash memories have a single interface for programming, however, there are multiple methods to deliver the data to this interface.

There are four primary delivery to program an SPI serial flash through the SPI interface:

1. Direct in-system programming (SPI direct interface connect)
2. Indirect in-system programming or ISP (Xilinx iMPACT, JTAG tool vendor or custom solution)
3. Third-party programmers (off-board programming)
4. Embedded processor (in-system programming)

Use the direct in-system programming method (method 1) to program the SPI serial flash memory on the Spartan-3A DSP 1800A starter platform.

Programming the Serial Flash with the Provided Helloworld Files

Follow the subsequent steps to load the Helloworld bitstream provided with the reference system.

1. Edit the `burn_intel_s33.bat` file in the `FLASH_BURN` directory to select the pre-built `helloworld.bit` bitstream in the `ready_for_download\Flash_files` directory as mentioned below:

```
REM set bitstream= ../implementation/download

set bitstream = ../ready_for_download/Flash_files/helloworld
```

2. Follow the steps step 1 to step 5 and then steps step 13 to step 17 in the "Creating and Programming New Serial Flash Files for the HelloWorld Application" section on page 16 to load the serial flash with the pre-built HelloWorld demonstration.

Creating and Programming New Serial Flash Files for the HelloWorld Application

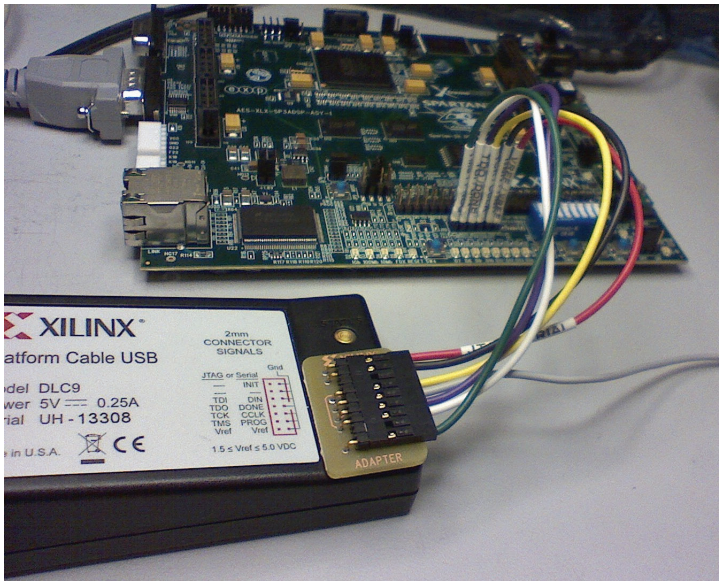
This procedure stores the HelloWorld application bitstream in the serial flash and then configures the Spartan-3A DSP FPGA from that flash. If the user prefers a different BIT file, change the location of the input BIT file used in the `burn_intel_s33.bat` batch file found in step 9 below.

1. Turn Power OFF.
2. Disconnect the JTAG cable from J2 or J4. Install flyleads on the JTAG cable, if this has not already been done.
3. Plug JTAG cable flyleads onto the SPI connector (J10). Connect the leads as defined in Table 2..

Table 2: **Connection of JTAG Cable Flyleads to Board Connector J10**

Flyleads	Board J10	Flyleads	Board J10
VREF	VCC	TDO	MISO
GND	GND	TDI	MOSI
TCK	SCK	TMS	SEL

As shown in Figure 14, the 2nd row of J10 is not used during this procedure



X1106_14_121508

Figure 14: **JTAG Cable Flyleads to Board Connector J10**

4. Add a jumper to JP7 to ground the FPGA PROG_B pin.
5. Turn power ON.
6. Open the reference system project in XPS.
7. In the Applications tab, select the **helloworld** project for BRAM initialization by right-clicking on the project, then select **Mark to Initialize BRAMs**. Ensure that no other applications are marked for BRAM initialization.
8. Select **Device Configuration** → **Update Bitstream** to generate the MicroBlaze hardware platform, build the Board Support Package (BSP), compile the project, and initialize the bitstream with the HelloWorld application.
9. A batch file, `burn_intel_s33.bat`, is provided in the `FLASH_BURN` directory to simplify the burning of the Intel S33 serial flash using the XIP and iMPACT programs. iMPACT tool is used in batch mode to convert the bitstream to an MCS file compatible with SPI configuration. XIP is then used to burn this MCS to the Intel S33 serial flash.
10. Open the `burn_intel_s33.bat` batch file for editing. Edit the batch file to point to the generated file, `download.bit`, in the `implementation` directory.

Note: Ensure that the following line that selects to the generated bitstream is *uncommented* and that he line to select the helloworld is *commented* using REM as shown below:

```
set bitstream= ../implementation/download

REM set bitstream = helloworld
```

11. If the PC4 is being used rather than the Platform Cable USB, change the XIP argument from **`-select_cable 2`** to **`-select_cable 6`**
12. Close the batch file.
13. Double-click the `burn_intel_s33.bat` batch file.
14. Verify that the programming was successful as indicated by the following statement near the end of batch file command prompt

```
--> Total byte mismatches [0] (see [verify.txt])
```

15. Turn off power to the board.
16. Unplug the JTAG cable flyleads.
17. Remove the JP7 jumper to release PROG_B.

This will program the serial flash memory with the Helloworld demo application. Proceed to the section, "Running the Helloworld Demonstration Out of Serial Flash" section on page 18, to run the demonstration.

Running the Helloworld Demonstration Out of Serial Flash

Once all the files are loaded into the serial flash memory, the Helloworld demonstration can be run out of the serial flash. The subsequent section provides details for running the Helloworld demonstration out of the serial flash.

Preparing to Run the Helloworld Demonstration out of Serial Flash

Ensure that the SP3ADSP1800A development board is set up properly with the serial cable and power supply connected. Set up a serial terminal program, such as HyperTerminal, to view the output of the demonstrations. Set the serial terminal program as shown in Figure 15: Baud Rate of **115200**, Data Bits to **8**, Parity to **None**, and Flow Control to **None**.

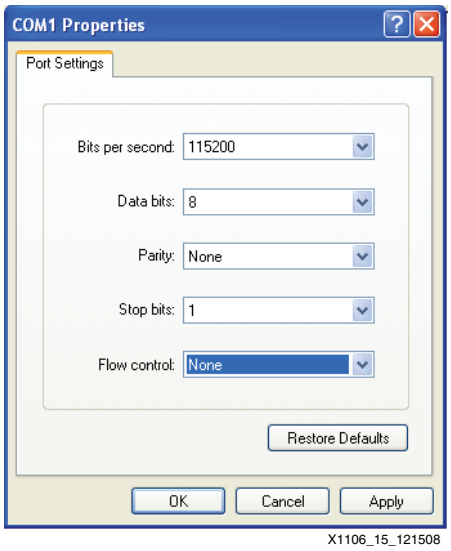


Figure 15: **Hyperterminal Settings**

Running the Helloworld Demonstration in SPI Mode

To run the files out of the StrataFlash in SPI mode, set the J9 configuration jumpers to SPI mode. for example, M1 and M2 closed. Apply power to the development board or press the PROG button to start the Helloworld demonstration. Upon power up, the FPGA is configured from the image stored in the Intel S33 Serial Flash. The DONE LED (D1) lights and **Hello World** application is displayed on the HyperTerminal as shown in Figure 16.

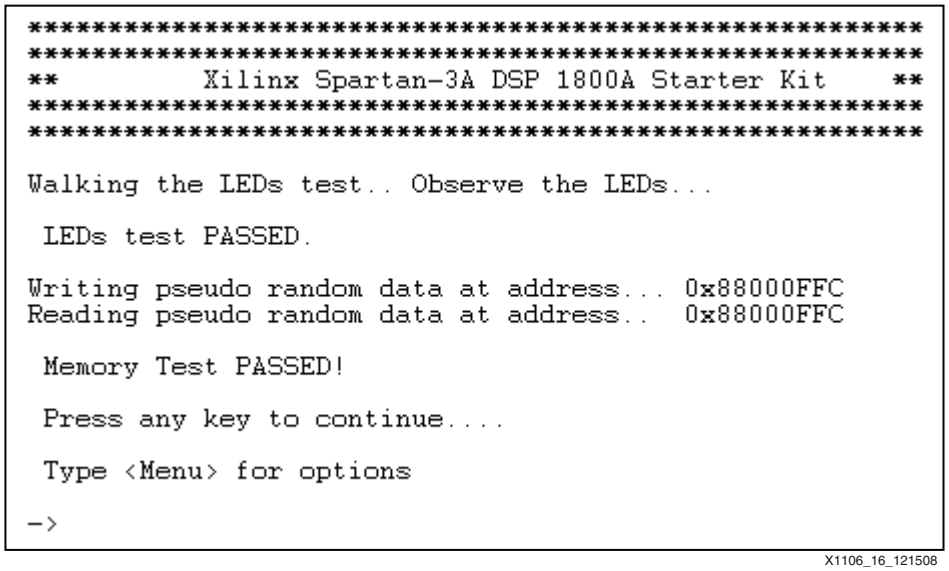


Figure 16: **Helloworld Output**

References

1. UG486 MicroBlaze Development Kit Spartan-3A DSP 1800A Edition User Guide
2. UG485 Getting Started with the MicroBlaze Development Kit - Spartan-3A DSP 1800A Edition

Conclusion

The BlueCat Linux reference system is provided with the MicroBlaze Development Kit - Spartan-3A DSP Edition. This application note describes how to use the files provided and create new files for programming the Intel StrataFlash PROM with the BlueCat Linux demonstrations in the BPI mode and the Intel Serial Flash PROM with the Helloworld demonstration in the SPI mode.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
6/17/08	1.0	Initial Xilinx release.
12/19/08	1.1	Updated to 10.1.3. Added JFFS2 root file system information.
1/27/09	1.2	Updated figure 7. Minor text updates.

Notice of Disclaimer

Xilinx is disclosing this Application Note to you “AS-IS” with no warranty of any kind. This Application Note is one possible implementation of this feature, application, or standard, and is subject to change without further notice from Xilinx. You are responsible for obtaining any rights you may require in connection with your use or implementation of this Application Note. XILINX MAKES NO REPRESENTATIONS OR WARRANTIES, WHETHER EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL XILINX BE LIABLE FOR ANY LOSS OF DATA, LOST PROFITS, OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR INDIRECT DAMAGES ARISING FROM YOUR USE OF THIS APPLICATION NOTE.