

# BlueCat Linux Board Support Guide

---

BlueCat Linux 4.0

DOC-0483-02

*For Intel IXDP1200 Advanced Development Platform*

Product names mentioned in the *BlueCat Linux Board Support Guide for Intel IXDP1200 Advanced Development Platform* are trademarks of their respective manufacturers and are used here for identification purposes only.

Copyright ©1987-2002, LynuxWorks, Inc. All rights reserved.  
U.S. Patents 5,469,571; 5,594,903

Printed in the United States of America.

All rights reserved. No part of *BlueCat Linux Board Support Guide for Intel IXDP1200 Advanced Development Platform* may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photographic, magnetic, or otherwise, without the prior written permission of LynuxWorks, Inc.

LynuxWorks, Inc. makes no representations, express or implied, with respect to this documentation or the software it describes, including (with no limitation) any implied warranties of utility or fitness for any particular purpose; all such warranties are expressly disclaimed. Neither LynuxWorks, Inc., nor its distributors, nor its dealers shall be liable for any indirect, incidental, or consequential damages under any circumstances.

(The exclusion of implied warranties may not apply in all cases under some statutes, and thus the above exclusion may not apply. This warranty provides the purchaser with specific legal rights. There may be other purchaser rights which vary from state to state within the United States of America.)

---

<b>PREFACE</b>	.....	<b>v</b>
	For More Information .....	v
	Typographical Conventions .....	vi
	Special Notes .....	vii
	Technical Support .....	vii
<b>CHAPTER 1</b>	<b>OVERVIEW .....</b>	<b>1</b>
<b>CHAPTER 2</b>	<b>DOWNLOADING AND BOOTING BLUECAT LINUX ON THE TARGET .....</b>	<b>3</b>
	Prerequisites .....	3
	Download and Boot Procedure Overview .....	4
	Setting up Hardware .....	4
	Connecting Target Board Serial Ports to Host .....	4
	Connecting Target Board Ethernet Card to Host .....	5
	Setting up the Firmware .....	5
	Downloading a BlueCat Linux System to Flash .....	8
	Booting a Demo System from Flash .....	9
	Booting a Demo over a Network using OS Loader .....	11
	Booting a Demo over a Network using CygMon Firmware .....	12
<b>CHAPTER 3</b>	<b>KERNEL CONFIGURATION OPTIONS .....</b>	<b>15</b>
<b>CHAPTER 4</b>	<b>SUPPORTED DEMO SYSTEMS.....</b>	<b>35</b>
	Demo Systems .....	35
	developer Demo System .....	36
	ixasdk Demo System .....	36
	osloader Demo System .....	40
	showcase Demo System .....	40
	zebra Demo System .....	40
	Using Selected RPM Packages .....	45
	Using BusyBox .....	45
	Creating a BlueCat Linux System for BusyBox .....	45
	Using TinyLogin RPM Package .....	50
<b>CHAPTER 5</b>	<b>SUPPORTED DEVICE DRIVERS .....</b>	<b>55</b>
	Flash Support .....	55

---

	Configuring Custom Flash Chips .....	56
<b>CHAPTER 6</b>	<b>BLUECAT LINUX SPECIFIC APIS .....</b>	<b>59</b>
	LED Display Control .....	59
<b>CHAPTER 7</b>	<b>DEFECT FIXES AND KNOWN LIMITATIONS.....</b>	<b>61</b>
	IXDP1200 Limitations and Workarounds .....	61

---

# — *Preface*

---

## For More Information

For more information on the features of BlueCat Linux, refer to the following printed and online documentation.

- *BlueCat Linux Release Notes*

This printed document contains late-breaking information about the current release of BlueCat.

- *BlueCat Linux User's Guide*

This document contains information about installing, configuring and using BlueCat Linux.

- Online information

The complete BlueCat Linux documentation set is available on the BlueCat Documentation CD-ROM. Books are provided in both HTML and PDF formats.

Updates to these documents are available online at the LynuxWorks Website: <http://www.lynuxworks.com>.

Additional information about commands and utilities is provided online with the **man** command. For example, to find information about the GNU GCC compiler, use the following syntax:

```
man gcc
```

## Typographical Conventions

The typefaces used in this manual, summarized below, emphasize important concepts. All references to file names and commands are case sensitive and should be typed accurately.

Kind of Text	Examples
Body text; <i>italicized</i> for emphasis, new terms, and book titles	Refer to the <i>BlueCat Linux User's Guide</i> .
Environment variables, file names, functions, methods, options, parameter names, path names, commands, and computer data	<code>ls</code> <code>-l</code> <code>myprog.c</code> <code>/dev/null</code>
Commands that need to be highlighted within body text, or commands that must be typed as is by the user are <b>bolded</b> .	<code>login: <b>myname</b></code> <code># <b>cd /usr/home</b></code>
Text that represents a variable, such as a file name or a value that must be entered by the user	<code>cat &lt;filename&gt;</code> <code>mv &lt;file1&gt; &lt;file2&gt;</code>
Blocks of text that appear on the display screen after entering instructions or commands	<code>Linux version 2.4.10-1 (bin@build1) (gcc version 2.95.3 20010315 (release)) #5</code> <code>Tue Dec 18 13:33:08 MSK 2001</code> <code>Processor: Intel StrongARM-IXP1200</code> <code>revision 3</code> <code>Architecture: Intel IXP1200</code> <code>On node 0 totalpages: 32768</code> <code>zone(0): 32768 pages.</code> <code>zone(1): 0 pages.</code> <code>zone(2): 0 pages.</code>
Keyboard options, button names, and menu sequences	<b>Enter</b> , <b>Ctrl-C</b>

## Special Notes

The following notations highlight any key points and cautionary notes that may appear in this manual.

---

**NOTE:** These callouts note important or useful points in the text.

---



**CAUTION!** Used for situations that present minor hazards that may interfere with or threaten equipment/performance.

---

## Technical Support

LynuxWorks Technical Support is available Monday through Friday (holidays excluded) between 8:00 AM and 5:00 PM Pacific Time (U.S. Headquarters) or between 9:00 AM and 6:00 PM Central European Time (Europe).

### LynuxWorks U.S. Headquarters

email [support@lnxw.com](mailto:support@lnxw.com)

phone (408) 979-3940  
800-327-5969

fax (408) 979-3945

### LynuxWorks Europe

email [tech\\_europe@lnxw.com](mailto:tech_europe@lnxw.com)

phone (+33) 1 30 85 06 00

fax (+33) 1 30 85 06 06

### World Wide Web

website <http://www.lynuxworks.com>

customer support <http://www.lynuxworks.com/support/custhelp.php3>



The *BlueCat Linux Board Support Guide for Intel IXDP1200 Advanced Development Platform* provides information about the big-endian BlueCat Linux Board Support Package (BSP) for the Intel IXDP1200 Advanced Development Platform, which includes the IXDP1200 Network Processor based on the Intel SA-1 Core (ARM architecture compliant).

Throughout this Board Support Guide (BSG), the BSP is referred to as the “*ixp1200*” and the board as the “*IXDP1200*.”

- *Chapter 1* is an overview of this BSG.
- *Chapter 2* describes the BlueCat Linux download and boot procedures for the IXDP1200 board.
- *Chapter 3* provides details of the configuration of the prebuilt BlueCat Linux kernel contained in the *ixp1200* BSP.
- *Chapter 4* lists the BlueCat Linux demo systems supported by the *ixp1200* BSP.
- *Chapter 5* lists the device drivers in the *ixp1200* BSP.
- *Chapter 6* describes BlueCat Linux-specific Application Programming Interfaces (APIs) for the IXDP1200 board.



---

# *Downloading and Booting BlueCat Linux on the Target*

This chapter provides instructions for downloading a BlueCat Linux demo system from a cross development host onto the target board, and then booting the demo system on the target.

---

## Prerequisites

This document provides instructions for downloading and booting BlueCat Linux systems on the target board. Example scenarios of using BlueCat Linux demo systems are also presented. A basic familiarity with the target board hardware and operation is required. The user must also have an understanding of system administration for the particular cross development host on which BlueCat Linux and the Board Support Package (BSP) are installed. It is assumed that the user has manufacturer documentation for the target board as well as system administration material for the development host.

Before downloading and booting BlueCat Linux on the target board, it is assumed that the default BlueCat Linux IXDP1200 configuration and the `ixp1200` BSP are installed on the cross development host. This means that the user must:

1. Install the BlueCat Linux ARM, big-endian Core onto the cross development host, as described in the “Installing the Default Configuration” section in the “Installation” Chapter of the *BlueCat Linux User’s Guide*.
2. Install the `ixp1200` BSP onto the cross development host as detailed in the “Installing Support for Target Boards” section of Chapter 1, “Installation” in the *BlueCat Linux User’s Guide*.
3. Activate support for the `ixp1200` BSP as detailed in the “Activating Support for a Target Board” section of Chapter 1, “Installation” in the *BlueCat Linux User’s Guide*.

## Download and Boot Procedure Overview

The procedures for downloading and booting a BlueCat Linux system on an IXDP1200 board consists of the following main steps:

- Setting up hardware
  - Downloading and booting a BlueCat Linux system from target flash memory
- OR
- Downloading and booting a BlueCat Linux system over a network using the BlueCat Linux OS loader or the CygMon firmware

Downloading and booting a BlueCat Linux system can be performed using the BlueCat OS loader demo system. The OS loader demo system includes the `i_osloader` and the `osloader` downloadable images. The `osloader` image is configured with the base functionality of the BlueCat Linux OS loader, including the ability to download BlueCat Linux images from a TFTP host, and execute them in RAM. `i_osloader` is extended with support for the Journalling Flash File System (JFFS) and can thus be used to download a BlueCat Linux system into target flash memory.

Please refer to Chapter 3, “Downloading and Booting BlueCat Linux” in the *BlueCat Linux User’s Guide* for a discussion of the OS loader.

---

## Setting up Hardware

### Connecting Target Board Serial Ports to Host

The target board has one serial port (the J28 DB9-connector). This port is used by the BootMgr firmware, the CygMon firmware, and the BlueCat Linux embedded system.

This port needs to be connected, preferably, to the COM1 port on the development host. Use the serial cable included with the IXDP1200 board. The serial port connected to the target has a baud rate of 38400.

## Connecting Target Board Ethernet Card to Host

The Ethernet port on the target board is used to provide a standard network connection for the board and, in particular, to load BlueCat Linux embedded systems onto the board from a network.

The Ethernet port (the J22 RJ45-connector) on the IXDP1200 must be used to connect the board to a LAN.

Networking must be set up on the host system. In particular, the user must choose a unique IP address for the host as well as the target board. These addresses are referred to as `<host_ip>` and `<target_board_ip>`, respectively. For more information on how to set up networking on the host, please refer to the host operating system documentation.

The TFTP service must be enabled on the host. Refer to “Setting Up a TFTP Server” in Chapter 3 of the *BlueCat Linux User’s Guide* for more information.

## Setting up the Firmware

Use the following procedure to set up the firmware options for BlueCat Linux operations:

---

**NOTE:** The instructions outlined in the rest of this chapter are for the IXDP1200 board with Cygmon version 2.05.

---

1. Reset the target board.
2. Press the **Space Bar** to stop autoboot.
3. At the BootMgr console, enter:

```
[BootMgr]: C
BootMgr Version 2.0.83
CPU Revision 6901C123
OS list:
  0 Flash Utility
  1 Diagnostics
  2 VxWorks
  3 Cygmon
  4 Vmon
IO Type list:
  0 Serial
  1 HPC
Default OS: 3
Countdown value: 5
Disable initial display: 0
Default IO type: 0
SDRAM Window Size: 4K
```

```
SDRAM Window Offset: 06FFF000
Upstream Window Size: 1M
Enter blank line to leave value unchanged
Default OS: 3
Countdown value: 5
Disable initial display: 0
Default IO type: 0
SDRAM Window Size: 4K
SDRAM Window Offset: 06FFF000
Upstream Window Size: 1M

[BootMgr]: b
Intel 82559 found, MAC address: 00:90:D7:00:08:D2, IO
address=00000000, IRQ=12
IXP1200 is running in big endian mode.

Cygmon, the Cygnus ROM monitor.
Cygmon was written by Robert Richardson and Bob Manson of Cygnus
Solutions
Copyright(c) 1997, 1998, 1999 Cygnus Solutions

Version: release 2.05
This image was built on Wed Aug 22 02:57:39 PDT 2001

CPU: IXP1200 StrongARM(R) Microprocessor rev. C0
Board: Intel(R) IXM1200 Network Processor Base Card
StrongARM is a Registered Trademark of Advanced RISC Machines
Limited.
Intel is a Registered Trademark of Intel Corporation
Other Brands and Trademarks are the property of their respective
owners.

cygmon>
```

#### 4. At the CygMon console, enter:

```
cygmon> bootoptions
Current IXP1200 IP Boot Options in flash ROM is invalid.
You need go through this option once to setup valid option.
Type "help bootoptions" to get more information.
[BootOptions]:
 1 BOOTP/Manual (manual)
 2 Local IP (<target_board_ip>)
 3 Remote IP (<host_ip>)
 4 Linux kernel file name (zImage)
 5 Linux ramdisk file name (ramdisk_img.gz)
 6 Linux kernel load offset (C4008000)
 7 Linux boot command line string ()
 8 Countdown to auto-boot linux (1)
Enter option number (1 to 8), "q" to quit, "s" to save: 1
Enter "b" for BOOTP, "m" for manual: m
[BootOptions]:
 1 BOOTP/Manual (manual)
 2 Local IP (<target_board_ip>)
 3 Remote IP (<host_ip>)
 4 Linux kernel file name (zImage)
 5 Linux ramdisk file name (ramdisk_img.gz)
 6 Linux kernel load offset (C4008000)
 7 Linux boot command line string ()
 8 Countdown to auto-boot linux (1)
```

```

Enter option number (1 to 8), "q" to quit, "s" to save: 2
Local IP (in dotted decimal notation): 0.0.0.0
[BootOptions]:
  1 BOOTP/Manual (manual)
  2 Local IP (0.0.0.0)
  3 Remote IP (<host_ip>)
  4 Linux kernel file name (zImage)
  5 Linux ramdisk file name (ramdisk_img.gz)
  6 Linux kernel load offset (C4008000)
  7 Linux boot command line string ()
  8 Countdown to auto-boot linux (1)
Enter option number (1 to 8), "q" to quit, "s" to save: 3
Remote IP (in dotted decimal notation): 0.0.0.0
[BootOptions]:
  1 BOOTP/Manual (manual)
  2 Local IP (0.0.0.0)
  3 Remote IP (0.0.0.0)
  4 Linux kernel file name (zImage)
  5 Linux ramdisk file name (ramdisk_img.gz)
  6 Linux kernel load offset (C4008000)
  7 Linux boot command line string ()
  8 Countdown to auto-boot linux (1)
Enter option number (1 to 8), "q" to quit, "s" to save: 6
Linux kernel load offset (must >= C0200000): C0800000
[BootOptions]:
  1 BOOTP/Manual (manual)
  2 Local IP (0.0.0.0)
  3 Remote IP (0.0.0.0)
  4 Linux kernel file name (zImage)
  5 Linux ramdisk file name (ramdisk_img.gz)
  6 Linux kernel load offset (C0800000)
  7 Linux boot command line string ()
  8 Countdown to auto-boot linux (1)
Enter option number (1 to 8), "q" to quit, "s" to save: 4
Kernel file name: ENTER
[BootOptions]:
  1 BOOTP/Manual (manual)
  2 Local IP (0.0.0.0)
  3 Remote IP (0.0.0.0)
  4 Linux kernel file name ()
  5 Linux ramdisk file name (ramdisk_img.gz)
  6 Linux kernel load offset (C0800000)
  7 Linux boot command line string ()
  8 Countdown to auto-boot linux (1)
Enter option number (1 to 8), "q" to quit, "s" to save: 5
Ramdisk file name: ENTER
[BootOptions]:
  1 BOOTP/Manual (manual)
  2 Local IP (0.0.0.0)
  3 Remote IP (0.0.0.0)
  4 Linux kernel file name ()
  5 Linux ramdisk file name ()
  6 Linux kernel load offset (C0800000)
  7 Linux boot command line string ()
  8 Countdown to auto-boot linux (1)
Enter option number (1 to 8), "q" to quit, "s" to save: s
Please wait for IXPL200 saving your option into flash ROM.
Your option has been saved into flash ROM.
cygmon>

```

## 5. Reset the target board.

As the result, the CygMon firmware is prepared to boot and download a BlueCat Linux system.

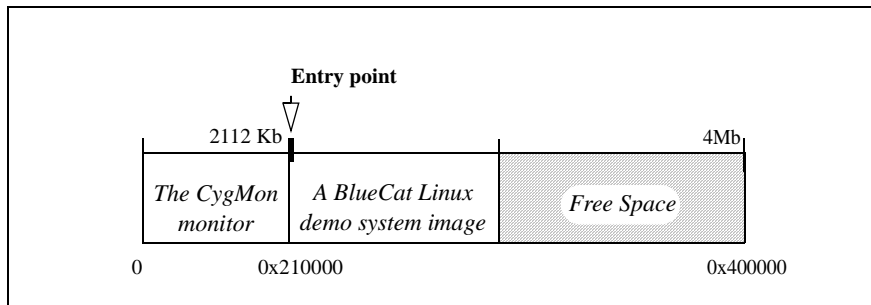
**NOTE:** Both the local IP address and remote IP address are set to 0.0.0.0 to prevent the CygMon firmware from downloading and running kernel images from a network automatically.

## Downloading a BlueCat Linux System to Flash

This section provides instructions on how to download a BlueCat Linux embedded system into target flash memory using the BlueCat Linux OS loader. Refer to the *BlueCat Linux User's Guide* for details on the OS loader.

These instructions are applicable to any demo system provided in the BlueCat Linux distribution. This chapter uses the `osloader` demo system as an example.

The following figure shows how the flash memory on the IXDP1200 board is partitioned for BlueCat Linux.



**Figure 2-1: Partitioning Flash Memory**

Use the following procedure to download `osloader` onto the target board:

1. Copy the `osloader.kdi` and `i_osloader.kdi` files from the `$BLUECAT_PREFIX/demo/osloader` directory to the `/tftpboot` directory on the host.
2. Reset the target board.

3. At the CygMon console, enter:

```
cygmon> tftp <host_ip> <target_board_ip> \  
/tftpboot/i_osloader.kdi C0800000  
  
TFTPing /path_to/i_osloader.kdi...  
file transfer complete - total 2069504 bytes  
  
cygmon>
```

where *<target\_board\_ip>* is the IP address of the target, *<host\_ip>* is the IP address of the development host.

4. At the CygMon console, enter:

```
cygmon> go C0800000  
  
cygmon_handle_exception() monitor_loop return = 0
```

These commands start the BlueCat Linux OS loader.

5. At the BlueCat Linux OS loader prompt (>), type the following commands:

```
> set IF eth0  
> set IP <target_board_ip>  
> set HOST <host_ip>  
> set FILE tftp osloader.kdi  
> exec flash_fdisk /dev/mtdchar0 40-70  
> flash /dev/mtdchar1 erase  
> reset
```

where *<target\_board\_ip>* is the IP address of the target, and *<host\_ip>* is the IP address of the development host.

After these commands have been performed, the `osloader` demo system is programmed into target flash memory, and can be booted as described in the following section.

---

## Booting a Demo System from Flash

The following procedure is used to boot the `osloader` demo system installed into target flash memory. For details on how to install a demo system into flash memory, refer to the previous section.

1. Reset the target board.
2. At the CygMon console, enter the following command to download a BlueCat Linux system into flash memory:

```
cygmon> go 210000
```

This command starts the `osloader` demo system programmed into flash. The following output appears:

```
cygmon_handle_exception() monitor_loop return = 0
Uncompressing Linux..... done,
booting the kernel.
Linux version 2.4.10 (bin@build1) (gcc version 2.95.3 20010315
(release)) #1 Tue Dec 18 13:07:44 MSK 2001
Processor: Intel StrongARM-IXP1200 revision 3
Architecture: Intel IXP1200
On node 0 totalpages: 32768
zone(0): 32768 pages.
zone(1): 0 pages.
zone(2): 0 pages.
Kernel command line: root=101
Calibrating delay loop... 133.12 BogoMIPS
Memory: 128MB = 128MB total
Memory: 127680KB available (933K code, 192K data, 48K init)
Dentry-cache hash table entries: 16384 (order: 5, 131072 bytes)
Inode-cache hash table entries: 8192 (order: 4, 65536 bytes)
Mount-cache hash table entries: 2048 (order: 2, 16384 bytes)
Buffer-cache hash table entries: 8192 (order: 3, 32768 bytes)
Page-cache hash table entries: 32768 (order: 5, 131072 bytes)
POSIX conformance testing by UNIFIX
PCI: bus0: Fast back to back transfers enabled
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Starting kswapd v1.8
IXP1200 serial driver version 1.2
IXP1200 LED Display Driver loaded & registered
IXP1200 Real Time Clock Driver version 1.09 installed.
block: queued sectors max/low 84800kB/28266kB, 256 slots per queue
RAMDISK driver initialized: 16 RAM disks of 28472K size 1024
blocksize
eepro100.c:v1.09j-t 9/29/99 Donald Becker
http://cesdis.gsfc.nasa.gov/linux/drivers/eepro100.html
eepro100.c: $Revision: 1.36 $ 2000/11/17 Modified by Andrey V.
Savochkin <saw@saw.sw.com.sg> and others
eth0: OEM i82557/i82558 10/100 Ethernet, 00:90:D7:00:08:D2, I/O at
0x8400, IRQ 11.
Board assembly ffffff-255, Physical connectors present: RJ45
Primary interface chip i82555 PHY #1.
Secondary interface chip i82555.
General self-test: passed.
Serial sub-system self-test: passed.
Internal registers self-test: passed.
ROM checksum self-test: passed (0x1d68d8db).
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP
IP: routing cache hash table of 1024 buckets, 8Kbytes
TCP: Hash tables configured (established 8192 bind 8192)
NetWinder Floating Point Emulator V0.95 (c) 1998-1999 Rebel.com
RAMDISK: Compressed image found at block 2648
RAMDISK: ext2 filesystem found at block 0
```

```
RAMDISK: Loading 620 blocks [1 disk] into ram disk... done.  
VFS: Mounted root (ext2 filesystem).  
Freeing init memory: 48K  
BlueCat Loader Shell  
>
```

---

## Booting a Demo over a Network using OS Loader

The following demonstrates booting the `showcase` demo system over a network using the BlueCat Linux OS loader.

1. Copy the `showcase.kernel` and `showcase.rfs` files from the `$BLUECAT_PREFIX/demo/showcase` directory to the `/tftpboot` directory on the cross development host.
2. Reset the target board.
3. At the CygMon console, enter the following commands:

```
cygmon> go 210000
```

These commands start the `osloader` demo system from flash. As a result, the BlueCat OS loader prompt (`>`) appears in the BlueCat Linux console.

4. At the BlueCat OS loader prompt (`>`), enter the following commands:

```
> set IF eth0  
> set IP <target_board_ip>  
> set HOST <host_ip>  
> set KERNEL tftp showcase.kernel  
> set RFS tftp showcase.rfs  
> set CMD ramdisk_size=16384  
> boot
```

where `<target_board_ip>` is the IP address of the target, and `<host_ip>` is the IP address of the development host.

These commands load the `showcase` demo system from a network onto the target board and automatically start it.

## Booting a Demo over a Network using CygMon Firmware

The IXDP1200 board can be configured to download and start a demo system from a network automatically at board power-up. Use the following procedure to prepare the board to download and boot BlueCat Linux from a network automatically:

1. Copy the `showcase.kdi` file from the `$BLUECAT_PREFIX/demo/showcase` directory to the `/tftpboot` directory on the host.
2. Reset the target board.
3. At the CygMon console, enter:

```
cygmon> bootoptions
Current IXDP1200 IP Boot Options in flash ROM is invalid.
You need go through this option once to setup valid option.
Type "help bootoptions" to get more information.
[BootOptions]:
 1 BOOTP/Manual (manual)
 2 Local IP (0.0.0.0)
 3 Remote IP (0.0.0.0)
 4 Linux kernel file name (zImage)
 5 Linux ramdisk file name (ramdisk_img.gz)
 6 Linux kernel load offset (C4008000)
 7 Linux boot command line string ()
 8 Countdown to auto-boot linux (1)
Enter option number (1 to 8), "q" to quit, "s" to save: 1
Enter "b" for BOOTP, "m" for manual: m
[BootOptions]:
 1 BOOTP/Manual (manual)
 2 Local IP (0.0.0.0)
 3 Remote IP (0.0.0.0)
 4 Linux kernel file name (zImage)
 5 Linux ramdisk file name (ramdisk_img.gz)
 6 Linux kernel load offset (C4008000)
 7 Linux boot command line string ()
 8 Countdown to auto-boot linux (1)
Enter option number (1 to 8), "q" to quit, "s" to save: 2
Local IP (in dotted decimal notation): 172.17.3.9
[BootOptions]:
 1 BOOTP/Manual (manual)
 2 Local IP (172.17.3.9)
 3 Remote IP (0.0.0.0)
 4 Linux kernel file name (zImage)
 5 Linux ramdisk file name (ramdisk_img.gz)
 6 Linux kernel load offset (C4008000)
 7 Linux boot command line string ()
 8 Countdown to auto-boot linux (1)
Enter option number (1 to 8), "q" to quit, "s" to save: 3
Remote IP (in dotted decimal notation): 172.17.0.1
[BootOptions]:
 1 BOOTP/Manual (manual)
 2 Local IP (172.17.3.9)
 3 Remote IP (172.17.0.1)
```

```

4 Linux kernel file name (zImage)
5 Linux ramdisk file name (ramdisk_img.gz)
6 Linux kernel load offset (C4008000)
7 Linux boot command line string ()
8 Countdown to auto-boot linux (1)
Enter option number (1 to 8), "q" to quit, "s" to save: 6
Linux kernel load offset (must >= C0200000): C0800000
[BootOptions]:
1 BOOTP/Manual (manual)
2 Local IP (172.17.3.9)
3 Remote IP (172.17.0.1)
4 Linux kernel file name (zImage)
5 Linux ramdisk file name (ramdisk_img.gz)
6 Linux kernel load offset (C0800000)
7 Linux boot command line string ()
8 Countdown to auto-boot linux (1)
Enter option number (1 to 8), "q" to quit, "s" to save: 4
Kernel file name: showcase.kdi
[BootOptions]:
1 BOOTP/Manual (manual)
2 Local IP (172.17.3.9)
3 Remote IP (172.17.0.1)
4 Linux kernel file name (showcase.kdi)
5 Linux ramdisk file name (ramdisk_img.gz)
6 Linux kernel load offset (C0800000)
7 Linux boot command line string ()
8 Countdown to auto-boot linux (1)
Enter option number (1 to 8), "q" to quit, "s" to save: s
Please wait for IXP1200 saving your option into flash ROM.
Your option has been saved into flash ROM.
cygmon>

```

#### 4. Reset the target board.

As a result, the CygMon firmware is programmed to download and boot the showcase demo system via a network automatically upon board power-up.



The ixp1200 BSP comes with a default BlueCat Linux kernel. This kernel has a number of configuration parameters. These parameters and their options are shown in the tables that follow.

**Table 3-1: Kernel Configuration Parameters for the ixp1200 BSP**

Parameters	Table Number
Code Maturity Level Options	Table 3-2
Loadable Module Support	Table 3-3
System Type	Table 3-4
IXDP1200 Implementations	Table 3-5
General Setup	Table 3-6
Parallel Port Support	Table 3-7
Memory Technology Devices	Table 3-8
RAM/ROM/Flash Chip Drivers	Table 3-9
Mapping Drivers for Chip Access	Table 3-10
Self-Contained MTD Device Drivers	Table 3-11
Disk-On-Chip Device Drivers	Table 3-12
NAND Flash Device Drivers	Table 3-13
Plug and Play Configuration	Table 3-14
Block Devices	Table 3-15
Multi-Device Support (RAID and LVM)	Table 3-16
Networking Options	Table 3-17
QoS and/or Fair Queueing	Table 3-18

**Table 3-1: Kernel Configuration Parameters for the ixp1200 BSP (Continued)**

Parameters	Table Number
Network Device Support	Table 3-19
ARCnet Devices	Table 3-20
Ethernet (10 or 100 Mbit)	Table 3-21
Wireless LAN (non-ham radio)	Table 3-22
Token Ring Devices	Table 3-23
Wan Interfaces	Table 3-24
Amateur Radio Support	Table 3-25
IrDA (infrared) Support	Table 3-26
ATA/IDE/MFM/RLL Support	Table 3-27
SCSI Support	Table 3-28
IEEE 1394 (FireWire) Support	Table 3-29
I2O Support	Table 3-30
ISDN Subsystem	Table 3-31
Input Core Support	Table 3-32
Character Devices	Table 3-33
Serial Drivers	Table 3-34
I2C Support	Table 3-35
L3 Serial Bus Support	Table 3-36
Mice	Table 3-37
Joysticks	Table 3-38
Watchdog Cards	Table 3-39
Floppy Tape Device Driver, Ftape	Table 3-40
Multimedia Devices	Table 3-41
File Systems	Table 3-42
Network File Systems	Table 3-43
Partition Types	Table 3-44
Sound	Table 3-45

**Table 3-1: Kernel Configuration Parameters for the ixp1200 BSP (Continued)**

Parameters	Table Number
Multimedia Capabilities Port Drivers	Table 3-46
USB Support	Table 3-47
Bluetooth Support	Table 3-48
Kernel Hacking	Table 3-49
Modular Advanced Power Management	Table 3-50
Messenger Support	Table 3-51

**Table 3-2: Code Maturity Level Options**

Option	Value	Description
CONFIG_EXPERIMENTAL	Y	Prompt for development and/or incomplete code/drivers
CONFIG_OBSOLETE	N	Prompt for obsolete code/drivers

**Table 3-3: Loadable Module Support**

Option	Value	Description
CONFIG_MODULES	Y	Enable loadable module support
CONFIG_MODVERSIONS	Y	Set version information on all symbols for modules
CONFIG_KMOD	Y	Kernel module loader

**Table 3-4: System Type**

Options	Value	Description
CONFIG_ARCH_IXP1200	Y	Intel(R) IXDP1200 Network Processor

Table 3-5: IXDP1200 Implementations

Option	Value	Description
CONFIG_SPECTACLE_ISLAND	Y	Build kernel for Spectacle Island
CONFIG_HOST_IXP1200	Y	Build kernel for IXDP1200 boards in host mode

Table 3-6: General Setup

Option	Value	Description
CONFIG_ANGELBOOT	N	Load kernel using Angel Debug Monitor
CONFIG_BLUECAT_IGNORE_PRINTK	N	BlueCat Linux Ignore <code>printk</code>
CONFIG_BLUECAT_THUMB	N	BlueCat Linux kernel support for THUMB binaries
CONFIG_BLUECAT_LOADER	N	BlueCat Linux OS loader
CONFIG_BLUECAT_SMALL_FOOTPRINT	N	BlueCat Linux small memory footprint
CONFIG_PCI_NAMES	N	PCI device name database
CONFIG_HOTPLUG	N	Support hot-pluggable devices
CONFIG_NET	Y	Networking support
CONFIG_BLUECAT_MEMSIZE	N	Memory sizing benchmarks
CONFIG_SYSVIPC	N	System V IPC
CONFIG_BSD_PROCESS_ACCT	N	BSD Process Accounting
CONFIG_SYSCTL	N	Sysctl support
CONFIG_FPE_NWFPE	Y	NWFPE math emulation
CONFIG_FPE_FASTFPE	N	FastFPE math emulation (Experimental)
CONFIG_KCORE_ELF	Y	Kernel core ( <code>/proc/kcore</code> ) format
CONFIG_BINFMT_AOUT	N	Kernel support for <code>a.out</code> binaries
CONFIG_BINFMT_ELF	Y	Kernel support for ELF binaries
CONFIG_BINFMT_MISC	N	Kernel support for MISC binaries
CONFIG_PM	N	Power management support (Experimental)

**Table 3-6: General Setup (Continued)**

Option	Value	Description
CONFIG_ARTHUR	N	RISC OS personality
CONFIG_LEDS	N	Timer and CPU usage LEDS
CONFIG_ALIGNMENT_TRAP	Y	Kernel-mode alignment trap handler

**Table 3-7: Parallel Port Support**

Option	Value	Description
CONFIG_PARPORT	N	Parallel port support

**Table 3-8: Memory Technology Devices**

Option	Value	Description
CONFIG_MTD	Y	Memory Technology Device (MTD) support
CONFIG_MTD_DEBUG	N	Debugging
CONFIG_MTD_PARTITIONS	Y	MTD partitioning support
CONFIG_MTD_REDBOOT_PARTS	N	RedBoot partition table parsing
CONFIG_MTD_BOOTLDR_PARTS	N	Compaq bootldr partition table parsing
CONFIG_MTD_AFS_PARTS	N	ARM Firmware Suite partition parsing
CONFIG_MTD_CHAR	Y	Direct character device access to MTD devices
CONFIG_MTD_BLOCK	Y	Caching block device access to MTD devices
CONFIG_FTL	N	FTL (Flash Translation Layer) support
CONFIG_NFTL	N	NFTL (NAND Flash Translation Layer) support

**Table 3-9: RAM/ROM/Flash Chip Drivers**

Option	Value	Description
CONFIG_MTD_CFI	N	Detect flash chips by Common Flash Interface (CFI) probe
CONFIG_MTD_JEDEC	Y	Detect non-CFI AMD/JEDEC-compatible flash chips
CONFIG_MTD_CFI_ADV_OPTIONS	Y	CFI advanced configuration options
CONFIG_MTD_CFI_LE_BYTE_SWAP	Y	Flash command/query data swapping
CONFIG_MTD_CFI_GEOMETRY	N	Specific CFI Flash geometry selection
CONFIG_MTD_CFI_INTELEXT	N	Support for Intel/Sharp flash chips
CONFIG_MTD_CFI_AMDSTD	N	Support for AMD/Fujitsu flash chips
CONFIG_MTD_RAM	N	Support for RAM chips in bus mapping
CONFIG_MTD_ROM	N	Support for ROM chips in bus mapping
CONFIG_MTD_ABSENT	N	Support for absent chips in bus mapping
CONFIG_MTD_OBSOLETE_CHIPS	N	Older (theoretically obsoleted now) drivers for non-CFI chips

**Table 3-10: Mapping Drivers for Chip Access**

Option	Value	Description
CONFIG_MTD_IXP1200	Y	Flash chip mapping on the Intel IXP1200 board
CONFIG_MTD_IXP1200_PART	:	Partitions layout
CONFIG_MTD_PCI	N	PCI MTD driver

**Table 3-11: Self-Contained MTD Device Drivers**

Option	Value	Description
CONFIG_MTD_PMC551	N	Ramix PMC551 PCI Mezzanine RAM card support
CONFIG_MTD_SDRAM	N	Uncached system RAM
CONFIG_MTD_MTDDRAM	N	Test driver using RAM
CONFIG_MTD_BLKMTD	N	TD emulation using block device

**Table 3-12: Disk-On-Chip Device Drivers**

Option	Value	Description
CONFIG_MTD_DOC1000	N	M-Systems Disk-On-Chip 1000
CONFIG_MTD_DOC2000	N	M-Systems Disk-On-Chip 2000 and Millennium
CONFIG_MTD_DOC2001	N	M-Systems Disk-On-Chip Millennium-only alternative driver

**Table 3-13: NAND Flash Device Drivers**

Option	Value	Description
CONFIG_MTD_NAND	N	NAND device support

**Table 3-14: Plug and Play Configuration**

Option	Value	Description
CONFIG_PNP	N	Plug and Play support

**Table 3-15: Block Devices**

Option	Value	Description
CONFIG_BLK_DEV_FD	N	Normal PC floppy disk support
CONFIG_BLK_CPQ_DA	N	Compaq SMART2 support
CONFIG_BLK_CPQ_CISS_DA	N	Compaq Smart Array 5xxx support
CONFIG_BLK_DEV_DAC960	N	Mylex DAC960/DAC1100 PCI RAID Controller support
CONFIG_BLK_DEV_LOOP	N	Loopback device support
CONFIG_BLK_DEV_NBD	N	Network block device support
CONFIG_BLK_DEV_MD	N	Multiple devices driver support
CONFIG_BLK_DEV_RAM	Y	RAM disk support
CONFIG_BLK_DEV_RAM_SIZE	28472	Default RAM disk size
CONFIG_BLK_DEV_INITRD	N	Initial RAM disk ( <code>initrd</code> ) support
CONFIG_BLUECAT_RFS	Y	BlueCat Linux RFS support

**Table 3-16: Multi-Device Support (RAID and LVM)**

Option	Value	Description
CONFIG_MD	N	Multiple device support (RAID and LVM)

**Table 3-17: Networking Options**

Option	Value	Description
CONFIG_PACKET	N	Packet socket
CONFIG_NETLINK	N	Kernel/User netlink socket
CONFIG_NETFILTER	N	Network packet filtering (Replaces <code>ipchains</code> )
CONFIG_FILTER	N	Socket filtering
CONFIG_UNIX	Y	UNIX domain sockets

**Table 3-17: Networking Options (Continued)**

Option	Value	Description
CONFIG_INET	Y	TCP/IP networking
CONFIG_IP_MULTICAST	N	IP: Multicasting
CONFIG_IP_ADVANCED_ROUTER	N	IP: Advanced router
CONFIG_IP_PNP	N	IP: Kernel level autoconfiguration
CONFIG_NET_IPIP	N	IP: Tunneling
CONFIG_NET_IPGRE	N	IP: GRE tunnels over IP
CONFIG_INET_ECN	N	TCP Explicit Congestion Notification support
CONFIG_SYN_COOKIES	N	IP: TCP syncookie support (Not enabled per default)
CONFIG_IPV6	N	The IPv6 protocol (Experimental)
CONFIG_KHTTPD	N	Kernel httpd acceleration (Experimental)
CONFIG_ATM	N	Asynchronous Transfer Mode (ATM) (Experimental)
CONFIG_IPX	N	The IPX protocol
CONFIG_ATALK	N	AppleTalk protocol support
CONFIG_DECNET	N	DECnet Support
CONFIG_BRIDGE	N	802.1d Ethernet Bridging support
CONFIG_X25	N	CCITT X.25 Packet Layer (Experimental)
CONFIG_LAPB	N	LAPB Data Link Layer (Experimental)
CONFIG_LLC	N	802.2 LLC (Experimental)
CONFIG_NET_DIVERT	N	Frame Diverter (Experimental)
CONFIG_ECONET	N	Acorn Econet/AUN protocols (Experimental)
CONFIG_WAN_ROUTER	N	WAN Router
CONFIG_NET_FASTROUTE	N	Fast switching
CONFIG_NET_HW_FLOWCONTROL	N	Forwarding between two high speed interfaces

**Table 3-18: QoS and/or Fair Queueing**

Option	Value	Description
CONFIG_NET_SCHED	N	QoS and/or Fair Queueing (Experimental)

**Table 3-19: Network Device Support**

Option	Value	Description
CONFIG_NETDEVICES	Y	Network device support
CONFIG_DUMMY	N	Dummy net driver support
CONFIG_BONDING	N	Bonding driver support
CONFIG_EQUALIZER	N	EQL (serial line load balancing) support
CONFIG_TUN	N	Universal TUN/TAP driver support
CONFIG_FDDI	N	FDDI driver support
CONFIG_HIPPI	N	HIPPI driver support (Experimental)
CONFIG_PPP	N	PPP (Point-to-Point Protocol) support
CONFIG_SLIP	N	SLIP (serial line) support
CONFIG_NET_FC	N	Fibre Channel driver support
CONFIG_RCPCI	N	Red Creek Hardware VPN (Experimental)
CONFIG_SHAPER	N	Traffic Shaper (Experimental)

**Table 3-20: ARCnet Devices**

Option	Value	Description
CONFIG_ARCNET	N	ARCnet support

**Table 3-21: Ethernet (10 or 100 Mbit)**

Option	Value	Description
CONFIG_NET_ETHERNET	Y	Ethernet (10 or 100 Mbit)
CONFIG_HAPPYMEAL	N	Sun Happy Meal 10/100baseT PCI support
CONFIG_SUNGEM	N	Sun GEM support
CONFIG_NET_VENDOR_3COM	N	3COM cards
CONFIG_NET_VENDOR_SMC	N	Western Digital/SMC cards
CONFIG_NET_VENDOR_RACAL	N	Racal-Interlan (Micom) NI cards
CONFIG_HP100	N	HP 10/100VG PCLAN (ISA, EISA, PCI) support
CONFIG_NET_PCI	Y	EISA, VLB, PCI and on board controllers
CONFIG_PCNET32	N	AMD PCnet32 PCI support
CONFIG_ADAPTEC_STARFIRE	N	Adaptec Starfire support (Experimental)
CONFIG_TULIP	N	DECchip Tulip (dc21x4x) PCI support
CONFIG_DE4X5	N	Generic DECchip & DIGITAL EtherWORKS PCI/EISA
CONFIG_DGRS	N	Digi Intl. RightSwitch SE-X support
CONFIG_DM9102	N	Davicom DM910x/DM980x support
CONFIG_EEPRO100	Y	EtherExpress PRO/100 support
CONFIG_FEALNX	N	Myson MTD-8xx PCI Ethernet support
CONFIG_NATSEMI	N	National Semiconductor DP83810 series PCI Ethernet support
CONFIG_NE2K_PCI	N	PCI NE2000 and clones support
CONFIG_8139TOO	N	RealTek RTL-8139 Fast Ethernet Adapter support
CONFIG_SIS900	N	SiS 900/7016 PCI Fast Ethernet Adapter support
CONFIG_EPIC100	N	SMC EtherPower II
CONFIG_SUNDANCE	N	Sundance Alta support
CONFIG_TLAN	N	TI ThunderLAN support
CONFIG_VIA_RHINE	N	VIA Rhine support

**Table 3-21: Ethernet (10 or 100 Mbit) (Continued)**

Option	Value	Description
CONFIG_WINBOND_840	N	Winbond W89c840 Ethernet support
CONFIG_NET_POCKET	N	Pocket and portable adapters

**Table 3-22: Wireless LAN (non-ham radio)**

Option	Value	Description
CONFIG_NET_RADIO	N	Wireless LAN (non-ham radio)

**Table 3-23: Token Ring Devices**

Option	Value	Description
CONFIG_TR	N	Token Ring driver support

**Table 3-24: Wan Interfaces**

Option	Value	Description
CONFIG_WAN	N	Wan interfaces support

**Table 3-25: Amateur Radio Support**

Option	Value	Description
CONFIG_HAMRADIO	N	Amateur radio support

---

**Table 3-26: IrDA (infrared) Support**

Option	Value	Description
CONFIG_IRDA	N	IrDA subsystem support

**Table 3-27: ATA/IDE/MFM/RLL Support**

Option	Value	Description
CONFIG_IDE	N	ATA/IDE/MFM/RLL support

**Table 3-28: SCSI Support**

Option	Value	Description
CONFIG_SCSI	N	SCSI support

**Table 3-29: IEEE 1394 (FireWire) Support**

Option	Value	Description
CONFIG_IEEE1394	N	IEEE 1394 (FireWire) support (Experimental)

**Table 3-30: I2O Support**

Option	Value	Description
CONFIG_I2O	N	I2O support

**Table 3-31: ISDN Subsystem**

Option	Value	Description
CONFIG_ISDN	N	ISDN support

**Table 3-32: Input Core Support**

Option	Value	Description
CONFIG_INPUT	N	Input core support
CONFIG_INPUT_MOUSEDEV_SCREEN_X	1024	Horizontal screen resolution
CONFIG_INPUT_MOUSEDEV_SCREEN_Y	768	Vertical screen resolution

**Table 3-33: Character Devices**

Option	Value	Description
CONFIG_VT	N	Virtual terminal
CONFIG_SERIAL	N	Standard/generic (8250/16550 and compatible UARTs) serial support
CONFIG_SERIAL_NONSTANDARD	N	Non-standard serial port support
CONFIG_SERIAL_IXP1200	Y	IXDP1200 serial port support
CONFIG_SERIAL_IXP1200_CONSOLE	Y	Console on IXDP1200 serial port
CONFIG_LED_IXP1200	Y	IXDP1200 LED control
CONFIG_UNIX98_PTYS	Y	Unix98 PTY support
CONFIG_UNIX98_PTY_COUNT	32	Maximum number of Unix98 PTYs in use (0-2048)
CONFIG_QIC02_TAPE	N	QIC-02 tape support
CONFIG_INTEL_RNG	N	Intel i8x0 Random Number Generator support
CONFIG_NVRAM	N	/dev/nvram support
CONFIG_RTC	N	Enhanced Real Time Clock support
CONFIG_DTLK	N	Double Talk PC internal speech card support

**Table 3-33: Character Devices (Continued)**

Option	Value	Description
CONFIG_R3964	N	Siemens R3964 line discipline
CONFIG_APPLICOM	N	Applicom intelligent fieldbus card support
CONFIG_AGP	N	/dev/agpgart (AGP Support)
CONFIG_DRM	N	Direct Rendering Manager (XFree86 DRI support)
CONFIG_MWAVE	N	ACP Modem (Mwave) support

**Table 3-34: Serial Drivers**

Option	Value	Description
CONFIG_SERIAL_8250	N	8250/16550 and compatible serial support (Experimental)

**Table 3-35: I2C Support**

Option	Value	Description
CONFIG_I2C	N	I2C support

**Table 3-36: L3 Serial Bus Support**

Option	Value	Description
CONFIG_L3	N	L3 serial bus support

**Table 3-37: Mice**

Option	Value	Description
CONFIG_BUSMOUSE	N	Bus Mouse Support
CONFIG_MOUSE	N	Mouse Support (not serial and bus mice)

**Table 3-38: Joysticks**

Option	Value	Description
CONFIG_INPUT_GAMEPORT	N	Game port support

**Table 3-39: Watchdog Cards**

Option	Value	Description
CONFIG_WATCHDOG	N	Watchdog Timer support

**Table 3-40: Floppy Tape Device Driver, Ftape**

Option	Value	Description
CONFIG_FTAPPE	N	Ftape (QIC-80/Travan) support

**Table 3-41: Multimedia Devices**

Option	Value	Description
CONFIG_VIDEO_DEV	N	Video for Linux

**Table 3-42: File Systems**

Option	Value	Description
CONFIG_QUOTA	N	Quota support
CONFIG_AUTOFS_FS	N	Kernel automounter support
CONFIG_AUTOFS4_FS	N	Kernel automounter v4 support (also supports v3)
CONFIG_REISERFS_FS	N	Reiserfs support
CONFIG_ADFS_FS	N	ADFS file system support

**Table 3-42: File Systems (Continued)**

Option	Value	Description
CONFIG_AFFS_FS	N	Amiga FFS file system support (Experimental)
CONFIG_HFS_FS	N	Apple Macintosh file system support (Experimental)
CONFIG_BFS_FS	N	BFS file system support (Experimental)
CONFIG_CMS_FS	N	CMS file system support (Experimental)
CONFIG_EXT3_FS	N	Ext3 journaling file system support (Experimental)
CONFIG_FAT_FS	N	DOS FAT file system support
CONFIG_EFS_FS	N	EFS file system support (Read-only) (Experimental)
CONFIG_JFFS_FS	Y	Journaling Flash File System (JFFS) support (Experimental)
CONFIG_JFFS_FS_VERBOSE	0	JFFS debugging verbosity (0 = quiet, 3 = noisy)
CONFIG_JFFS_PROC_FS	N	JFFS stats available in <code>/proc</code> file system
CONFIG_JFFS2_FS	N	Journaling Flash File System v2 (JFFS2) support (Experimental)
CONFIG_CRAMFS	N	Compressed ROM file system support
CONFIG_TMPFS	N	Virtual memory file system support (former shm file system)
CONFIG_RAMFS	N	Simple RAM-based file system support
CONFIG_ISO9660_FS	N	ISO 9660 CD-ROM file system support
CONFIG_MINIX_FS	N	Minix file system support
CONFIG_FREEVXFS_FS	N	FreeVxFS file system support (VERITAS VxFS(TM) compatible)
CONFIG_NTFS_FS	N	NTFS file system support (Read-only)
CONFIG_HPFS_FS	N	OS/2 HPFS file system support
CONFIG_PROC_FS	Y	<code>/proc</code> file system support
CONFIG_DEVFS_FS	N	<code>/dev</code> file system support (Experimental)
CONFIG_DEVPTS_FS	Y	<code>/dev/pts</code> file system for Unix98 PTYs
CONFIG_QNX4FS_FS	N	QNX4 file system support (Read-only) (Experimental)

**Table 3-42: File Systems (Continued)**

Option	Value	Description
CONFIG_ROMFS_FS	N	ROM file system support
CONFIG_EXT2_FS	Y	Second extended file system support
CONFIG_SYSV_FS	N	System V/Xenix/V7/Coherent file system support
CONFIG_UDF_FS	N	UDF file system support (Read-only)
CONFIG_UFS_FS	N	UFS file system support (Read-only)

**Table 3-43: Network File Systems**

Option	Value	Description
CONFIG_CODA_FS	N	Coda file system support (advanced network file system)
CONFIG_INTERMEZZO_FS	N	InterMezzo file system support (Experimental, replicating file system)
CONFIG_NFS_FS	Y	NFS file system support
CONFIG_NFS_V3	N	Provide NFSv3 client support
CONFIG_NFSD	N	NFS server support
CONFIG_SMB_FS	N	SMB file system support (to mount Windows shares, etc.)
CONFIG_NCP_FS	N	NCP file system support (to mount NetWare volumes)

**Table 3-44: Partition Types**

Option	Value	Description
CONFIG_PARTITION_ADVANCED	N	Advanced partition selection

---

**Table 3-45: Sound**

Option	Value	Description
CONFIG_SOUND	N	Sound support

**Table 3-46: Multimedia Capabilities Port Drivers**

Option	Value	Description
CONFIG_MCP	N	Multimedia drivers

**Table 3-47: USB Support**

Option	Value	Description
CONFIG_USB	N	Support for USB
CONFIG_USB_STORAGE_SDD09	N	SanDisk SDDR-09 (and other SmartMedia) support

**Table 3-48: Bluetooth Support**

Option	Value	Description
CONFIG_BLUEZ	N	HCI EMU (virtual device) driver

**Table 3-49: Kernel Hacking**

Option	Value	Description
CONFIG_NO_FRAME_POINTER	Y	Compile kernel without frame pointer
CONFIG_DEBUG_ERRORS	N	Verbose kernel error messages
CONFIG_DEBUG_USER	N	Verbose user fault messages
CONFIG_DEBUG_INFO	N	Include debugging information in kernel binary

**Table 3-49: Kernel Hacking (Continued)**

Option	Value	Description
CONFIG_DEBUG_SLAB	N	Debug memory allocations
CONFIG_MAGIC_SYSRQ	N	Magic SysRq key
CONFIG_BLUECAT_KDBG	N	Include kdbg kernel debugger
CONFIG_DEBUG_SPINLOCK	N	Spinlock debugging
CONFIG_DEBUG_LL	N	Kernel low-level debugging functions

**Table 3-50: Modular Advanced Power Management**

Option	Value	Description
CONFIG_BLUECAT_APM	N	MAPM support

**Table 3-51: Messenger Support**

Option	Value	Description
CONFIG_BLUECAT_IOPMAN	N	Enable IOP Manager support
CONFIG_BLUECAT_MSNG	N	Enable Messenger Support

This chapter provides information about BlueCat Linux demo systems supported by the ixp1200 Board Support Package (BSP).

## Demo Systems

The following table lists the demo systems supported in the ixp1200 BSP distribution, the boot devices supported by each demo, and their respective RAM and ROM requirements.

**Table 4-1: Demo Systems Supported by ixp1200 BSP**

Demo	Boot Devices Supported by Default	ROM Requirements	RAM Requirements
developer	<b>Network</b> using OS loader, <b>Network</b> using CygMon	3395 KB	17000 KB
ixasdk	<b>Network</b> using OS loader, <b>Network</b> using CygMon	3395 KB	17000 KB
osloader	<b>Flash</b> , <b>Network</b> using OS loader, <b>Network</b> using CygMon	1392 KB	6000 KB
showcase	<b>Network</b> using OS loader, <b>Network</b> using CygMon	2546 KB	15000 KB
zebra	<b>Network</b> using OS loader, <b>Network</b> using CygMon	3380 KB	17500 KB

## developer Demo System

The `developer` demo system is a package consisting of the functionalities of the `shell`, `ftp`, `ping`, `gdb`, and `vl_demo` systems. Refer to Chapter 4 of the *BlueCat Linux User's Guide* for descriptions of `developer` and its component demo systems.

## ixasdk Demo System

The `ixasdk` demo system demonstrates the Intel IXA SDK functionality. Because the `ixasdk` demo system is specific to the IXDP1200 board, its description is not included in the *BlueCat Linux User's Guide*.

### SYNOPSIS

The Intel IXA SDK functionality demonstration

### REQUIREMENTS

Storage	Medium
RAM	Large
Network	Yes
Disk	None
Special	Two test machines (Test1 and Test2) connected using straight twisted-pair cables to the ports #1 and #8 respectively of the Intel 21440 Ethernet Controller of the IXDP1200 board. Test machines must be running Red Hat Linux 7.x.

### Kernel Option

```
ramdisk_size=28472
```

### DESCRIPTION

Intel IXA SDK is software designed to implement the framework for Internet packet filtering using IXDP1200-specific hardware such as the IXDP1200 microengines and 100 Mbit and 1 Gbit IXP1200 Ethernet ports.

The `ixasdk` demo system demonstrates the process of routing Ethernet packets between two hosts using the Intel IXA software.

The following steps are needed to configure the routing process:

1. Boot the `ixasdk` demo system on the IXDP1200 board and login as `root` (without a password).

- To start the routing process, type the following commands on the IXDP1200 console:

```
bash-2.04# cd /ixasdk/bin/arm-be
bash-2.04# ./ixstart ixsys.config-l3fwdr
```

The output similar to the following will appear:

```
Start module gigadriv
Multipacket mode is on
device id is 9f
Cannot register service: RPC: Unable to receive; errno = Connection
refused
unable to register (SA1200TARGET, SA1200VERSION, udp).ixconfig:
Initialized Resource Manager
ixconfig: ix_task_init successful
ixconfig: Created OSSL thread
ixconfig: ix_res_open_sync successful
ixconfig: ix_ns_open_sync successful
ixconfig: initialized IX API
ixconfig: Read configuration file
ixconfig: Set up interfaces
ixconfig: Set up Microcode
ixconfig: Starting micro aces
ixconfig: Starting microace ifaceInput
ixconfig: Started Microace: ifaceInput
ixconfig: Starting microace ifaceOutput
ixconfig: Started Microace: ifaceOutput
ixconfig: Starting microace L3Fwdr
ixconfig: Started Microace: L3Fwdr
ixconfig: Started MicroACES
ixconfig: Starting regular aces
stackace: Init Module loaded at c488f7e4
stackace: ix_init Module loaded at c4884060
stackace: kstack_mod.o[stackAce]: received argc=3
stackace: argv[0] is './kstack_mod.o'
stackace: argv[1] is 'stackAce'
stackace: argv[2] is 'ifaceInput'
stackace: argv[3] is NULL
ixconfig: Started other aces
ixconfig: Configuring micro aces
ixconfig: Setting properties for the different interface aces
calling portMgmt Init.
calling logicalIfMgmt Init.
ixconfig: connecting to OMS for portMgmt.
ixconfig: connecting to OMS for logicalIfMgmt.
ixconfig: setting MAC address for port 0.
ixconfig: Clearing promiscuous mode for port 0.
ixconfig: adding interface to port 0.
ixconfig: setting MAC address for port 1.
ixconfig: Clearing promiscuous mode for port 1.
ixconfig: adding interface to port 1.
ixconfig: setting MAC address for port 2.
ixconfig: Clearing promiscuous mode for port 2.
ixconfig: adding interface to port 2.
ixconfig: setting MAC address for port 3.
ixconfig: Clearing promiscuous mode for port 3.
ixconfig: adding interface to port 3.
ixconfig: setting MAC address for port 4.
ixconfig: Clearing promiscuous mode for port 4.
```

```
ixconfig: adding interface to port 4.
ixconfig: setting MAC address for port 5.
ixconfig: Clearing promiscuous mode for port 5.
ixconfig: adding interface to port 5.
ixconfig: setting MAC address for port 6.
ixconfig: Clearing promiscuous mode for port 6.
ixconfig: adding interface to port 6.
ixconfig: setting MAC address for port 7.
ixconfig: Clearing promiscuous mode for port 7.
ixconfig: adding interface to port 7.
ixconfig: setting MAC address for port 16.
ixconfig: Clearing promiscuous mode for port 16.
ixconfig: adding interface to port 16.
ixconfig: disconnecting from OMS for portMgmt.
ixconfig: disconnecting from OMS for logicalIfMgmt.
ixconfig: calling logicalIfMgmt fini.
ixconfig: calling portMgmt fini.
ixconfig: Completed IDL configuration.
ixconfig: Configured MicroACES
ixconfig: Configuring regular aces
ixconfig: Configured other ACES
Binding all the aces
ixconfig: Bound all the aces together
ixconfig: Running shell commands
stackconfig: ix_task_init successful
stackconfig: Created OSSL thread
stackcondevice eth1 entered promiscuous mode
fig: ix_res_opendeviceth1 left promiscuous mode
_sync successfuldevice eth2 entered promiscuous mode

stackcdevice eth2 left promiscuous mode
config: idevice eth3 entered promiscuous mode
x_ns_open_sync sdevice eth3 left promiscuous mode
uccessfudevice eth4 entered promiscuous mode
1
stackdevice eth4 left promiscuous mode
config: device eth5 entered promiscuous mode
initialized IX Adevice eth5 left promiscuous mode
PI
stacdevice eth6 entered promiscuous mode
kconfig: Read codevice eth6 left promiscuous mode
nfiguratdevice eth7 entered promiscuous mode
ion file
stackdevice eth7 left promiscuous mode
config: device eth8 entered promiscuous mode
setting up interdevice eth8 left promiscuous mode
face ethdevice eth17 entered promiscuous mode
1, with mac 00:00device eth17 left promiscuous mode
1:02:03:04:05
stackconfig: ip: a010001, mask: fffffff0, bcast: a0100ff
stackconfig: setting up interface eth2, with mac 00:01:02:03:04:06
stackconfig: ip: a020001, mask: fffffff0, bcast: a0200ff
stackconfig: setting up interface eth3, with mac 00:01:02:03:04:07
stackconfig: ip: a030001, mask: fffffff0, bcast: a0300ff
stackconfig: setting up interface eth4, with mac 00:01:02:03:04:08
stackconfig: ip: a040001, mask: fffffff0, bcast: a0400ff
stackconfig: setting up interface eth5, with mac 00:01:02:03:04:09
stackconfig: ip: a050001, mask: fffffff0, bcast: a0500ff
stackconfig: setting up interface eth6, with mac 00:01:02:03:04:10
stackconfig: ip: a060001, mask: fffffff0, bcast: a0600ff
stackconfig: setting up interface eth7, with mac 00:01:02:03:04:11
stackconfig: ip: a070001, mask: fffffff0, bcast: a0700ff
stackconfig: setting up interface eth8, with mac 00:01:02:03:04:12
```

```
stackconfig: ip: a080001, mask: fffffff0, bcast: a0800ff
stackconfig: setting up interface eth17, with mac 00:01:02:03:04:21
stackconfig: ip: a110001, mask: fffffff0, bcast: a1100ff
l3config: initialized IX API
l3config: Read configuration file
l3config: Configured L3 Interfaces
l3config: routeadd: a010000, fffffff0, 0, 0
l3config: routeadd: /bin/route add -net 10.1.0.0 netmask 255.255.255.0 dev
eth1
l3config: routeadd: a020000, fffffff0, 0, 1
l3config: routeadd: /bin/route add -net 10.2.0.0 netmask 255.255.255.0 dev
eth2
l3config: routeadd: a030000, fffffff0, 0, 2
l3config: routeadd: /bin/route add -net 10.3.0.0 netmask 255.255.255.0 dev
eth3
l3config: routeadd: a040000, fffffff0, 0, 3
l3config: routeadd: /bin/route add -net 10.4.0.0 netmask 255.255.255.0 dev
eth4
l3config: routeadd: a050000, fffffff0, 0, 4
l3config: routeadd: /bin/route add -net 10.5.0.0 netmask 255.255.255.0 dev
eth5
l3config: routeadd: a060000, fffffff0, 0, 5
l3config: routeadd: /bin/route add -net 10.6.0.0 netmask 255.255.255.0 dev
eth6
l3config: routeadd: a070000, fffffff0, 0, 6
l3config: routeadd: /bin/route add -net 10.7.0.0 netmask 255.255.255.0 dev
eth7
l3config: routeadd: a080000, fffffff0, 0, 7
l3config: routeadd: /bin/route add -net 10.8.0.0 netmask 255.255.255.0 dev
eth8
l3config: Configured L3 Routes
l3config: disconnecting from OMS
l3config: Finished L3 Configuration
ixconfig: Ran shell commands
ixconfig: Enabled the microengines
```

3. To configure a network interface on the first test machine (Test1), execute the following commands on Test1 from the `root`'s account:

```
bash# ifconfig eth0 down
bash# ifconfig eth0 10.1.0.10 netmask 255.255.255.0
bash# route add default gw 10.1.0.1
```

4. To configure a network interface on the second test machine (Test2), execute the following commands on Test2 from the `root`'s account:

```
bash# ifconfig eth0 down
bash# ifconfig eth0 10.8.0.10 netmask 255.255.255.0
bash# route add default gw 10.8.0.1
```

5. The routing process is established now. To check this, execute the following command on Test1:

```
bash# ping 10.8.0.10
```

or the following command on Test2:

```
bash# ping 10.1.0.10
```

## osloader Demo System

`osloader` is the BlueCat Linux OS loader used to boot a BlueCat Linux system on the target board. Refer to Chapter 4 of the *BlueCat Linux User's Guide* for details.

## showcase Demo System

The `showcase` demo system starts and configures the Apache HTTP daemon turning the target board into a Web server. Refer to Chapter 4 of the *BlueCat Linux User's Guide* for details.

## zebra Demo System

The `zebra` demo system demonstrates the Zebra routing mechanism. Because the `zebra` demo system is specific to the IXDP1200 board its description is not included in the *BlueCat Linux User's Guide*.

### SYNOPSIS

The Zebra Route Server functionality demonstration

### REQUIREMENTS

Storage	Tiny
RAM	Small
Network	Yes
Disk	None
Special	Three machines: Demo Machine (IXDP1200 target) and two test machines (Test1 and Test2) are connected by a twisted-pair cable to an Ethernet HUB. Although Test1 Machine and Test2 Machine are in the same physical network, they belong to different IP subnets. Demo Machine is running BlueCat Linux. Test1 Machine and Test2 Machine are running Red Hat Linux 7.1.

Kernel Option On all machines the `CONFIG_IP_MULTICAST` and `CONFIG_NETLINK` options must be set to `Y`.

## DESCRIPTION

GNU Zebra is a free software distributed under GNU Generic Public License that manages a TCP/IP-based routing service with routing protocol support such as BGP4, BGP4+, OSPFv2, OSPFv3, RIPv1, RIPv2, and RIPng.

Routing is a process by which a host with multiple network connections decides where to deliver the IP packets it receives.

Each host keeps a special list of routing rules called a routing table. This table contains rows with at least three fields: a destination address, the name of the interface where the packet is to be routed, and optionally, the IP address of another machine that carries the packet on its next step through the network.

The routing process is very simple. The incoming packet is received, the destination address is examined, and then it is compared with each entry in the table. The entry that best matches the address is selected, and the packet is forwarded to the specified interface. If the gateway field is filled, then the packet is forwarded to that host via the specified interface. The destination address is otherwise assumed to be on the network supported by the interface.

There are two types of routing: static (manual) routing and dynamic routing. A static routing *algorithm* is simply a definition of the routing table established by the network administrator in order to allow routing on the network. The routing table never changes without a manual update by the administrator. Dynamic routing is a technique developed to automatically adjust routing tables in the event of network failures. The most common dynamic routing protocols are RIP (Routing Information Protocol) and OSPF (Open Shortest Path First Protocol). RIP is very common on small networks. OSPF is capable at handling large network configurations, and it is suited to environments where there is a large number of possible paths through the network.

With Zebra installed, the machine exchanges routing information with other routers using routing protocols.

Traditional routing software is made as a one process program providing all routing protocol functionalities. Zebra takes a different approach. It is made up of a collection of daemons that work together to build a routing table. There is no need for these daemons to be running on the same machine. There may be several protocol-specific routing daemons and Zebra that acts as the kernel routing manager. It is easy to add a new routing protocol daemons to the entire routing system without affecting any other software by running only the protocol daemon

associated with routing protocols in use. Thus, the user may run a specific daemon and send routing reports to a central routing console.

The `zebra` demo system is intended to demonstrate advantages of Zebra's modularity in configuring a network using the RIP and OSPF routing protocols. Although only these two popular protocols are included in `zebra`, the demo system can easily be extended with support of any routing protocol from a number of protocols supported in the Zebra software. For such an extension, the user must create configuration files for desired routing protocols and then update the demo `.spec` file to include the daemons from the Zebra software and the created configuration files. For more information on creating configuration files, refer to Zebra documentation in the `$BLUECAT_PREFIX/cdt/doc/zebra_trg-0.91a` directory.

Because BlueCat Linux for IXDP1200 supports one Ethernet device, only one network interface is available on this target. When it is necessary to support multiple networks using the same Ethernet device, the Linux *aliasing* feature is involved. With Linux aliasing, a number of network interfaces, each with a unique IP address, is created on the Ethernet device. Each created network interface supports its own subnet.

In the `zebra` demo, two network interfaces are necessary on the Demo Machine to support Test1 Machine and Test2 Machine, which belong to different subnets. The following table shows network settings used to run the `zebra` demo system:

Machine Name	IP Address	Subnet Mask
Demo Machine		
eth0	172.17.3.9	255.255.0.0
eth0:0	172.21.0.1	255.255.0.0
Test1 Machine	172.17.0.5	255.255.0.0
Test2 Machine	172.21.0.2	255.255.0.0

The following procedure describes necessary steps to install, configure, run, and test the `zebra` demo system.

1. On Demo Machine, bring up the network interface for Ethernet Device 2 manually using the `ifconfig` command:

```
# ifconfig eth0:0 172.21.0.1 netmask \  
255.255.0.0
```

2. On Demo Machine, start the RIP and OSPF network protocols by running the `ripd` and `ospfd` daemons:

```
# ripd -d
# ospfd -d
```

3. Make sure that both Ethernet Device 1 and Ethernet Device 2 on Demo Machine have been started by entering the `ifconfig -a` command:

```
# ifconfig -a
eth0 Link encap:Ethernet HWaddr xx:xx:xx:xx:xx
inet addr:172.17.3.9 Bcast:172.17.255.255 Mask:255.255.0.0

eth0:0 Link encap:Ethernet HWaddr xx:xx:xx:xx:xx:xx
inet addr:172.21.0.1 Bcast:172.21.255.255 Mask:255.255.0.0
```

4. On the Test1 and Test2 machines, install the `zebra` RPM from the Red Hat Linux 7.1 distribution, if it is not already installed:

```
# rpm -i zebra-0.91a-3.i386.rpm
```

5. On Test1 Machine and Test2 Machine, configure `zebra` by creating the `zebra.conf`, `ripd.conf`, and `ospfd.conf` configuration files in the `/etc/zebra` directory with the following contents:

- `zebra.conf`

```
password zebra
enable password zebra
interface lo
ip address 127.0.0.1/8
interface eth0
ip address <ip_address>/16
log stdout
```

where `<ip_address>` is `172.17.0.5` for Test1 Machine and `172.21.0.2` for Test2 Machine.

- `ripd.conf`

```
password zebra
enable password zebra
router rip
network eth0
```

```
- ospfd.conf  
password zebra  
enable password zebra  
router ospf
```

6. On Test1 Machine and Test2 Machine, start `zebra` by typing the following command:

```
# zebra -d
```

To demonstrate the RIP or OSPF routing protocols, a respective daemon must be started on a Test Machine:

7. To check the RIP protocol, start the `ripd` daemon on a Test machine:

```
# ripd -d
```

or

To check the OSPF protocol, start the `ospfd` daemon on a Test machine:

```
# ospfd -d
```

After starting either daemon, a Test Machine exchanges information using a respective protocol to create and update the IP routing table. This allows the user to access a Test Machine that belongs to another IP subnet. No additional effort is required to access another subnet.

8. Use the `ping` utility to check a network connectivity. To `ping` from Test1 Machine to Test2 Machine, type the following:

```
# ping 172.21.0.2
```

9. To examine the IP routing tables on the Test machines, use the `netstat` utility:

- On Test 1 Machine:

```
# netstat -nr
```

Destination	Gateway	Genmask	Iface
172.17.0.0	0.0.0.0	255.255.0.0	eth0
172.21.0.0	172.17.3.9	255.255.0.0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	lo
0.0.0.0	172.17.0.1	0.0.0.0	eth0

The above example shows that the IP address `172.21.0.0` and the respective gateway `172.17.3.9` exist in the table.

- On Test 2 Machine:

```
# netstat -nr
Destination Gateway      Genmask      Iface
172.21.0.0  0.0.0.0      255.255.0.0  eth0
172.17.0.0  172.21.0.1  255.255.0.0  eth0
127.0.0.0   0.0.0.0      255.0.0.0    lo
0.0.0.0     172.21.0.1  0.0.0.0      eth0
```

The above example shows that the IP address 172.17.0.0 and the respective gateway 172.21.0.1 exist in the table.

Although this demo system is based on a single Ethernet device, it can be extended to use multiple Ethernet devices.

---

## Using Selected RPM Packages

This section describes how to use selected RPM packages that are frequently deployed in the embedded systems environment.

### Using BusyBox

The BusyBox RPM package combines tiny versions of many common UNIX utilities into a single small executable. It provides minimalist replacements for most utilities found usually in `fileutils`, `shellutils`, `findutils`, `textutils`, `grep`, `gzip`, `tar`, etc. BusyBox provides a fairly complete POSIX environment for any small or embedded system.

The utilities in BusyBox generally have fewer options than their full-featured GNU counterparts; however, the options included provide the expected functionality and behave much like their GNU correlates.

### Creating a BlueCat Linux System for BusyBox

This section describes the steps for creating and booting a BlueCat Linux system containing BusyBox, and demonstrates the use of BusyBox utilities.

1. Create a new directory by typing

```
BlueCat:$ mkdir -p \
    $BLUECAT_PREFIX/demo/busybox/local
```

2. Set up the BlueCat Linux kernel configuration using the standard kernel configuration tools, and copy the kernel configuration file to the `BLUECAT_PREFIX/demo/busybox` directory.

```
BlueCat:$ cd BLUECAT_PREFIX/usr/src/linux
BlueCat:$ make xconfig
BlueCat:$ cp .config \
BLUECAT_PREFIX/demo/busybox/busybox.config
```

---

**NOTE:** The kernel configuration file for the `developer demo` (`BLUECAT_PREFIX/demo/developer/developer.config`) is also recommended as a starting point.

---

3. Create a BlueCat Linux Kernel Downloadable Image, `busybox.kernel`.

```
BlueCat:$ cd BLUECAT_PREFIX/demo/busybox
BlueCat:$ mkkernel ./busybox.config \
./busybox.kernel ./busybox.disk
```

4. Create a specification file (`busybox.spec`) with the following minimal directives:

```
strip on

mkdir /dev
mknod /dev/console c 5 1

mkdir /lib
mkdir -p /usr/lib
mkdir /bin
mkdir /sbin
mkdir -p /etc/rc.d
mkdir /proc

cp ./local/fstab ./local/inittab /etc
cp ./local/rc.sysinit /etc/rc.d

lcd ${BLUECAT_PREFIX}/sbin
cp reboot busybox /sbin

ln -s /sbin/busybox /sbin/init
ln -s /sbin/busybox /sbin/ifconfig
ln -s /sbin/busybox /sbin/route
ln -s /sbin/busybox /bin/mount
ln -s /sbin/busybox /bin/sh
ln -s /sbin/busybox /bin/ping

chmod 711 /etc/rc.d/rc.sysinit
chmod 755 /bin /sbin
# End of File
```

5. Create the `local/fstab` file with the following contents:

```
proc    /proc    proc    defaults    0    0
```

6. Create the `local/inittab` file with the following contents:

```
# System initialization.
::sysinit:/etc/rc.d/rc.sysinit
::respawn:/bin/sh
```

---

**NOTE:** The first two fields in every record of the `inittab` file are ignored by the BusyBox `init`, so they must be empty. For example, the line `1:12345:respawn:/bin/sh` is not valid.

---

7. Create the `local/rc.sysinit` file with the following contents:

```
#!/bin/sh

PATH=/bin:/sbin:/usr/bin:/usr/sbin
export PATH
mount -a
```

8. Create a root file system image (`busybox.rfs`) by entering the following command:

```
BlueCat:$ mkrootfs -lv ./busybox.spec ./busybox.rfs
```

---

**NOTE:** Makefile for the developer demo system can be used as a starting point to produce the BusyBox kernel and RFS images.

---

## Booting BusyBox Images from a Network

Use the following procedure to boot the BlueCat Linux with BusyBox utility from a network using the BlueCat Linux OS loader. Refer to Chapter 2, “Downloading and Booting BlueCat Linux on the Target” for details on the OS loader.

1. At the OS loader prompt (`>`), type the following commands:

```
> set IF eth0
> set IP <target_board_ip>
> set HOST <host_ip>
> set KERNEL tftp busybox.kernel
> set RFS tftp busybox.rfs
> boot
```

Where `<target_board_ip>` is the IP address of the target, and `<host_ip>` is the IP address of the development host.

2. The following output appears:

```
Linux version 2.4.10-1 (bin@build1) (gcc version 2.95.3 20010315
(release)) #5 Tue Dec 18 13:33:08 MSK 2001
Processor: Intel StrongARM-IXP1200 revision 3
Architecture: Intel IXP1200
On node 0 totalpages: 32768
zone(0): 32768 pages.
zone(1): 0 pages.
zone(2): 0 pages.
Kernel command line: ramdisk_size=28472 root=101
Calibrating delay loop... 133.12 BogoMIPS
Memory: 128MB = 128MB total
Memory: 126848KB available (848K code, 208K data, 48K init)
Dentry-cache hash table entries: 16384 (order: 5, 131072 bytes)
Inode-cache hash table entries: 8192 (order: 4, 65536 bytes)
Mount-cache hash table entries: 2048 (order: 2, 16384 bytes)
Buffer-cache hash table entries: 8192 (order: 3, 32768 bytes)
Page-cache hash table entries: 32768 (order: 5, 131072 bytes)
POSIX conformance testing by UNIFIX
PCI: bus0: Fast back to back transfers enabled
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Starting kswapd v1.8
pty: 256 Unix98 ptys configured
IXP1200 serial driver version 1.2
block: queued sectors max/low 84232kB/28077kB, 256 slots per queue
RAMDISK driver initialized: 16 RAM disks of 28472K size 1024
blocksize
eepro100.c:v1.09j-t 9/29/99 Donald Becker
http://cesdis.gsfc.nasa.gov/linux/drivers/eepro100.html
eepro100.c: $Revision: 1.36 $ 2000/11/17 Modified by Andrey V.
Savochkin <saw@saw.sw.com.sg> and others
eth0: OEM i82557/i82558 10/100 Ethernet, 00:90:D7:00:08:D2, I/O at
0x8400, IRQ 11.
Board assembly ffffff-255, Physical connectors present: RJ45
Primary interface chip i82555 PHY #1.
Secondary interface chip i82555.
General self-test: passed.
Serial sub-system self-test: passed.
Internal registers self-test: passed.
ROM checksum self-test: passed (0x1d68d8db).
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP
IP: routing cache hash table of 1024 buckets, 8Kbytes
TCP: Hash tables configured (established 8192 bind 8192)
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
NetWinder Floating Point Emulator V0.95 (c) 1998-1999 Rebel.com
RAMDISK: Compressed image found at block 3147184
Freeing BlueCat RFS memory: 896K
RAMDISK: ext2 filesystem found at block 0
RAMDISK: Loading 2467 blocks [1 disk] into ram disk... done.
VFS: Mounted root (ext2 filesystem).
Freeing init memory: 48K
init started: BusyBox v0.52pre (2001.12.05-11:36+0000) multi-call
binary

hush -- the humble shell v0.01 (testing)
```

## Using BusyBox Utilities

This section provides the examples of using the BusyBox utilities. Entering a command from the following list results in the respective output:

- `ls`

```
/ # ls /
bin      etc      lost+found  sbin
dev      lib      proc        usr
```

- `cat`

```
/ # cat /etc/inittab
# System initialization
::sysinit:/etc/rc.d/rc.sysinit

::respawn:/bin/sh
```

- `chmod`

```
/ # chmod a-x /sbin/reboot
/ # ls -la /sbin/reboot
-rw-r--r--  1 0      0          7812 Dec 18  2001 /sbin/reboot
/ # chmod 755 /sbin/reboot
/ # ls -la /sbin/reboot
-rwxr-xr-x  1 0      0          7812 Dec 18  2001 /sbin/reboot
```

- `echo`

```
/ # echo !!!!
!!!!
```

- `date`

```
/ # date
Thu Jan  1 00:06:57 UTC 1970
```

- `uname`

```
/ # uname -a
Linux (none) 2.4.10-1 #5 Tue Dec 18 13:33:08 MSK 2001 armv4b unknown
```

- `mount`

```
/ # mount
/dev/root on / type ext2 (rw)
proc on /proc type proc (rw)
```

- `ifconfig`

```
/ # ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:90:D7:00:08:D2
          BROADCAST MULTICAST  MTU:1500  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:100
RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
Interrupt:11 Base address:0x8400

/ # ifconfig eth0 172.17.3.12
/ # ping -c 5 172.17.0.1
PING 172.17.0.1 (172.17.0.1): 56 data bytes
64 bytes from 172.17.0.1: icmp_seq=0 ttl=255 time=0.7 ms
64 bytes from 172.17.0.1: icmp_seq=1 ttl=255 time=0.4 ms
64 bytes from 172.17.0.1: icmp_seq=2 ttl=255 time=0.4 ms
64 bytes from 172.17.0.1: icmp_seq=3 ttl=255 time=0.4 ms
64 bytes from 172.17.0.1: icmp_seq=4 ttl=255 time=0.4 ms

--- 172.17.0.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.4/0.7 ms
```

## Using TinyLogin RPM Package

The TinyLogin RPM package is a suite of tiny UNIX utilities for handling logging into, being authenticated by, changing one's password for, and otherwise maintaining users and groups on an embedded system. It also provides shadow password support to enhance system security.

This section describes the steps necessary for creating and booting a BlueCat Linux system containing TinyLogin and demonstrates use of the utility.

## Creating a BlueCat Linux System for TinyLogin

Use the following procedure to create a BlueCat Linux image for TinyLogin:

1. Create a new directory:

```
BlueCat:$ mkdir -p \
    $BLUECAT_PREFIX/demo/tinylogin/local
```

2. Set up the BlueCat Linux kernel configuration by using the standard kernel configuration tools, and copy the kernel configuration file to the \$BLUECAT\_PREFIX/demo/tinylogin directory.

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

```
BlueCat:$ cp .config \
           $BLUECAT_PREFIX/demo/tinylogin/tinylogin.config
```

---

**NOTE:** The kernel configuration file for the developer demo (\$BLUECAT\_PREFIX/demo/developer/developer.config) is also recommended as a starting point.

---

3. Create the BlueCat kernel downloadable image (tinylogin.kernel):

```
BlueCat:$ cd $BLUECAT_PREFIX/demo/tinylogin
BlueCat:$ mkkernel ./tinylogin.config \
           ./tinylogin.kernel ./tinylogin.disk
```

4. Create a specification file (tinylogin.spec) that contains the following minimal directives:

```
strip on

mkdir /dev
mknod /dev/console c 5 1
ln -s /dev/console /dev/tty
ln -s /dev/console /dev/ttyl

mkdir /bin
mkdir /sbin
mkdir -p /etc/rc.d
mkdir /proc
mkdir /tmp
mkdir -p /usr/bin

mkdir /root

mkdir /dev/pts
mknod /dev/ptmx c 5 2

chmod 0666 /dev/ptmx

cp ./local/fstab ./local/passwd ./local/inittab /etc
cp ./local/securetty ./local/shadow /etc
cp ./local/rc.sysinit /etc/rc.d
cp ${BLUECAT_PREFIX}/etc/shells /etc
chmod 644 /etc/shells
cp ${BLUECAT_PREFIX}/etc/group /etc

lcd ${BLUECAT_PREFIX}/sbin
cp reboot init mingetty /sbin

cp ${BLUECAT_PREFIX}/usr/bin/tinylogin /usr/bin
ln -s /usr/bin/tinylogin /usr/bin/passwd
ln -s /usr/bin/tinylogin /bin/login

lcd ${BLUECAT_PREFIX}/bin
cp mount bash ls cat hostname /bin
ln -s /bin/bash /bin/sh

chmod 711 /etc/rc.d/rc.sysinit
```

```
chmod 755 /bin /sbin /usr/bin

chmod 04755 /usr/bin/tinylogin
# End of File
```

---

**NOTE:** In this `.spec` file, the `/bin/login` and `/usr/bin/passwd` symbolic links point to `/usr/bin/tinylogin`. This allows the user to change his/her password by simply typing **passwd**.

---

5. Create the `local/fstab` file with the following contents:

```
none /proc      proc
none /dev/pts   devpts
```

6. Create the `local/inittab` file with the following contents:

```
id:1:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

1:12345:respawn:/sbin/mingetty tty1
```

7. Create the `local/securetty` file with the following contents:

```
console
tty1
```

8. Create the `local/passwd` file with the following contents:

```
root:x:0:0:/root:/bin/bash
guest:x:500:10::/bin/bash
```

9. Create the `local/shadow` file:

```
root::10942:0:99999:7:::
guest::500:10:99999:7:::
```

10. Create the `local/rc.sysinit` file with the following contents:

```
#!/bin/sh

PATH=/bin:/sbin:/usr/bin:/usr/sbin
export PATH

mount -a
hostname myhostname
```

11. Create a root file system image (`tinylogin.rfs`) by entering the following command:

```
BlueCat:$ mkrootfs -lv ./tinylogin.spec \  
./tinylogin.rfs
```

---

**NOTE:** Makefile for the `developer` demo system can be used as a starting point to produce the TinyLogin kernel and RFS images.

---

## Booting the TinyLogin Images from a Network

Use the following procedure to boot BlueCat Linux with the TinyLogin utility from a network using the BlueCat Linux OS loader. Refer to Chapter 2, “Downloading and Booting BlueCat Linux on the Target” for additional details about the BlueCat Linux OS loader.

At the OS loader prompt (`>`), type the following commands:

```
> set IF eth0  
> set IP <target_board_ip>  
> set HOST <host_ip>  
> set KERNEL tftp tinylogin.kernel  
> set RFS tftp tinylogin.rfs  
> boot
```

where `<target_board_ip>` is the IP address of the target, and `<host_ip>` is the IP address of the development host.

The TinyLogin utility is loaded onto the target board and started automatically.

## Using TinyLogin

This section provides the examples of using the TinyLogin utility:

- Changing the guest password:

```
myhostname login: guest  
bash-2.04$ passwd  
Changing password for guest  
Enter the new password (minimum of 5, maximum of 8 characters)  
  
Please use a combination of upper and lower  
case letters and numbers.  
Enter new password: <new_guest_password>
```

```
Re-enter new password: <new_guest_password>
passwd[13]: password for 'guest' changed by user 'guest'
Password changed.
bash-2.04$ exit
myhostname login: guest
Password: <new_guest_password>
bash-2.04$ exit
```

- Changing root password:

```
myhostname login: root
login[16]: root login on 'console'

bash-2.04# passwd
Changing password for root
Enter the new password (minimum of 5, maximum of 8 characters)

Please use a combination of upper and lower
case letters and numbers.
Enter new password: <new_root_password>
Re-enter new password: <new_root_password>
passwd[17]: password for 'root' changed by user 'root'
Password changed.
bash-2.04# exit
myhostname login: root
Password: <new_root_password>
login[18]: root login on 'console'

bash-2.04# exit
```

- Getting root permissions:

```
myhostname login: guest
Password: <guest_password>
bash-2.04$ tinylogin su
Password:
login[17]: root login on 'console'

bash-2.04#
```

The following table shows the device drivers supported by the ixp1200 BSP.

**Table 5-1: Device Drivers Supported by the ixp1200 BSP**

Hardware Device	Device Drivers	Location in Source Tree	Kernel Configuration Options
<b>UART</b> IXDP1200 built-in serial controller	serial_ixp1200.c	drivers/char	CONFIG_SERIAL_IXP1200 CONFIG_SERIAL_IXP1200_CONSOLE
<b>LED Display</b>	led_ixp1200.c led.h	drivers/char include/linux	CONFIG_LED_IXP1200
<b>Ethernet</b> Intel 82559 Adapter	eepr0100.c	drivers/net	CONFIG_EEPRO100
<b>NOR Flash</b>	ixp1200.c	drivers/mtd/maps	CONFIG_MTD_IXP1200 CONFIG_MTD_IXP1200_PART
<b>Real Time Clock</b>	rtc_ixp1200.c rtc.h	drivers/char include/linux	CONFIG_IXP1200_RTC

## Flash Support

The BlueCat Linux Flash/JFFS management sub system is extended with support for the specific Flash devices present on the board. This provides a standard set of Flash management features available in BlueCat Linux. For a detailed description of Flash memory support and the Journalling Flash File System (JFFS) in BlueCat Linux refer to Chapter 6 “Flash Support and Journalling Flash File System” in the *BlueCat Linux User’s Guide*.

## Configuring Custom Flash Chips

The following files in the BlueCat Linux kernel must be updated to support custom Flash chips:

- `include/asm-arm/arch-ixp1200/hardware.h`

Change the following line:

```
#define FLASH_BASE UNSIGNED (0xf3000000)
```

to

```
#define FLASH_BASE <virtual_address>
```

where *<virtual\_address>* is a virtual address to which the custom Flash chip is to be mapped. (If the chip size is 8Mb or less, then the `FLASH_BASE` value can be used.)

- `include/asm-arm/arch-ixp1200/ixp1200.h`

Change the following line:

```
#define PHYS_FLASH_BASE UNSIGNED 0x00000000
```

to

```
#define PHYS_FLASH_BASE <physical_address>
```

where *<physical\_address>* is an address of the custom Flash chip on the internal bus.

- `include/asm-arm/arch-ixp1200/board.h`

Change the following line:

```
#define PHYS_FLASH_SIZE 0x00800000
```

to

```
#define PHYS_FLASH_SIZE <size>
```

where *<size>* is the size of the custom Flash.

The values used in the ixp1200 BSP are as follows:

```
<physical_address> 0x00000000
```

```
<virtual_address> 0xF3000000
```

```
<size> 0x00800000
```

- `drivers/mtd/maps/ixp1200.c`

Change the following line:

```
buswidth:      2,
```

to

```
buswidth:      <bus_width>,
```

where `<bus_width>` is the width of the Flash bus.

If the custom Flash chips do not support CFI autodetection, then a detection module must be written for this device. For detailed information on how to write the detection module, refer to Chapter 6 “Flash Support and Journalling Flash File System” in the *BlueCat Linux User’s Guide*.



This chapter describes the BlueCat Linux Application Programming Interface (API) specific to the IXDP1200 board.

---

## LED Display Control

BlueCat Linux for IXDP1200 has a device driver for the on-board LED display. The driver is configured using the `CONFIG_LED_IXP1200` kernel configuration option, which can be enabled in the **Character Devices** submenu of the `make xconfig` interface. The driver provides an `ioctl()` interface for the user-space applications, defined in the `$BLUECAT_PREFIX/usr/src/linux/include/linux/led.h` file.

The driver is accompanied by the `$BLUECAT_PREFIX/bin/bluecat_led` utility that provides control of the LED display from the shell command line or shell scripts. The utility can be used to control the LED display on the target. The utility syntax is as follows:

```
bluecat_led <value>
```

where `<value>` is a numerical value to be displayed on the LED.

For example:

```
# bluecat_led 1.2
# bluecat_led 55
```

Please note that the `bluecat_led` utility requires that an LED driver special device file (`/dev/led`) be present in the root file system. The major number of the device is 233. For example, the special `/dev/led` file can be created by the following command:

```
mknod /dev/led c 223 0
```

Also, the LED display driver provides the standard direct interface to allow control of the LED display from the shell command line or shell scripts. This feature can be used to control the LED display on the target. The syntax of the command is as follows:

```
echo <value> >/dev/led
```

where <value> is a numerical value to be displayed on the LED. For example:

```
# echo 57 >/dev/led
```

# Defect Fixes and Known Limitations

The table below shows defect fixes in this release of BlueCat Linux:

**Table 7-1: Defect Fixes in BlueCat Linux**

Platform	Subcomponent	ID	Summary
All	BlueCat Linux Misc	16057	<b>Ctrl-C, Ctrl-Z</b> , etc., do not work from shell.
All	BlueCat Linux OS loader	16358	The BLOSH <b>ntar</b> command hangs the system.
All	BlueCat Linux Misc	17308	Certain sequence of file updates sometimes causes the FFS to crash after reboot.
All	BlueCat Linux Debuggers	18404	Update BlueCat <b>gdbserver</b> to process signals to <b>gdb</b> on the host correctly.
ARM	BlueCat Linux Drivers	17677	Error generated when <code>jffs_garbage_collect_thread():free_size == 0</code>

## IXDP1200 Limitations and Workarounds

- Only 4 MB of the on-board Flash memory is supported. There are two 4 MB Flash chips on the IXDP1200 board that share the 32-bit BootROM data bus. Bits 0-15 are used by the first chip and bits 16-31 are used by the second chip. The GPIO[3] pin allows the user to select between two modes: 16-bit and 32-bit. Because of firmware limitations, the boot procedure works in the 16-bit mode only, and it is impossible to switch to the 32-bit mode on the operational board. So, the second Flash chip is not accessible.

- Kernel debugger (KDBG) has not been tested, as there is only one serial port on the board.
- Only the `osloader` demo system can be programmed into Flash. The size of the `showcase`, `developer` and `zebra` demo systems is larger than the size of supported Flash memory (4 MB).
- Debugging of multithreaded applications via GDB is not supported.
- When executed on the test machines as described in the BlueCat Linux IXDP1200 BSG, the `zebra` demo prints the following message onto the system console:

```
ZEBRA: can't create router advertisement socket: Address family not supported by protocol
```

Despite this message, the `zebra` demo continues to operate as described in the BSG.

- GDB debugging over serial line is not supported, as there is only one serial port on the board.
- If `mkrootfs` is terminated (either by error or by a signal), it tries to clean all its temporary files before exiting. However, due to certain features of the Cygwin execution environment, these temporary files can remain uncleared in the `/tmp` directory on a Windows host. It is recommended that the `/tmp` directory be regularly checked and cleaned.
- The `tclx` RPM package is not included in the Windows-hosted distribution.
- On Windows hosts, some file permissions (including `r` and `s`) always have default values. To set permissions different from the default values, the `chmod` command should be used in the `.spec` file.
- The supported/tested version of Cygmon is 2.05.