

BlueCat Linux Board Support Guide

BlueCat Linux 4.0
DOC-0567-00

for the GbE HBA Board

Product names mentioned in *BlueCat Linux Board Support Guide for the GbE HBA Board* are trademarks of their respective manufacturers and are used here for identification purposes only.

Copyright ©1987 - 2003, 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 the GbE HBA Board* 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.)

Contents

	Typographical Conventions	v
	Special Notes	vi
	Technical Support	vi
CHAPTER 1	OVERVIEW	1
CHAPTER 2	DOWNLOADING AND BOOTING BLUECAT LINUX ON THE TARGET	3
	Supported Hosts	3
	Prerequisites	3
	Downloading and Booting Overview	4
	Setting up Hardware	5
	Connecting Target Board Serial Port to Host	5
	Connecting Target Board Ethernet Cards to Host	5
	The RedBoot Enhanced Autoboot Feature	6
	Setting up the RedBoot Firmware	6
	Downloading a BlueCat System to Flash	7
	Booting a Demo System from Flash	9
	Booting a Demo System from a Network	9
	Booting a Demo System from a Network Using RedBoot	9
	Booting a Demo System from a Network Using OS Loader	10
CHAPTER 3	KERNEL CONFIGURATION OPTIONS	11
CHAPTER 4	SUPPORTED DEMO SYSTEMS.....	31
	Demo Systems	31
	developer Demo System	31
	osloader Demo System	32

	showcase Demo System	32
	Using Selected RPM Packages	32
	Using BusyBox RPM Package	32
	Using TinyLogin RPM Package	36
	Using Zebra RPM Package	41
<hr/>	CHAPTER 5 SUPPORTED DEVICE DRIVERS	47
<hr/>	CHAPTER 6 BLUECAT LINUX-SPECIFIC APIS.....	49
	Advanced MMU/Cache Management Services	49
	Advanced MMU/Cache Management Services Overview	49
	Advanced MMU/Cache Management Services Reference	50
<hr/>	CHAPTER 7 KNOWN PROBLEMS AND LIMITATIONS.....	53
	GbE HBA Board Problems and Limitations	53

Preface

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; *italicized* for emphasis, new terms, and book titles

Refer to the *BlueCat Linux User's Guide*.

Environment variables, file names, functions, methods, options, parameter names, path names, commands, and computer data

```
ls
-l
myprog.c
/dev/null
```

Commands that need to be highlighted within body text, or commands that must be typed as is by the user are **bolded**.

```
login: myname
# cd /usr/home
```

Text that represents a variable, such as a file name or a value that must be entered by the user

```
cat <filename>
mv <file1> <file2>
```

Blocks of text that appear on the display screen after entering instructions or commands

```
Loading file /tftpboot/shell.kdi
into 0x4000
.....
File loaded. Size is 1314816
Copyright 2002 LynuxWorks, Inc.
All rights reserved.

LynxOS (ppc) created Mon Jan 17
17:50:22 GMT 2002
user name:
```

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

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

fax (408) 979-3945

LynuxWorks Europe

email 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 the GbE HBA Board* provides information about the BlueCat Linux Board Support Package (BSP) for the ADI GbE HBA board. The ADI GbE HBA board is a PCB employing the Intel 80200 processor based on the XScale architecture.

Throughout this Board Support Guide (BSG), the BSP is referred to as the “gbe_hba” and the board as the “GbE HBA,” or simply as the “target board.”

The chapters of this BSG provide the information listed below:

- *Chapter 1* is an overview of the individual chapters.
- *Chapter 2* describes BlueCat Linux downloading and booting procedures for the GbE HBA board using the BlueCat Linux `osloader` and `showcase` demo systems as examples.
- *Chapter 3* provides configuration option information about the gbe_hba BSP’s default BlueCat Linux kernel.
- *Chapter 4* summarizes BlueCat Linux demo systems supported by the gbe_hba BSP.
- *Chapter 5* provides a list of gbe_hba BSP-supported device drivers, with important information about each of them.
- *Chapter 6* describes BlueCat Linux-specific Application Programming Interfaces (APIs) for the GbE HBA board.
- *Chapter 7* describes known limitations and workarounds for this release.

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 and then booting the demo system on the target board.

Supported Hosts

BlueCat Linux for the GbE HBA board is supported on the hosts listed in Table 2-1.

Table 2-1: Supported Hosts

Host	Target Family
Windows 2000/Pro with SP1 or later	ARM/little endian
Windows XP	ARM/little endian
PC running Red Hat Linux 7.1	ARM/little endian
PC running Red Hat Linux 7.2	ARM/little endian
SuSE 7.2	ARM/little endian
SuSE 7.3	ARM/little endian

Prerequisites

This document is a guide to downloading and booting BlueCat Linux systems onto the user's target board. Scenarios that use demo systems included in the BlueCat Linux distribution are presented. As such, a basic familiarity with the target board

hardware and operation is required before using this guide. The user must also have an understanding of system administration for the particular cross-development host on which BlueCat Linux and the BSP are installed. It is assumed that the user has the manufacturer's documentation for the target board as well as the system administration reference material for the cross-development host.

Before downloading and booting BlueCat Linux on the target board, it is assumed that the default BlueCat Linux XScale configuration and the gbe_hba BSP have been installed on the cross-development host. This means that the user must:

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

Downloading and Booting Overview

The procedure for downloading and booting a BlueCat Linux system on the GbE HBA target consists of the following main steps:

- Setting up hardware
- Downloading and booting a BlueCat Linux system from target Flash memory or a network

Downloading and booting of a BlueCat Linux system can be performed using either the RedBoot firmware or the BlueCat OS Loader demo system.

The OS Loader demo system includes the `i_osloader` and the `osloader` downloadable images. `osloader` is the image with the base functionality of the BlueCat OS Loader configured in. This includes the ability to download BlueCat images from a TFTP host, execute them in RAM, and other important features. `i_osloader` is extended with support for the Journalling Flash File System (JFFS) and can thus be used to download a desired BlueCat Linux custom or demo system into the target board's Flash memory.

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

Setting up Hardware

Connecting Target Board Serial Port to Host

The target board serial port is used both by the RedBoot firmware and the BlueCat Linux system console. Before using the board, the serial port needs to be connected to the development host. It is recommended to connect the target serial connector to COM1 on the host.

The serial port on the host must be configured to a baud rate of 115200.

Throughout this chapter, the terminal window connected to the serial connector is referred to as the RedBoot console or BlueCat Linux console, depending on the context.

Connecting Target Board Ethernet Cards to Host

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

There are two Ethernet ports on the target board. The user must use at least the first port to connect the GbE HBA to a LAN.

It is also required that the user set up networking on the host system. In particular, the user must choose a unique IP address for the development host as well as for the target board. These addresses are referred to as *development_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.

TFTP 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.

The RedBoot Enhanced Autoboot Feature

The RedBoot firmware version 2.20 installed on the GbE HBA has an enhanced autoboot feature that allows booting BlueCat Linux embedded systems automatically at board power-up. To disable this feature, do either of the following:

- Ground the MP2 pin.
- Write 0xFFFFFFFF at the address 0x80000 by issuing the following command at the RedBoot console:

```
RedBoot> fis erase -f 0x80000 -l 0x20000
```

Setting up the RedBoot Firmware

To set up the RedBoot firmware options for BlueCat Linux operations (enhanced autoboot is disabled), perform the following steps:

1. Reset the target board.
2. At the RedBoot console, enter:

```
RedBoot> fconfig
Run script at boot: false
Use BOOTP for network configuration: false Enter
Gateway IP address: Enter
Local IP address: target_board_IP
Local IP address mask: 255.255.255.0 Enter
Default server IP address: development_host_IP
Console baud rate: 115200
DNS server IP address: Enter
GDB connection port: 9000 Enter
Force console for special debug messages: false Enter
Network debug at boot time: false Enter
Update RedBoot non-volatile configuration - continue (y/n)? y
... Unlock from 0x007c0000-0x007c1000: .
... Erase from 0x007c0000-0x007c1000: .
... Program from 0xcffd2000-0xcffd3000 at 0x007c0000: .
... Lock from 0x007c0000-0x007c1000: .
```

where *target_board_IP* is the IP address of the target and *development_host_IP* is the IP address of the development host.

3. Reset the target board.

Downloading a BlueCat System to Flash

This section provides instructions on how a BlueCat embedded system can be downloaded into the target Flash memory using the BlueCat Linux OS Loader. For additional details about the BlueCat Linux OS Loader, refer to the *BlueCat Linux User's Guide*.

Specifically, these instructions are applicable to any of the demo systems. This chapter uses the `osloader` demo system as an example.

The following figure shows how the Flash memory on the GbE HBA board is partitioned for BlueCat Linux.

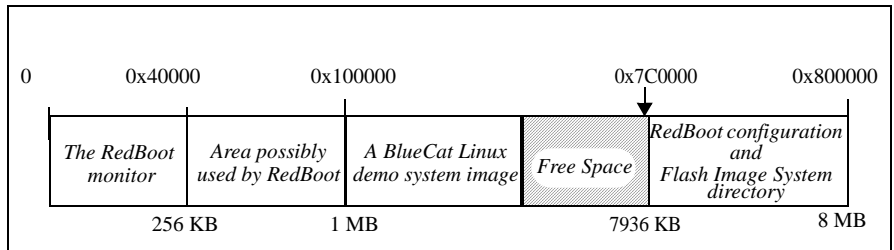


Figure 2-1: Partitioning of the Flash Memory

To download the `osloader` demo system into the target board, perform the following steps:

1. Copy the `i_osloader.kdi` file from the `BLUECAT_PREFIX/demo/osloader` directory to the `/tftpboot` directory on the development host.
2. Copy the `osloader.kdi` file from the `BLUECAT_PREFIX/demo/osloader` directory to the `/tftpboot` directory on the development host.
3. Reset the target board.
4. At the RedBoot console, enter the following commands:

```
RedBoot> load -r -b 0xC0200000 i_osloader.kdi
RedBoot> go 0xC0200000
```

These commands load the `i_osloader` system to RAM and start it. As a result, the BlueCat OS Loader prompt will appear in the BlueCat console.

5. At the BlueCat OS Loader prompt, type the following commands:

```
> set IF eth0
> set IP target_board_IP
> set HOST development_host_IP
> set FILE tftp osloader.kdi
> exec flash_fdisk /dev/mtdchar0 8-15
> flash /dev/mtdchar1 erase
> reset
```

where `target_board_IP` is the IP address of the target and `development_host_IP` is the IP address of the development host.

After these commands have been performed, the `osloader` demo is programmed into Flash and can be booted as described in “Booting a Demo System from Flash” section below.

Booting a Demo System from Flash

NOTE: To start a demo system programmed into Flash automatically on board power-up, the user must unground the MP2 pin.

The following procedure is used to boot the `osloader` demo installed into the Flash memory. For detailed information on how to install the demo system to Flash, refer to the “Downloading a BlueCat System to Flash” section.

1. Reset the target board.
2. At the RedBoot console, type the following:

```
RedBoot> go 0x100000
```

This command will start the `osloader` demo system programmed into Flash.

Booting a Demo System from a Network

A BlueCat Linux demo system can be booted from a network using either the RedBoot firmware or OS Loader.

Booting a Demo System from a Network Using RedBoot

The following example demonstrates booting the `showcase` demo system over a network using the RedBoot firmware.

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

```
RedBoot> load -r -b 0xC0200000 showcase.kdi
```

```
RedBoot> go 0xC0200000
```

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

Booting a Demo System from a Network Using OS Loader

The following example demonstrates booting the `showcase` demo system over a network using OS Loader.

1. Copy the `osloader.kdi` file from the `$(BLUECAT_PREFIX)/demo/osloader` directory to the `/tftpboot` directory on the development host.
2. Copy the `showcase.kernel` and `showcase.rfs` files from the `$(BLUECAT_PREFIX)/demo/showcase` directory to the `/tftpboot` directory on the cross-development host.
3. Reset the target board.
4. At the RedBoot console, enter the following commands:

```
RedBoot> load -r -b 0xC0200000 osloader.kdi
RedBoot> go 0xC0200000
```

These commands start the `osloader` demo system from RAM. As a result, the BlueCat OS Loader prompt will appear in the BlueCat console.

5. At the BlueCat OS Loader prompt, enter the following commands:

```
> set IF eth0
> set IP target_board_IP
> set HOST development_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 `development_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 then automatically start it.

Kernel Configuration Options

The gbe_hba BSP comes with a default BlueCat Linux kernel. This kernel has a number of configuration options. This chapter details these options in the tables listed in Table 3-1: "BlueCat Linux Default Configuration for the gbe_hba BSP Distribution" below.

Table 3-1: BlueCat Linux Default Configuration for the gbe_hba BSP Distribution

Table Number and Configuration Parameter
Table 3-2: "Code Maturity Level Options"
Table 3-3: "Loadable Module Support"
Table 3-4: "System Type"
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: "NAND Flash Device Driver"
Table 3-12: "Plug and Play Configuration"
Table 3-13: "Block Devices"
Table 3-14: "Multidevice Support (RAID and LVM)"
Table 3-15: "Networking Options"
Table 3-16: "QoS and/or Fair Queueing"
Table 3-17: "Network Device Support"

Table 3-1: BlueCat Linux Default Configuration for the gbe_hba BSP Distribution (Continued)

Table Number and Configuration Parameter
Table 3-18: "Arcnet Devices"
Table 3-19: "Ethernet (10 or 100Mbit)"
Table 3-20: "Ethernet (1000 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 (Experimental)"
Table 3-29: "I2O Device 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: "Ftape, the Floppy Tape Device Driver"
Table 3-40: "Multimedia Devices"
Table 3-41: "File System"
Table 3-42: "Network File Systems"
Table 3-43: "Partition Types"

Table 3-1: BlueCat Linux Default Configuration for the gbe_hba BSP Distribution (Continued)

Table Number and Configuration Parameter
Table 3-44: "Sound"
Table 3-45: "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 3-2: 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 module symbols
CONFIG_KMOD	Y	Kernel module loader

Table 3-4: System Type

Option	Value	Description
CONFIG_ARCH_GBE_HBA	Y	ARM system type

Table 3-5: General Setup

Option	Value	Description
CONFIG_ANGELBOOT	N	Load kernel using Angel Debug Monitor
CONFIG_BLUECAT_IGNORE_PRINTK	N	BlueCat Ignore printk
CONFIG_BLUECAT_THUMB	N	BlueCat kernel support for THUMB binaries
CONFIG_BLUECAT_LOADER	N	BlueCat OS Loader
CONFIG_BLUECAT_SMALL_FOOTPRINT	N	BlueCat 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	Y	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 (/proc/kcore) format
CONFIG_BINFMT_AOUT	N	Kernel support for a.out binaries
CONFIG_BINFMT_ELF	Y	Kernel support for ELF binaries
CONFIG_BINFMT_MISC	N	Kernel support for MISC binaries

Table 3-5: General Setup (Continued)

Option	Value	Description
CONFIG_PM	N	Power Management support (experimental)
CONFIG_ARTHUR	N	RISC OS personality
CONFIG_ALIGNMENT_TRAP	Y	Kernel-mode alignment trap handler

Table 3-6: Parallel Port Support

Option	Value	Description
CONFIG_PARPORT	N	Parallel port support

Table 3-7: 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 char 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 3-8: RAM/ROM/Flash Chip Drivers

Option	Value	Description
CONFIG_MTD_CFI	Y	Detect Flash chips by Common Flash Interface (CFI) probe
CONFIG_MTD_JEDEC	N	Detect non-CFI AMD/JEDEC-compatible Flash chips
CONFIG_MTD_CFI_ADV_OPTIONS	N	Flash chip driver advanced configuration options
CONFIG_MTD_CFI_INTELEXT	Y	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 3-9: Mapping Drivers for Chip Access

Option	Value	Description
CONFIG_MTD_PHYSMAP	N	CFI Flash device in physical memory map
CONFIG_MTD_NORA	N	CFI Flash device mapped on Nora
CONFIG_MTD_ARM_INTEGRATOR	N	CFI Flash device mapped on ARM Integrator/P720T
CONFIG_MTD_CDB89712	N	Cirrus CDB89712 evaluation board mappings
CONFIG_MTD_GBE_HBA	Y	Flash chip mapping on the GbE HBA board
CONFIG_MTD_GBE_HBA_PART	:	Partitions layout
CONFIG_MTD_PCI	N	PCI MTD driver

Table 3-10: Self-contained MTD Device Drivers

Option	Value	Description
CONFIG_MTD_PMC551	N	Ramix PMC551 PCI Mezzanine RAM card support
CONFIG_MTD_SLRAM	N	Uncached system RAM
CONFIG_MTD_MTDRAM	N	Test driver using RAM
CONFIG_MTD_BLKMTD	N	MTD emulation using block device
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 (see help)

Table 3-11: NAND Flash Device Driver

Option	Value	Description
CONFIG_MTD_NAND	N	NAND device support

Table 3-12: Plug and Play Configuration

Option	Value	Description
CONFIG_PNP	N	Plug and Play support

Table 3-13: Block Devices

Option	Value	Description
CONFIG_BLK_DEV_FD	N	Normal PC floppy disk support
CONFIG_BLK_CPQ_DA	N	Compaq SMART2 support

Table 3-13: Block Devices (Continued)

Option	Value	Description
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_RAM	Y	RAM disk support
CONFIG_BLK_DEV_RAM_SIZE	28472	Default RAM disk size
CONFIG_BLK_DEV_INITRD	N	Initial RAM disk (initrd) support
CONFIG_BLUECAT_RFS	Y	BlueCat RFS support

Table 3-14: Multidevice Support (RAID and LVM)

Option	Value	Description
CONFIG_MD	N	Multiple devices driver support (RAID and LVM)
CONFIG_MD_LINEAR	N	Linear (append) mode
CONFIG_MD_RAID0	N	RAID-0 (striping) mode
CONFIG_MD_RAID1	N	RAID-1 (mirroring) mode
CONFIG_MD_RAID5	N	RAID-4/RAID-5 mode
CONFIG_MD_MULTIPATH	N	Multipath I/O support

Table 3-15: Networking Options

Option	Value	Description
CONFIG_PACKET	N	Packet socket
CONFIG_NETLINK	N	Kernel/user netlink socket
CONFIG_NETFILTER	N	Network packet filtering (replaces ipchains)

Table 3-15: Networking Options (Continued)

Option	Value	Description
CONFIG_FILTER	N	Socket filtering
CONFIG_UNIX	Y	UNIX domain sockets
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	IP: TCP Explicit Congestion Notification support
CONFIG_SYN_COOKIES	N	IP: TCP syncookie support (disabled 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
CONFIG_X25	N	CCITT X.25 Packet Layer (experimental)
CONFIG_LAPB	N	LAPB Data Link Driver (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

Table 3-15: Networking Options (Continued)

Option	Value	Description
CONFIG_NET_FASTROUTE	N	Fast switching (see help)
CONFIG_NET_HW_FLOWCONTROL	N	Forwarding between high speed interfaces

Table 3-16: QoS and/or Fair Queueing

Option	Value	Description
CONFIG_NET_SCHED	N	QoS and/or fair queueing

Table 3-17: 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 device driver support
CONFIG_FDDI	N	FDDI driver support
CONFIG_HIPPI	N	HIPPI driver support (experimental)
CONFIG_PPP	N	Point-to-Point Protocol (PPP) support
CONFIG_SLIP	N	Serial Line Internet Protocol (SLIP) support
CONFIG_NET_FC	N	Fibre Channel driver support
CONFIG_RCPCI	N	Red Creek Hardware VPN (experimental)
CONFIG_SHAPER	N	Traffic Shaper (experimental)

Table 3-18: Arcnet Devices

Option	Value	Description
CONFIG_ARCNET	N	ARCnet support

Table 3-19: Ethernet (10 or 100Mbit)

Option	Value	Description
CONFIG_NET_ETHERNET	N	Ethernet (10 or 100Mbit)

Table 3-20: Ethernet (1000 Mbit)

Option	Value	Description
CONFIG_ACENIC	N	Alteon AceNIC/3Com 3C985/NetGear GA620 Gigabit support
CONFIG_DL2K	N	D-Link DL2000-based Gigabit Ethernet support
CONFIG_NS83820	N	National Semiconduct DP83820 support
CONFIG_HAMACHI	N	Packet Engines Hamachi GNIC-II support
CONFIG_YELLOWFIN	N	Packet Engines Yellowfin Gigabit-NIC support (experimental)
CONFIG_SK98LIN	N	SysKonnect SK-98xx support
CONFIG_E1000	Y	Intel PRO/1000 support

Table 3-21: Wireless LAN (Non-Ham Radio)

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

Table 3-22: Token Ring Devices

Option	Value	Description
CONFIG_TR	N	Token Ring driver support

Table 3-23: WAN Interfaces

Option	Value	Description
CONFIG_WAN	N	WAN interfaces support

Table 3-24: Amateur Radio Support

Option	Value	Description
CONFIG_HAMRADIO	N	Amateur Radio support

Table 3-25: IrDA (Infrared) Support

Option	Value	Description
CONFIG_IRDA	N	IrDA subsystem support

Table 3-26: ATA/IDE/MFM/RLL Support

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

Table 3-27: SCSI Support

Option	Value	Description
CONFIG_SCSI	N	SCSI support?

Table 3-28: IEEE 1394 (FireWire) Support (Experimental)

Option	Value	Description
CONFIG_IEEE1394	N	IEEE 1394 (Firewire) Support (experimental)

Table 3-29: I2O Device Support

Option	Value	Description
CONFIG_I2O	N	I2O support

Table 3-30: ISDN Subsystem

Option	Value	Description
CONFIG_ISDN	N	ISDN support

Table 3-31: 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 3-32: Character Devices

Option	Value	Description
CONFIG_VT	N	Virtual terminal
CONFIG_SERIAL	Y	Standard/generic (8250/16550 and compatible UARTs) serial support
CONFIG_SERIAL_CONSOLE	Y	Support for console on serial port
CONFIG_SERIAL_EXTENDED	N	Extended dumb serial driver options
CONFIG_SERIAL_NONSTANDARD	N	Nonstandard serial port support
CONFIG_UNIX98_PTYS	Y	Unix98 PTY support
CONFIG_UNIX98_PTY_COUNT	32	Maximum number of Unix98 PTYs in use (0-2048)
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_RTC_DS1307	N	DS1307 Real Time Clock
CONFIG_DTLK	N	Double Talk PC internal speech card support
CONFIG_R3964	N	Siemens R3964 line discipline
CONFIG_APPLICOM	N	Applicom intelligent fieldbus card support

Table 3-33: Serial Drivers

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

Table 3-34: I2C Support

Option	Value	Description
CONFIG_I2C	N	I ² C support

Table 3-35: L3 Serial Bus Support

Option	Value	Description
CONFIG_L3	N	L3 support

Table 3-36: Mice

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

Table 3-37: Joysticks

Option	Value	Description
CONFIG_QIC02_TAPE	N	QIC-02 tape support

Table 3-38: Watchdog Cards

Option	Value	Description
CONFIG_WATCHDOG	N	Watchdog Timer support

Table 3-39: Ftape, the Floppy Tape Device Driver

Option	Value	Description
CONFIG_FTAPE	N	Ftape (QIC-80/Travan) 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 3-40: Multimedia Devices

Option	Value	Description
CONFIG_VIDEO_DEV	N	Video for Linux

Table 3-41: File System

Option	Value	Description
CONFIG_QUOTA	N	Quota support
CONFIG_AUTOFS_FS	N	Kernel automounter support
CONFIG_AUTOFS4_FS	N	Kernel automounter version 4 support (also supports v3)
CONFIG_REISERFS_FS	N	Reiserfs support
CONFIG_ADFS_FS	N	ADFS file system support
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)

Table 3-41: File System (Continued)

Option	Value	Description
CONFIG_CMS_FS	N	CMS file system support (experimental)
CONFIG_EXT3_FS	N	Ext3 journalling file system support (experimental)
CONFIG_FAT_FS	N	DOS FAT fs support
CONFIG_EFS_FS	N	EFS file system support (read only) (experimental)
CONFIG_JFFS_FS	Y	Journalling Flash File System (JFFS) support
CONFIG_JFFS_FS_VERBOSE	0	JFFS debugging verbosity (0 = quiet, 3 = noisy)
CONFIG_JFFS_PROC_FS	N	JFFS stats available in /proc filesystem
CONFIG_JFFS2_FS	N	Journalling Flash File System v2 (JFFS2) support
CONFIG_CRAMFS	N	Compressed ROM file system support
CONFIG_TMPFS	N	Virtual memory file system support (former shm fs)
CONFIG_RAMFS	N	Simple RAM-based file system support
CONFIG_ISO9660_FS	N	ISO 9660 CDROM file system support
CONFIG_MINIX_FS	N	Minix fs support
CONFIG_FREEVXFS_FS	N	FreeVxFS file system support (VERITAS VxFS® 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	/proc file system support
CONFIG_DEVFS_FS	N	/dev file system support (experimental)

Table 3-41: File System (Continued)

Option	Value	Description
CONFIG_DEVPTS_FS	Y	/dev/pts file system for Unix98 PTYs
CONFIG_QNX4FS_FS	N	QNX4 file system support (read only) (experimental)
CONFIG_ROMFS_FS	N	ROM file system support
CONFIG_EXT2_FS	Y	Second extended fs 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 3-42: Network File Systems

Option	Value	Description
CONFIG_CODA_FS	N	Coda file system support (advanced network fs)
CONFIG_INTERMEZZO_FS	N	InterMezzo file system support (experimental, replicating fs)
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 3-43: Partition Types

Option	Value	Description
CONFIG_PARTITION_ADVANCED	N	Advanced partition selection

Table 3-44: Sound

Option	Value	Description
CONFIG_SOUND	N	Sound support

Table 3-45: Multimedia Capabilities Port Drivers

Option	Value	Description
CONFIG_MCP	N	Multimedia drivers

Table 3-46: USB Support

Option	Value	Description
CONFIG_USB	N	Support for USB

Table 3-47: Bluetooth Support

Option	Value	Description
CONFIG_BLUEZ	N	Bluetooth subsystem support

Table 3-48: 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
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 3-49: Modular Advanced Power Management

Option	Value	Description
CONFIG_BLUECAT_APM	N	MAPM support

Table 3-50: Messenger Support

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

CHAPTER 4 *Supported Demo Systems*

This chapter provides information about BlueCat Linux demo systems supported by the `gbe_hba` BSP.

Demo Systems

Table 4-1 lists the demo systems supported in the `gbe_hba` BSP distribution, the boot devices supported by each demo system, and their respective RAM and ROM requirements.

Table 4-1: Demo Systems Supported by `gbe_hba` BSP

Demo	Boot Devices Supported by Default	ROM Requirements	RAM Requirements
<code>developer</code>	Flash (using the OS Loader) Network (using the OS Loader) Network (using RedBoot)	3741 KB	20480 KB
<code>osloader</code>	Flash (using the OS Loader) Network (using the OS Loader) Network (using RedBoot)	945 KB	9216 KB
<code>showcase</code>	Flash (using the OS Loader) Network (using the OS Loader) Network (using RedBoot)	2869 KB	18944 KB

developer Demo System

The `developer` demo system is a package consisting of the functionalities of the `shell`, `ftp`, `ping`, `gdb`, and `v1_demo` systems. For a description of `developer`

and its component demo systems, refer to Chapter 4 of the *BlueCat Linux User's Guide*.

osloader Demo System

`osloader` is the BlueCat OS Loader system 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.

Using Selected RPM Packages

This section provides a description on how to use selected RPM packages that are frequently deployed in the embedded systems environment.

Using BusyBox RPM Package

The BusyBox RPM package combines tiny versions of many common UNIX utilities into a single small executable. It provides minimalist replacements for most UNIX utilities, such as `fileutils`, `shellutils`, `findutils`, `textutils`, `grep`, `gzip`, and `tar`. 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 cousins. The options that are included, however, provide the expected functionality and behave very much like their GNU counterparts.

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

Creating a BlueCat Linux System for BusyBox

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

1. Create a new directory:

```
BlueCat:$ mkdir -p \  
$BLUECAT_PREFIX/demo/busybox/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/busybox` directory. For instance, type the following commands:

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

```
BlueCat:$ make xconfig
```

Select **save** and **Exit** to update the `.config` file, then type the following command:

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

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

3. Create the BlueCat kernel downloadable image (`busybox.kernel`):

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

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

4. Create a `.spec` file (`busybox.spec`) that contains 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
```

```
ln -s ${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
cp ${BLUECAT_PREFIX}/lib/libnss_files-*.so /lib
# 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 to produce the BusyBox kernel and RFS images. The user must change the line `KDI_NAME = developer` to `KDI_NAME = busybox` and then run the `make all` command.

Booting the BusyBox Images from a Network

To boot the BlueCat Linux with the BusyBox utility from a network using the BlueCat Linux OS Loader, perform the steps listed below. 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 development_host_IP
> set KERNEL tftp busybox.kernel
> set RFS tftp busybox.rfs
> set CMD ramdisk_size=28472
> boot
```

where *target_board_IP* is the IP address of the target and *development_host_IP* is the IP address of the development host.

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

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          7836 Mar 20  2003 /sbin/reboot
/ # chmod 755 /sbin/reboot
```

```
/ # ls -la /sbin/reboot
-rwxr-xr-x  1 0      0          7836 Mar 20  2003 /sbin/reboot
```

- echo

```
/ # echo !!!!!!!!!!
!!!!!!!!!!
```
- date

```
/ # date
Thu Jan  1 00:10:33 UTC 1970
```
- uname

```
/ # uname -a
Linux (none) 2.4.10-1 #35 Thu Mar 20 14:03:40 MSK 2003 armv5l unknown
```
- mount

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

```
/ # ifconfig eth0 192.168.4.11
#e1000: eth0 NIC Link is Up 100 Mbps Half Duplex

/ # ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:08:A2:00:02:BE
          inet addr:192.168.4.11  Bcast:192.168.4.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4 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:1080 (1.0 kb)  TX bytes:0 (0.0 b)
          Interrupt:27 Base address:0x400 Memory:40000000-40020000

/ # ping 192.168.4.121
PING 192.168.4.121 (192.168.4.121): 56 data bytes
64 bytes from 192.168.4.121: icmp_seq=0 ttl=255 time=1.3 ms
64 bytes from 192.168.4.121: icmp_seq=1 ttl=255 time=0.2 ms
64 bytes from 192.168.4.121: icmp_seq=2 ttl=255 time=0.3 ms
64 bytes from 192.168.4.121: icmp_seq=3 ttl=255 time=0.2 ms

--- 192.168.4.121 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.5/1.3 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 TinyLogin 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. For instance, type the following commands:

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

Select **Save** and **Exit** to update the `.config` file, then type the following command:

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

NOTE: The kernel configuration file for the developer demo system (\$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 `.spec` 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/tty1

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 symlinks point to /usr/bin/tinylogin. This allows the user to change the user's password simply by 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

l: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 to produce the TinyLogin kernel and RFS images. The user must change the line `KDI_NAME = developer` to `KDI_NAME = tinylogin` and then run the `make all` command.

Booting the TinyLogin Images from a Network

Use the following procedure to boot the 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 development_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 *development_host_IP* is the IP address of the development host.

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

Using TinyLogin Utility

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 the 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 the root permissions:

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

bash-2.04#
```

Using Zebra RPM Package

GNU Zebra is free software that manages a TCP/IP-based routing protocol. It takes multiserver and multithread approach to resolve the current complexity of the Internet.

GNU Zebra supports BGP4, BGP4+, OSPFv2, OSPFv3, RIPv1, RIPv2, and RIPng.

GNU Zebra is intended to be used as a Route Server and a Route Reflector. It is not a toolkit; it provides full routing power under a new architecture. GNU Zebra is unique in design in that it has a process for each protocol.

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

Creating a BlueCat Linux System for Zebra

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

1. Create a new directory:

```
BlueCat:$ mkdir -p $BLUECAT_PREFIX/demo/zebra/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/zebra directory. For instance, type the following commands:

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

```
BlueCat:$ make xconfig
```

Select **save** and **Exit** to update the `.config` file, then type the following command:

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

NOTE: In the kernel configuration the following options must be set to Y:

```
CONFIG_NETLINK=Y
CONFIG_RTNETLINK=Y
```

By default Zebra is configured to communicate with the kernel via the netlink socket.

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

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

4. Create a .spec file (zebra.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
# Standard 16550 serial driver device
mknod /dev/ttyS0 c 4 64
mknod /dev/ttyS1 c 4 65

mkdir -p /lib/security
mkdir -p /usr/lib
mkdir /bin
mkdir /sbin
mkdir -p /etc/rc.d
mkdir -p /etc/pam.d
mkdir -p /etc/xinetd.d
mkdir -p /etc/zebra
mkdir /proc
mkdir /tmp
mkdir -p /usr/bin
mkdir -p /usr/sbin
mkdir -p /var/run
mkdir -p /usr/libexec

mkdir -p /var/log/zebra

mkdir /root

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

chmod 0666 /dev/ptmx

cp ./local/fstab ./local/passwd ./local/inittab ./local/mtab /etc
cp ./local/other /etc/pam.d
cp ./local/rc.sysinit /etc/rc.d
cp ./local/hosts /etc
```

```

cp ./local/protocols /etc
cp ./local/resolv.conf /etc
cp ${BLUECAT_PREFIX}/etc/pwdb.conf /etc
cp ${BLUECAT_PREFIX}/etc/nsswitch.conf /etc
cp ${BLUECAT_PREFIX}/etc/services /etc

cp ${BLUECAT_PREFIX}/etc/security /etc

cp ./local/shadow /etc
cp ./local/pam.d /etc
cp ./local/xinetd.d/* /etc/xinetd.d
cp ./local/zebra.conf /etc/zebra/

cp ${BLUECAT_PREFIX}/lib/libnss_files-*.so /lib
cp ${BLUECAT_PREFIX}/lib/libnss_dns-*.so /lib
cp ${BLUECAT_PREFIX}/lib/libpwdb.so /lib
cp ${BLUECAT_PREFIX}/lib/security /lib

cp ./local/empty /var/log/wtmp

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

cp ${BLUECAT_PREFIX}/lib/security/pam_permit.so /lib/security

cp ${BLUECAT_PREFIX}/etc/xinetd.conf /etc

cp ${BLUECAT_PREFIX}/usr/bin/telnet /usr/bin

cp ${BLUECAT_PREFIX}/etc/shells /etc
chmod 644 /etc/shells

cp ${BLUECAT_PREFIX}/etc/group /etc

#
# General Binaries
#
lcd ${BLUECAT_PREFIX}/bin
cp ping mount bash cat ls hostname ps /bin
cp login /bin
ln -s /bin/bash /bin/sh

cp ${BLUECAT_PREFIX}/usr/bin/vtysd /usr/bin

# internet services utils
cp ${BLUECAT_PREFIX}/usr/sbin/xinetd /usr/sbin
cp ${BLUECAT_PREFIX}/usr/sbin/in.telnetd /usr/sbin
cp ${BLUECAT_PREFIX}/usr/sbin/zebra /usr/sbin

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

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

# End of File

```

5. Create the local/inittab file with the following contents:

```

id:1:initdefault:

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

```

```
l0:0:wait:/sbin/halt
l6:6:wait:/sbin/reboot

ca::ctrlaltdel:/sbin/shutdown -t3 -r now

pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting
Down"

pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown
Cancelled"

l:12345:respawn:/sbin/mingetty ttyl
```

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

```
#!/bin/sh

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

mount -a
xinetd -stayalive -reuse

hostname myhostname

zebra -d
```

7. Create the `local/zebra.conf` file of the following contents:

```
!
! zebra configuration file
!
hostname Router
password zebra
enable password zebra
!
! Interface's description.
!
interface lo
ip address 127.0.0.1/8
interface eth0
ip address 192.168.4.11/24
!
! Static default route.
!
ip route 207.21.185.0 255.255.255.0 192.168.4.254
log stdout
```

NOTE: This configuration file sets the Zebra password to `zebra`. The user has to enter this password any time when connecting to Zebra or changing the Zebra configuration mode by entering the `enable` command at the command prompt.

8. Copy the `fstab`, `passwd`, `mtab`, `other`, `hosts`, `protocols`, `resolv.conf`, `shadow`, `pam.d/*`, `xinetd.d/*`, and empty files from the `$BLUECAT_PREFIX/demo/developer/local` directory to the `$BLUECAT_PREFIX/demo/zebra/local` directory.

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

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

NOTE: Makefile for the developer demo system can be used to produce the Zebra kernel and RFS images. The user must change the line `KDI_NAME = developer` to `KDI_NAME = zebra` and then run the `make all` command.

Booting the Zebra Images from a Network

Use the following procedure to boot the BlueCat Linux with the Zebra 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 BlueCat Linux OS Loader.

At the OS Loader prompt, type the following commands:

```
> set IF eth0
> set IP target_board_IP
> set HOST development_host_IP
> set CMD ramdisk_size=8192
> set KERNEL tftp zebra.kernel
> set RFS tftp zebra.rfs
> boot
```

where `target_board_IP` is the IP address of the target and `development_host_IP` is the IP address of the development host.

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

Using Zebra Utility

This section provides an examples of using the Zebra utility:

```
myhostname login: root
bash-2.04# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:08:A2:00:02:BE
          inet addr:192.168.4.11  Bcast:192.168.4.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2  errors:0  dropped:0  overruns:0  frame:0
          TX packets:0  errors:0  dropped:0  overruns:0  carrier:0
```

```
collisions:0 txqueuelen:100
Interrupt:27 Base address:0x400 Memory:40000000-40020000

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

bash-2.04# ping -c 2 192.168.4.104
PING 192.168.4.104 (192.168.4.104) from 192.168.4.11 : 56(84) bytes of
data.
64 bytes from 192.168.4.104: icmp_seq=0 ttl=255 time=59 usec
Warning: time of day goes back, taking countermeasures.
64 bytes from 192.168.4.104: icmp_seq=1 ttl=255 time=599 usec

--- 192.168.4.104 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.059/0.329/0.599/0.270 ms
bash-2.04# ping -c 3 207.21.185.6
PING 207.21.185.6 (207.21.185.6) from 192.168.4.11 : 56(84) bytes of data.
64 bytes from 207.21.185.6: icmp_seq=0 ttl=254 time=2 usec
Warning: time of day goes back, taking countermeasures.
64 bytes from 207.21.185.6: icmp_seq=1 ttl=254 time=737 usec
64 bytes from 207.21.185.6: icmp_seq=2 ttl=254 time=481 usec

--- 207.21.185.6 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.002/0.406/0.737/0.305 ms
bash-2.04# telnet localhost 2601
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is zebra (version 0.91a).
Copyright 1996-2001 Kunihiro Ishiguro.

User Access Verification

Password: zebra
Router> enable
Password: zebra
Router# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       B - BGP, > - selected route, * - FIB route

C>* 127.0.0.0/8 is directly connected, lo
C>* 192.168.4.0/24 is directly connected, eth0
S>* 207.21.185.0/24 [1/0] via 192.168.4.254, eth0
Router#
```

Table 5-1 lists the device drivers supported by the gbe_hba BSP.

Table 5-1: Device Drivers Supported by the gbe_hba BSP

Hardware Device	Device Drivers	Location in Source Tree	Kernel Configuration Options	Notes
UART, implemented in HBACC Xilinx	serial.c	drivers/char	CONFIG_SERIAL CONFIG_SERIAL_CONSOLE	
Ethernet, Intel 82545/82546 Gigabit adapters	*.c *.h	drivers/net/e1000	CONFIG_E1000	
Flash, 8 MB	arm_flash0.c	drivers/mtd/maps	CONFIG_MTD_GBE_HBA CONFIG_MTD_GBE_HBA_PART	Supported via JFFS and an MTD driver

This chapter describes the BlueCat Linux API specific to the Intel 80200 processor.

Advanced MMU/Cache Management Services

The BlueCat Linux kernel for GbE HBA provides a set of advanced MMU/cache management services for the Intel XScale 80200 processor. These are declared in the `$BLUECAT_PREFIX/usr/src/linux/include/asm/i80200.h` file.

Advanced MMU/Cache Management Services Overview

The purpose of the advanced MMU/cache management services is to provide an access to Intel i80200 cache control capabilities. The kernel components, mostly device drivers, can use these services to greatly increase performance of a selected piece of critical code. Locking the MMU/cache for a particular purpose effectively allocates the processor execution resources for a particular task at the expense of the other tasks in the system. What follows are the most common examples of how the MMU/cache management can solve various problems:

- A device driver can lock its interrupt handler code into the instruction cache to provide more prompt hardware interrupt response. In some cases (e.g., a device driver for a sound device that provides audio data to be played), this might be the only way to be able to meet a hardware device events response requirements.
- If a driver implements mission critical calculations to be completed as fast as possible, locking the code and data into the respective caches combined with the interrupts/scheduling disabled can provide an execution environment that fully utilizes the processor, thus effectively making the code run as fast as possible on this hardware.

- Advanced MMU/cache management provides means for general tasks targeting the overall optimization of the Linux kernel performance and real-time characteristics.

Advanced MMU/Cache Management Services Reference

What follows is the list of services available for the kernel components:

- `void i80200_dcache_lock(unsigned int addr, int lines)`
Lock a number of lines at the specified virtual address into the data cache.

This service can be used to lock a data area into the data cache. This will ensure the accesses to the locked area are cached. The contents of the area are not pushed out of the cache by accessing other addresses.
- `void i80200_icache_lock(unsigned int addr, int lines)`
Lock a number of lines at the specified virtual address into the instruction cache.

This service can be used to lock a piece of code into the instruction cache. This will ensure execution from the instruction cache. The contents of the area are not pushed out of the cache by execution code from other addresses.
- `void i80200_dcache_unlock(void)`
Unlocks the data cache.

This service invalidates all locks on the data cache, allowing the cache to operate in normal mode with hardware line replacement algorithm. The service can be used to restore the data cache normal operating mode when the advanced cache configuration is not needed anymore.
- `void i80200_icache_unlock(void)`
Unlocks the instruction cache.

This service invalidates all locks on the instruction cache, allowing the cache to operate in normal mode with hardware line replacement algorithm. The service can be used to restore the instruction cache normal operating mode when the advanced cache configuration is not needed anymore.
- `void i80200_itlb_lock(unsigned int addr)`
Translates and locks an ITLB entry.

The locked ITLB entry will ensure that the instruction fetch from the page at the specified virtual address will go straight through the ITLB translation mechanism without overhead for the page table tablewalk.

- `void i80200_dtlb_lock(unsigned int addr)`

Translates and locks an DTLB entry.

The locked DTLB entry will ensure that the accesses to the page at the specified virtual address will go straight through the DTLB translation mechanism without overhead for the page table tablewalk.

- `void i80200_itlb_unlock(void)`

Unlocks the ITLB.

This service invalidates all locks on the instruction TLB, allowing the TLB to operate in normal mode with hardware entry replacement algorithm. The service can be used to restore the ITLB normal operating mode when the advanced TLB configuration is not needed anymore.

- `void i80200_dtlb_unlock(void)`

Unlocks the DTLB.

This service invalidates all locks on the data TLB, allowing the TLB to operate in normal mode with hardware entry replacement algorithm. The service can be used to restore the DTLB normal operating mode when the advanced TLB configuration is not needed anymore.

- `void i80200_md_attr(unsigned int attr)`

Sets attributes for mini data cache.

`attr` defines mini data cache attributes as specified in the Auxiliary Control Register description in the Processor Manual. `attr` can be one of the following:

0 - Write back, Read allocate

1 - Write back, Read/Write allocate

2 - Write through, Read allocate

3 - Unpredictable.

- `void i80200_wb_coalescing(int enable)`

Enables or disables write coalescing.

This service allows the global disabling of write coalescing. All writes to an external memory will be performed in the program order without write

combining operations in the write buffer, regardless the values of the page attributes.

GbE HBA Board Problems and Limitations

The following are known problems and limitations of this release:

- Kernel debugger (KDBG) is supported but has not been explicitly tested. This is due to a limitation of the GbE HBA hardware, which provides a single serial port only. Since debugging of BlueCat over a serial line requires a dedicated serial port on the target, testing of the kernel debugger on the GbE HBA is not possible.
- For the same reason, debugging of user applications over a serial line is supported but not explicitly tested.
- Use the following command in order to use Ethernet in the `i_osloader` demo system:

```
make -f Makefile.i xconfig
```

Enable your network card, then type the following command:

```
make -f Makefile.i all
```

The demo now will have a correct Ethernet configuration and can be used to boot other BlueCat Linux demos over the network.

- If `mkrootfs` is terminated (either by an error or by a signal), it tries to clean all its temporary files before exiting. Due to certain features of the Cygwin execution environment, however, such temporary files can remain uncleaned in the `/tmp` directory on a Windows host. It is recommended that the `/tmp` directory be regularly checked and cleaned.
- Debugging of multithreaded applications via GDB is not supported.
- The `tc1x` 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.