

Intel[®] IXP400 Software: RedBoot* v2.04 Software

Software Release Notes

March 29, 2007



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Intel Corporation may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor_number for details.

The Intel® IXP400 Software may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

MPEG is an international standard for video compression/decompression promoted by ISO. Implementations of MPEG CODECs, or MPEG enabled platforms may require licenses from various entities, including Intel Corporation.

This Software Release Notes as well as the software described in it is furnished under license and may only be used or copied in accordance with the terms of the license. The information in this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Intel Corporation. Intel Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document.

Except as permitted by such license, no part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the express written consent of Intel Corporation.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature may be obtained by calling 1-800-548-4725 or by visiting Intel's website at <http://www.intel.com>.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino logo, Core Inside, FlashFile, i960, InstantIP, Intel, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel Core, Intel Inside, Intel Inside logo, Intel Leap ahead., Intel Leap ahead. logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skool, Sound Mark, The Journey Inside, VTune, Xeon, and Xeon Inside are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2007, Intel Corporation. All Rights Reserved.



Contents

1.0	Introduction	4
1.1	What's New in This Release	4
1.2	Support	4
1.3	Related Documents	4
1.4	RedBoot* Board Support Information Online	5
2.0	Obtaining the Software and NPE Support Components	6
3.0	Using the Certified Binaries	6
4.0	Programming RedBoot* into Flash Memory	10
4.1	Using VisionICE*	11
4.2	Using Macraigor* Raven*	11
4.3	Using Abatron* BDI2000*	11
5.0	Verifying RedBoot* Operation	12
6.0	Using RedBoot* to Update RedBoot	14
7.0	Setting Up and Configuring RedBoot*	14
7.1	Configuration	14
7.1.1	Initializing the Flash Image System (FIS)	15
7.1.2	Initializing the RedBoot* Configuration	15
7.2	NPE Ethernet Support	16
7.2.1	Setting Default Ethernet Port	17
7.2.2	NPE-A: UTOPIA vs. Ethernet	17
7.2.3	Setting the MAC Address	17
8.0	Booting Linux*	18
8.1	Loading the Linux* Kernel	18
8.2	Booting Linux From An IDE Hard Disk Drive	19
8.3	Endianness and Booting Linux*	19
8.4	Deployment Notes	19
9.0	Building Binary Images From Source	20
9.1	Preparing the Host Platform	20
9.2	Obtaining the Source	20
9.3	Installation and Setup of the Toolchain	21
9.4	Set Up the RedBoot* Source Tree	21
9.5	Build the RedBoot* Images	23
9.6	Configuring the RedBoot* Build for Other Targets	23
9.6.1	Setting the Target Platform	23
9.6.2	Selecting the eCos Configuration	24
10.0	Additional Features	24
10.1	Cache Policy	24
10.2	GPIO Mappings	25
10.3	PCI Enumeration	25



1.0 Introduction

This document provides information for installing and using Intel® IXP400 Software RedBoot* Boot-Loader v2.04 Software with NPE support on the following platforms:

- Intel® IXDP425 / IXCDP1100 Development Platform
- Intel® IXDP465 Development Platform
- Intel® IXDPG425 Network Gateway Development Platform
- Intel® IXP435 Multi-Service Residential Gateway Reference Platform
- ADI Engineering's Coyote* Gateway Reference Design (GRG)

Note: The Ethernet NPE support source code for Red Hat* RedBoot is not available in its public CVS repository. The NPE microcode image are provided by Intel to add this NPE capability to RedBoot.

1.1 What's New in This Release

The main new feature of this release are add support for

- Intel® IXP435 multi-service residential gateway reference platform
- B1-stepping of Intel® IXP42X product line
- Support E1000 card on Intel® IXP435 multi-service residential gateway reference platform. PCI NIC card that has been validated is Intel® Pro/1000 GT PCI NIC card.

1.2 Support

If you have any support issues, please contact your local Intel support representative. For general RedBoot support issues regarding the public RedBoot sources, contact Red Hat.

1.3 Related Documents

Document	Document Number
<i>Intel® IXP400 Software Programmer's Guide</i>	252539
<i>Intel XScale® Core Developer's Manual</i>	273473
<i>Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Datasheet</i>	252479
<i>Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Developer's Manual</i>	252480
<i>Intel® IXP4XX Product Line of Network Processors Specification Update</i>	252702
<i>Intel® IXP45X and Intel® IXP46X Product Line of Network Processors Developer's Manual</i>	306262
<i>Intel® IXP43X Product Line of Network Processors Developer's Manual</i>	316843
<i>Intel® IXDP425 Development Platform User's Guide</i>	273743



Document	Document Number
Intel® IXDP465 Development Platform User's Guide	306462
Intel® IXP435 Multi-Service Residential Gateway Reference Platform User's Guide	316848
Intel® IXDPG425 Gateway Reference Platform User's Guide	303146

Most documents are available online at:

<http://www.intel.com/design/network/products/npfamily/docs/ixp4xx.htm>

The software release documents are available at:

http://developer.intel.com/design/network/products/npfamily/download_ixp400.htm

The RedBoot home page is online at:

<http://ecos.sourceware.org/redboot/>

The RedBoot User's Guide is available online at:

<http://sources.redhat.com/ecos/docs-latest/redboot/redboot-guide.html>

For the latest eCos documentation, consult the online reference at:

<http://sources.redhat.com/ecos/docs-latest/>

Specific information is provided in the source code distribution in the directory:

docs\redboot

The following book, a guide on developing eCos applications by Anthony J. Massa provides an overview of RedBoot:

Massa, Anthony J., *Embedded Software Development with eCos*, ISBN 0-13-035473-2 Prentice Hall, New Jersey, 2003.

1.4 RedBoot* Board Support Information Online

- Intel® IXDP425 Development Platform:

<http://sources.redhat.com/ecos/boards/ixdp425.html>

- ADI Engineering's Coyote* Gateway Reference Design

<http://sources.redhat.com/ecos/boards/grg.html>



2.0 Obtaining the Software and NPE Support Components

The Intel® IXP400 Software RedBoot* Boot-Loader v2.04 Software and associated components are available at the following Intel website:

http://developer.intel.com/design/network/products/npfamily/download_ixp400.htm

This website has links to obtain the following components:

- Precompiled Red Hat-certified binaries v2.04 with NPE support — `npe_bin-070320.tar.gz`
- Microcode for the NPE support — `Redboot-v2_04-npe-microcode.zip`
- Links (URLs) to the Red Hat FTP site where the source code and toolchain for building RedBoot can be obtained.

Note: To modify the function of RedBoot or to reproduce existing binaries, download the NPE support component files from the Intel website and obtain the source code and toolchain from Red Hat.

3.0 Using the Certified Binaries

The quickest way to take advantage of NPE-enabled RedBoot is to use the Red Hat-certified binaries. This allows immediate use of the NPE MII/Ethernet interfaces.

There are multiple image configurations and file types provided. The configuration is encoded in the name of the image. The files formats are binary (.bin), ELF (.elf), S-record (srec), and image (.img).

- The files with "LE" appended to the name initialize the processor for little-endian mode operation.

Note: RedBoot little-endian is validated on IXDP465 platform and IXDP425 platform

- RAM - The image must be loaded into and runs from RAM
- ROM - the image runs from flash with a minimal RAM footprint
- ROMRAM - the image initially runs from flash but copies itself into and executes from RAM

Unzip the binaries using the following command:

```
$ tar xvfz npe_bins-070320.tar.gz
```

This creates the directory `redboot-bins` with the following contents:



```
npe_bin_070320/  
|-- grga  
| |-- redboot_RAM.bin  
| |-- redboot_RAM.elf  
| |-- redboot_RAM.img  
| |-- redboot_RAM.srec  
| |-- redboot_ROM.bin  
| |-- redboot_ROM.elf  
| |-- redboot_ROM.img  
| |-- redboot_ROM.srec  
| |-- redboot_RAMLE.bin  
| |-- redboot_RAMLE.elf  
| |-- redboot_RAMLE.img  
| |-- redboot_RAMLE.srec  
| |-- redboot_ROMLE.bin  
| |-- redboot_ROMLE.elf  
| |-- redboot_ROMLE.img  
| `-- redboot_ROMLE.srec
```



```
|-- ixdp465
|  |-- redboot_RAM.bin
|  |-- redboot_RAM.elf
|  |-- redboot_RAM.img
|  |-- redboot_RAM.srec
|  |-- redboot_RAMLE.bin
|  |-- redboot_RAMLE.elf
|  |-- redboot_RAMLE.img
|  |-- redboot_RAMLE.srec
|  |-- redboot_ROM.bin
|  |-- redboot_ROM.elf
|  |-- redboot_ROM.img
|  |-- redboot_ROM.srec
|  |-- redboot_ROMLE.bin
|  |-- redboot_ROMLE.elf
|  |-- redboot_ROMLE.img
|  |-- redboot_ROMLE.srec
|  |-- redboot_ROMRAM.bin
|  |-- redboot_ROMRAM.elf
|  |-- redboot_ROMRAM.img
|  |-- redboot_ROMRAM.srec
|  |-- redboot_ROMRAMLE.bin
|  |-- redboot_ROMRAMLE.elf
|  |-- redboot_ROMRAMLE.img
|  |-- redboot_ROMRAMLE.srec
```



```
|-- ixdp425
|
|  |-- redboot_RAM.bin
|
|  |-- redboot_RAM.elf
|
|  |-- redboot_RAM.img
|
|  |-- redboot_RAM.srec
|
|  |-- redboot_RAMLE.bin
|
|  |-- redboot_RAMLE.elf
|
|  |-- redboot_RAMLE.img
|
|  |-- redboot_RAMLE.srec
|
|  |-- redboot_ROM.bin
|
|  |-- redboot_ROM.elf
|
|  |-- redboot_ROM.img
|
|  |-- redboot_ROM.srec
|
|  |-- redboot_ROMLE.bin
|
|  |-- redboot_ROMLE.elf
|
|  |-- redboot_ROMLE.img
|
|  |-- redboot_ROMLE.srec
|
|  |-- redboot_ROMRAM.bin
|
|  |-- redboot_ROMRAM.elf
|
|  |-- redboot_ROMRAM.img
|
|  |-- redboot_ROMRAM.srec
|
|  |-- redboot_ROMRAMLE.bin
|
|  |-- redboot_ROMRAMLE.elf
|
|  |-- redboot_ROMRAMLE.img
|
|  `-- redboot_ROMRAMLE.srec
```



```
'-- ixdpg425
  |-- redboot_RAM.bin
  |-- redboot_RAM.elf
  |-- redboot_RAM.img
  |-- redboot_RAM.srec
  |-- redboot_ROM.bin
  |-- redboot_ROM.elf
  |-- redboot_ROM.img
  '-- redboot_ROM.srec
|-- kixrp435
|  |-- redboot_RAM.bin
|  |-- redboot_RAM.elf
|  |-- redboot_RAM.img
|  |-- redboot_RAM.srec
|  |-- redboot_ROM.bin
|  |-- redboot_ROM.elf
|  |-- redboot_ROM.img
|  |-- redboot_ROM.srec
|  |-- redboot_ROMRAM.bin
|  |-- redboot_ROMRAM.elf
|  |-- redboot_ROMRAM.img
|  |-- redboot_ROMRAM.srec
```

- a. 'grg' is the original acronym for ADI Engineering's Coyote* Gateway Reference Design

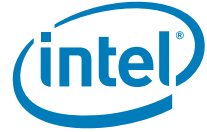
For further instructions on using the binary images, see [Section 4.0](#).

4.0 Programming RedBoot* into Flash Memory

The following instructions describe how to install the ROM image into flash.

Note:

Due to the IXP4XX product line processor memory configuration, it is necessary to byte-swap the target image if a Big-Endian image is used. This can be done by most stand-alone flash programmers or by some JTAG-based flash programmers. The VisionICE* JTAG tool can perform the byte swapping. Most JTAG tools such as JFLASH software and the BDI2000 do the byte swapping automatically when in BE mode.



Note: For the Intel® IXDP465 Development Platform, please refer to the *Intel® IXDP465 Development Platform User's Guide* (306462) for specific details related to installing/upgrading RedBoot.

4.1 Using VisionICE*

1. Invoke the VisionICE* software and connect to the target platform.
2. Using the command series **Tools > Convert object modules**, convert the s-record file to a Flat binary for flash programming by using the settings below and click **Convert**.
3. When complete, an output window is displayed.
4. To continue, press any key.
5. Using the command series **Tools > Program flash devices**, program the binary image to flash by clicking on **Erase** and **Program**.
6. When complete, an output window will be displayed
7. To continue, click **OK**.

4.2 Using Macraigor* Raven*

1. If a Big-Endian RedBoot image is used, byte-swap the image using `swapbytes.exe` (provided with the Macraigor* tool).
2. Convert the RedBoot binary file (or the file created in step 1) to s19 format using `BinToS19.exe` (provided with the Macraigor tool).
3. Use the default IXP4XX development platforms configuration.
4. Set the CPU Endian Configuration to "LITTLE ENDIAN"
5. Program the flash using the S19 file made in Step 2.

4.3 Using Abatron* BDI2000*

1. Run a TFTP server on your host and copy `redboot.bin` built in [Section 9.6](#) to your default TFTP download directory along with the BDI2000 configuration files for your hardware.
2. Use Telnet to connect to the bdi2000 and program the flash as follows:

```
$ telnet bdi2000
```
3. You should see the following prompt:



```
Trying 10.243.17.245...
Connected to bdi2000.
Escape character is '^]'.
BDI Debugger for XScale (030918)
=====
Core#0> erase
Erasing flash at 0x50000000
Erasing flash at 0x50020000
Erasing flash at 0x50040000
Erasing flash at 0x50060000
Erasing flash at 0x50080000
Erasing flash passed
Core#0> prog redboot.bin BIN
Programming redboot.bin , please wait ....
Programming flash passed
Core#0> verify
Verifying redboot.bin , please wait ....
Verifying target memory passed
Core#0>
```

5.0 Verifying RedBoot* Operation

To verify RedBoot operation requires that you have access to the RedBoot command line or the console. This is accomplished by connecting the platform console UART (typically UART0) to a host computer running a serial communication program like minicom (Linux*) or HyperTerminal* (Windows*). The settings for the host computer serial console are:

115200 baud, 8-databits, No parity, 1-stop bit and no flow control.

Note: Additional information regarding board support is provide with the source code distribution. Refer to [Section 1.3](#)

Power-off, connect the host and then power-on the platform. RedBoot will display announcements and boot-status messages similar to the following:



```

RedBoot> +...waiting for BOOTP information

Ethernet eth0: MAC address 00:0e:0c:63:cd:cf
IP:192.168.200.100/255.255.255.0, Gateway: 192.168.200.254
Default server: 192.168.200.254

RedBoot(tm) bootstrap and debug environment [ROM]
Red Hat certified release, version 2.04 - built 15:42:38, Mar 21, 2007

Platform: IXDP465 Development Platform (IXP465 533) BE
Copyright (C) 2000, 2001, 2002, 2003, 2004, 2007 Free Software Foundation, Inc.

RAM: 0x00000000-0x10000000, 0x0001f178-0x0fffd1000 available
FLASH: 0x50000000 - 0x51000000, 128 blocks of 0x00020000 bytes each.

RedBoot>

```

After a reset or power-cycle, the “+” character will appear within two seconds after power-on on the console. The boot-status/banner messages will display in about five seconds. The Boot-status/banner messages may be delayed by up to 30 seconds. This is typically due to the network configuration of the host or RedBoot. RedBoot attempts to initialize and obtain an IP address for the default network port and the time out for this is 30 seconds. If an IP address is obtained, the delay is about two to five seconds.

If this is the initial installation of RedBoot, the following warning may be displayed:

```
FLASH configuration checksum error or invalid key
```

This is normal and indicates that the flash configuration must be initialized for use. Refer to [Section 7.1](#) for additional information.

If the IP address assignment fails, the following message is displayed:

```
can't get BOOTP info for device!
```

The announcements, version number and “built” time may be different, depending on the source and when the build was done. The binaries provided by Intel are built and certified by Red Hat, and as such they report the certification message and the version number, similar to the following:

```

RedBoot(tm) bootstrap and debug environment [ROM]
Red Hat certified release, version 2.04 - built 15:42:38, Mar 21 2007

```



6.0 Using RedBoot* to Update RedBoot

RedBoot is capable of updating itself. The following steps generally apply to any platform, but because there may be special requirements, please refer to the appropriate platform's *User Guide* for the required hardware setup.

RedBoot must be executing from RAM in order to update the existing RedBoot image in flash. In the steps shown, a RAM-based RedBoot image is loaded and run. The RAM-based RedBoot is used to load the ROM image into RAM for storage to flash. Both `redboot_ROM.srec` and `redboot_RAM.srec` are on the tftp server for loading via network connection.

```
load -v redboot_RAM.srec
go -w 1 0x00100040
# <wait for RedBoot to restart>
load -v redboot_ROM.srec -b %{FREEMEMLO}
fis unlock RedBoot
fis create RedBoot -b %{FREEMEMLO}
fis lock RedBoot
```

In the unlikely event that the `fis unlock RedBoot` returns a “no such file” error message, enter the command `fconfig -i` to reformat the Flash configuration.

The last address parameter to the `go` command (0x00100040) is the entry point reported by the load command.

7.0 Setting Up and Configuring RedBoot*

This section provides RedBoot configuration and NPE Ethernet configuration extensions. For additional details on setup and configuration of RedBoot, refer to Red Hat online documentation or the documentation with the source distribution.

7.1 Configuration

If booting RedBoot for the first time, the FIS and flash configuration data must be initialized. After power-cycling the board, if the `fconfig` data is missing or does not match, the following warning is displayed:

```
FLASH configuration checksum error or invalid key.
```

This indicates that the flash configuration must be initialized for use.

There are two elements to configuring RedBoot for the platform:

- Setting up the board/platform configuration
- Initializing the FIS (Flash Image System).



The Flash Configuration (accessed via `fconfig`) contains board/platform-specific non-volatile configuration settings. The FIS is a simple 'database' of the images that are placed in and managed in flash by RedBoot.

Three steps are required to configure RedBoot and initialize the flash after installing RedBoot:

1. From the RedBoot command line run the command `fis init`
2. From the RedBoot command line run the command `fconfig -i`
3. Power-cycle the board.

Note: In general, it is a good idea to reinitialize the FIS and `fconfig` when updating the RedBoot image, though it is not strictly required. If the new RedBoot image supports additional `fconfig` options, they will not appear until after the `fconfig` internal structure is updated. This is accomplished by using the `fconfig -i` command. For example, the RedBoot v2.04 update adds the UTOPIA state configuration option for the IXDP465. This option will not appear until `fconfig -i` is run.

7.1.1 Initializing the Flash Image System (FIS)

Warning: After an upgrade, the use of the `fis init` command will remove any record of non-primary RedBoot images — and their location in flash — from the FIS. This applies to the operating system or relates the images in flash (e.g. `zImage` (Linux kernel) or `ramdisk.gz` (Linux ramdisk)). The actual images are not erased and are still present in flash, but there is no record (name, size, location, etc.) of them in the FIS. To restore a FIS record, an image will need to be reprogrammed using the `fis create` command. If the flash argument “-f” is added to the command (e.g., `fis init -f`) the FIS is cleared and all blocks of flash memory are erased. The primary RedBoot image is not affected.

The following is an example of the output from the `fis init` command:

```
RedBoot> fis init
About to initialize [format] FLASH image system - continue (y/n)? y
*** Initialize FLASH Image System
... Unlock from 0x50fe0000-0x51000000: .
... Erase from 0x50fe0000-0x51000000: .
... Program from 0x0ffdf000-0x0ffff000 at 0x50fe0000: .
... Lock from 0x50fe0000-0x51000000: .
Redboot>
```

7.1.2 Initializing the RedBoot* Configuration

This can be done using the `fconfig -i` command. The command prompts with a description and the existing (default value) for the configuration item. Accept the default value by pressing **Enter**, or backspace over the value and enter the new value.

RedBoot will prompt to commit updates to flash. A 'y' entry is required or the update is abandoned and no changes are made. The platform must be power-cycled to ensure the changes are activated.



Example output from executing the command on the IXDP465 is shown below. The UTOPIA option is only relevant to the IXDP465 platform. The defaults are used and accepted.

```
RedBoot> fconfig -i

Initialize non-volatile configuration - continue (y/n)? y

Run script at boot: false

Use BOOTP for network configuration: true

Default server IP address:

Console baud rate: 115200

GDB connection port: 9000

Force console for special debug messages: false

Network debug at boot time: false

Default network device: i82559_eth0

Utopia in NPE-A: true

Update RedBoot non-volatile configuration - continue (y/n)? y

... Unlock from 0x50fe0000-0x51000000: .
... Erase from 0x50fe0000-0x51000000: .
... Program from 0x0ffd000-0x0fff000 at 0x50fe0000: .
... Lock from 0x50fe0000-0x51000000: .

RedBoot>
```

7.2 NPE Ethernet Support

The RedBoot release supports two NPE Ethernet ports (NPE-B and NPE-C) on IXP42X product line-based boards, three NPE ports (NPE-A, NPE-B, and NPE-C) on the IXP45X/IXP46X product line-based boards, and two NPE Ethernet ports (NPE-A and NPE-C) on IXP43X product line-based boards.

The IXP435 reference platform has one Ethernet switch connected to NPE-C and Ethernet WAN connected to NPE-A.

The Intel® IXDPG425 Network Gateway Development Platform and ADI Engineering's Coyote* Gateway Reference Design have a four-port Ethernet switch connected to NPE-B and the Ethernet WAN connected to NPE-C.

Note: With the exception of the Intel® IXDPG425 Network Gateway Development Platform, support for PCI i82559-based Ethernet NICs is provided.



7.2.1 Setting Default Ethernet Port

For all platforms except the Intel® IXDPG425 Network Gateway Development Platform, the default Ethernet port is the PCI NIC. If the default Ethernet device is not found (for instance, the PCI card is not installed), RedBoot will try to use one of the other ports.

The default Ethernet port can be changed using `fconfig` command. Refer to [Section 7.1](#). The default port may also be set using the following command:

```
Redboot> fconfig net_device DeviceName
```

Valid values for DeviceName are:

- i82559_eth0
- e1000_eth0
- npe_eth0 (corresponding to NPE-B)
- npe_eth1 (corresponding to NPE-C)
- npe_eth2 (corresponding to NPE-A)
- npe_lan (corresponding to NPE-C)
- npe_wan (corresponding to NPE-A)

Note: The IXP421 network processor has only `npe_eth0`; `npe_eth0` and `npe_eth1` correspond to NPE-B and NPE-C on the IXP42X product line-based platforms. `npe_eth2` is for NPE-A on the IXP45X/IXP46X product line only. `npe_lan` and `npe_wan` correspond to NPE-C and NPE-A on IXP43X product line.

7.2.2 NPE-A: UTOPIA vs. Ethernet

On the IXDP465 platform NPE-A defaults to UTOPIA mode, not Ethernet. A flash configuration option is available to control the default use of NPE-A. The `fconfig` setting `utopia` holds the state of the NPE-A configuration. For example, the command:

```
RedBoot> fconfig utopia false
```

sets the default use of NPE-A to Ethernet. A power-cycle is required for the change to take effect.

Note: When upgrading from pre-release versions of RedBoot for the IXDP465, the message "FLASH configuration checksum error or invalid key." will not be displayed. But in order to use the `fconfig utopia` setting, you must initialize the flash configuration.

7.2.3 Setting the MAC Address

RedBoot will not attempt to use a network device until it has a valid MAC address.

The IXDP425 platform, the IXP435 reference platform, the IXDPG425 network gateway platform, and the IXDP465 platform store MAC addresses for the NPE Ethernet ports in onboard I²C EEPROMs. The ADI Engineering's Coyote* Gateway Reference Design uses the main flash to store the MAC addresses.



The platforms are shipped with the MAC address set in non-volatile storage specific to the platform. All platforms are shipped with proper labels for the IEEE MAC address assigned to the port. It may be necessary to change/set the MAC address of the port before it is used.

- To set the MAC address of an NPE port on ALL boards except the Coyote board, use the following command:

```
RedBoot> set_npe_mac
```

This command accepts a `-p` switch to specify the port (for example, 0 for NPE-B, and 1 for NPE-C). The MAC address is specified using a format of `xx:xx:xx:xx:xx:xx` or `xxxxxxxxxxxx`.

For example, to set the MAC address for `npe_eth0` (NPE-B), use the following command:

```
RedBoot> set_npe_mac -p 0 00:01:AF:00:20:EC
```

- On the Coyote board, set the MAC address of an NPE port by using the following RedBoot `fconfig -i` command and then setting a new value in the field "Network hardware address [MAC] for NPE eth0".

```
RedBoot> fconfig -i
```

Note:

The IXDP465 platform must be jumper-configured to use the processor I²C unit (instead of GPIO pins) and to allow writes to EEPROM. Please refer to the *Intel® IXDP465 Development Platform User's Guide*.

8.0 Booting Linux*

This section provides additional information on booting Linux on the available IXP4XX development platforms.

8.1 Loading the Linux* Kernel

To load and execute the Linux kernel, copy the zImage to a directory on the tftp boot server and type:

```
RedBoot> load -v -r -b %{FREEMEMLO} zImage
RedBoot> exec
```

You can pass an alternative command line to the Linux kernel using the `exec` command. This is useful when the default kernel command line (set via the `.config` file) needs to be adjusted. For example, to reduce the amount of memory Linux can use on the IXDP425 platform from 256M (the default) to 64M, use the `exec` command:

```
RedBoot> exec -c "console=ttyS0,115200 root=/dev/nfs ip=bootp mem=64M@0x00000000"
```



8.2 Booting Linux From An IDE Hard Disk Drive

RedBoot supports loading images from a PCI-attached IDE HDD.

Note: The disk must be pre-formatted with an ext3 file system and the images must be loaded from a known path on the disk. The setup and initialization of the HDD is beyond the scope of the release notes.

To load an image, add the arguments `-m disk` and the path to the image to the load command. For example:

```
RedBoot> load -v -r -b {%FREEMEMLO} -m disk hda1:path/to/zImage
RedBoot> exec <params>
```

Use the `disks` command to display disk information. `hda1` is the name of the disk.

8.3 Endianness and Booting Linux*

Note: It is recommended that the endianness of RedBoot match the endianness of the Linux kernel. If this is not initially possible, RedBoot is capable of handling the load and execution of the image.

There are two modifications required when loading and starting a Linux kernel that has a different endianness than RedBoot:

- The `swab` command is used to byteswap the image before executing the image.
- the `-x` switch must be used with the `exec` command.

For example, to load and launch a big-endian zImage and ramdisk using a little-endian RedBoot, use commands such as the following:

```
RedBoot> load -r -v -b 0x01600000 zImage
Raw file loaded 0x01600000-0x0168f4eb, assumed entry at 0x01600000
RedBoot> swab -b 0x01600000 -l 0x90000
RedBoot> exec -x 0x01600000
```

8.4 Deployment Notes

The Linux kernels are configured out-of-box to support a development model using an NFS-mounted root file system. When using the NPE-enabled RedBoot in this model, be aware that the kernel *only* supports the standard EtherExpress* Pro 10/100 adapter interface when it boots because this driver is statically linked with the kernel. You can continue to use the Ethernet adapter to support the kernel NFS mount during initial development. When ready to deploy, configure the kernel to use a flash-based file system, and also configure RedBoot to use either `npe_eth0` or `npe_eth1` device as the default network device in RedBoot.



9.0 Building Binary Images From Source

The steps in this section allow you to reproduce the binary images from source and are required should you find it necessary to modify RedBoot functionality. These steps are for building RedBoot on a host machine running Linux. To install the toolchain you will need root access.

The steps to build RedBoot are:

1. Prepare the host platform. See [Section 9.1](#)
2. Download the source and toolchain. See [Section 9.2](#)
3. Prepare the toolchain. See [Section 9.3](#).
4. Set up the source code. See [Section 9.4](#).
5. Execute the build steps. See [Section 9.5](#)

Information on how to modify the specific steps required to build for a different endianness or target platform is give in [Section 9.6](#).

Note: You can rebuild RedBoot from source code obtained from the public CVS repository. Building from this repository will result in a *non-Red Hat–certified* release and the version number is displayed as “unknown.”

9.1 Preparing the Host Platform

Before starting the RedBoot build, verify that your host platform meets the general requirements used to verify the instructions.

The RedBoot build was verified on a Intel® Pentium III 550-MHz class host system, with 256 MB of RAM, running Red Hat 7.3 or Red Hat 9.0 Linux, with the “Everything” option selected during installation. You should have approximately 1000 MB of free space available before proceeding with the build steps.

9.2 Obtaining the Source

In order to build the RedBoot binaries from source you need the NPE support components, the RedBoot source code, and the toolchain. The NPE support components can be obtained from the Intel Web site.

<http://www.intel.com/design/network/products/npfamily/docs/ixp4xx.htm>

The toolchain binaries (and source) can be obtained from the Red Hat Global Engineering support FTP server. The URL is given below:

<ftp://ftp.ges.redhat.com/private/gnupro-xscale-030422>

The required files are:

- Source code: `redboot-intel-ixp4xx-070320.tar.gz`
- toolchain: `i686-pc-liinux-gnulibc2.2-x-xscale-elf.tar.Z`
- NPE microcode (if NPE support require): `Redboot-v2_04-npe-microcode.zip`



Note: Downloading the files from the Red Hat server (or external FTP sites in general) will require that you have external access to the Internet. Companies have different policies on how they protect internal networks and may block access to external FTP sites. Please consult with your network system administrator if you have difficulties accessing this site. When connecting to the Red Hat server please use an FTP client that supports Passive FTP (PASV FTP mode).

9.3 Installation and Setup of the Toolchain

1. Extract the install script, by executing the following command:

```
$ unzip i686-pc-linux-gnulibc2.2-x-xscale-elf.tar.Z | tar xpf - Install
```

2. As root, install the toolchain using the following commands:

```
$ su
# ./Install --tape=i686-pc-linux-gnulibc2.2-x-xscale-elf.tar.Z binaries
```

Note: The default install directory is `/opt/redhat/xscale-030422`. This can be overridden by using the `--installdir=DIRNAME` argument with the `Install` script. To get a list of the install options, use the `-help` argument.

To ensure that the correct toolchain is used, add the path to the root of the install location to your `PATH` environment setting.

```
$ export PATH=/opt/redhat/xscale-030422/H-i686-pc-linux-gnulibc2.2/bin:${PATH}
```

Note: If you obtain the RedBoot source code from the public CVS repository, the `make` file is configured for an alternate toolchain. In order for the `make` file to be used unmodified, it is necessary to create symbolic links for `arm-elf` to their `xscale-elf` equivalents as shown below:

```
$ su
# cd /opt/redhat/xscale-030422/H-i686-pc-linux-gnulibc2.2/bin
# ln -s xscale-elf-gcc arm-elf-gcc
# ln -s xscale-elf-ar arm-elf-ar
# ln -s xscale-elf-objcopy arm-elf-objcopy
```

9.4 Set Up the RedBoot* Source Tree

These steps set up the source tree to build a ROM redboot image for the "ixdp465"

1. Make a working directory for the build and untar the source code for RedBoot and unzip the NPE support component zip files:



```
$ mkdir redboot-build
$ cd redboot-build
$ tar xvfz <path-to-file>/redboot-intel-ixp4xx-070320.tar.gz
$ unzip <path-to-file>/Redboot-v2_04-npe-microcode.zip
```

Untarring the source creates the directory `redboot-intel-ixp4xx-070320`. All steps following this will reference this as the `<work_dir>`.

2. Set up the build directory and environment:

```
$ cd redboot-intel-ixp4xx-070320
$ export TOPDIR= `pwd`
$ export ECOS_REPOSITORY=${TOPDIR}/packages
$ cd ${ECOS_REPOSITORY}
$ chmod +x ecosadmin.tcl
```

Note: The use of the `export` command is specific to the bash shell. Use `setenv` for c shell.

3. Copy the file NPE microcode into the source tree.

```
$ cp ../../IxnPeMicrocode.c ${ECOS_REPOSITORY}/devs/eth/intel/npe/npeDl/current/src
```

Note: You may find it necessary to rebuild the `ecosconfig` utility. Follow the steps to backup the original, rebuild and then install the `ecosconfig` utility on the host system. These steps assume that you start in the RedBoot source code install directory:

```
$ su
# cd /opt/redhat/xscale-030422/H-i686-pc-linux-gnulibc2.2/bin/
# cp ecosconfig ecosconfig.orig
# exit
$ mkdir ${TOPDIR}/ecbuild
$ cd ${TOPDIR}/ecbuild
$ ../host/configure
$ make
$ su
# cp tools/configtool/standalone/common/ecosconfig
/opt/redhat/xscale-030422/H-i686-pc-linux-gnulibc2.2/bin/
# exit
$
```



9.5 Build the RedBoot* Images

The following commands build a ROM-based image for the IXDP465:

```
$ export Board=ixdp465
$ mkdir ${TOPDIR}/build
$ cd ${TOPDIR}/build
$ ecosconfig new ${Board}_npe redboot
$ ecosconfig import ${ECOS_REPOSITORY}/hal/arm/xscale/${Board}/current/misc/
redboot_ROM.ecm
$ ecosconfig tree
$ make
```

When the build is complete the images are in the directory:

```
${TOPDIR}/build/install/bin/:
```

The images created are:

- redboot.bin
- redboot.elf
- redboot.img
- redboot.srec

Note:

The type of image (for example, ROM) is not encoded in the file name when built by these steps. The type is encoded in the file name by the release packaging done by Red Hat. If you desire the type to be encoded in the name you must manually rename the file.

9.6 Configuring the RedBoot* Build for Other Targets

The build system also supports building RedBoot for other platforms and memory configurations.

In order to build a RedBoot image for another platform you must clean-up the source tree to remove the previous build setup for the desired board and memory configuration and then run `make`:

1. Run `make clean` in the `${TOPDIR}/build` directory
2. Remove the directory `${TOPDIR}/build`
3. Repeat the steps in [Section 9.5](#) to set the target platform and eCos configuration.

9.6.1 Setting the Target Platform

Replace the `{Board}` environment variable (the Board name) in the `export` command listed in [Section 9.5](#) with one of the following:

- `grg` — For ADI Engineering's Coyote* Gateway Reference Design



- `ixdpg425` — For the Intel® IXDPG425 Network Gateway Development Platform
- `ixdp425` — For the Intel® IXDP425 Development Platform
- `ixdp465` — For the Intel® IXDP465 Development Platform
- `kixrp435` - For the Intel® IXP435 Multi-Service Residential Gateway Reference Platform

9.6.2 Selecting the eCos Configuration

The eCos configuration file (ecm file name extension) is used by the `ecosconfig import` command to configure the source tree to build the particular type of RedBoot image.

The configuration files have the type encoded in the file name in the same manner as the pre-compiled binaries. Refer to [Section 3.0](#) for these types.

For example:

- To build a RAM-based image, use the `redboot_RAM.ecm` file.
- To build a ROM-based LE image use the `redboot_ROMLE.ecm` file.
- To build a ROMRAM-based image use the `redboot_ROMRAM.ecm` file.

For the type of configurations available for a particular board see the available ecm files in:

```
packages/hal/arm/xscale/${Board}/current/misc/
```

10.0 Additional Features

This section provides information on features in RedBoot not necessarily noted in the online documentation or that are processor/platform-specific.

10.1 Cache Policy

The default MMU cache policy for the cached regions in RedBoot on the IXDP465 platform is set to a Write-Back mode. For all other boards, the MMU policy for the cached regions is set to Write-Through mode. In order to change the policy from Write-Through to Write-Back mode, the `IXP_MAP_SDRAM` memory map settings for SDRAM must be modified and an appropriate flush area must be created. This requires modifying the file `hal_platform_setup.h` in the RedBoot source code directory:

```
packages/hal/arm/xscale/<Board>/current/include
```

For additional information on cache settings, refer to the *Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Developer's Manual* or the *Intel® IXP45X and Intel® IXP46X Product Line of Network Processors Developer's Manual* or the *Intel® IXP43X Product Line of Network Processors Developer's Manual* as noted in [Section 1.3](#).



10.2 GPIO Mappings

For the IXDP465 platform, the GPIOs are mapped for many purposes. An FPGA is used to route GPIO to I/O (any GPIO to any I/O) or interrupt signals for all peripherals, eliminating the need for jumpers (as is the case with the IXDP425 platform). Under software control, this FPGA helps control the GPIO signals. Any I/O from the mezzanine card modules (or onboard peripheral) can be routed internally by the FPGA via software control to any of the GPIOs. The macros to access the GPIO are located in the source code file:

```
packages/hal/arm/xscale/<Board>/current/src/<Board>_misc.c.
```

10.3 PCI Enumeration

RedBoot will enumerate the PCI bus and assign memory and I/O resources to the devices found. Since the CPU window into PCI space is only 64M, it may be necessary for RedBoot to ignore certain devices that need special consideration. Each of the platforms provides a registration mechanism where a decision can be made whether or not RedBoot assigns resources to a given PCI device BAR. The function:

```
int hal_plf_pci_ignore_bar(void *dev_info, int bar)
```

is located in the source code directory:

```
packages/hal/arm/xscale/<Board>/current/src/<Board>_pci.c
```

This function returns a non-zero value if the given device BAR is ignored and no resources are assigned to it.

§ §

