Intel®IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor: Customizing RedBoot*

Application Note

September 2004

Document Number: 254308-002

Contents

1.0 Introduction 7
- 1.1 Audience 7
- 1.2 Related Documents 8
- 1.3 Acronyms 9

2.0 What is Customizing? 9
- 2.1 Production Versus Development Bootloader 9
- 2.2 Requirements to Consider 10
- 2.3 Licensing 10

3.0 Getting Started 11
- 3.1 Prerequisites 11
- 3.2 Development Steps 11
- 3.3 First Step — Modify the Version 12

4.0 Items to Address When Customizing for a New Board 12
- 4.1 Moving from the IXDP425 / IXCDP1100 platform to a Custom Board 12
- 4.2 Where to Make Modifications in the Source Tree 12
- 4.3 Updating the Build Sequence 13
  - 4.3.1 Add Packages Created 13
  - 4.3.2 Changing the BOARD Variable 13
  - 4.3.3 Apply Patches 14
  - 4.3.4 Rebuilding Notes 14

5.0 How-To's 14
- 5.1 Obtaining RedBoot from the Public CVS Repository 14
- 5.2 How to Modify the Version String Reported 15
- 5.3 Where to Modify Diagnostic/Console UART Use 15
- 5.4 Where to Adjust the Platform Initialization Steps 16
  - 5.4.1 How to Adjust for Alternate Flash Parts and Size 16
  - 5.4.2 How to Adjust for Alternate SDRAM Size 17
  - 5.4.3 Where to Adjust for PCI Changes 17
  - 5.4.4 Where to Reconfigure the GPIO Assignments 17
- 5.5 Simple Method of Adding an Interrupt Service Routine (ISR) 18
- 5.6 How Can I Restrict Access to Commands? 18
- 5.7 How to Set the Architecture ID for Proper Linux Boot Support 18
- 5.8 Do I Need to Use the EEPROM? 19
- 5.9 How to Remove/Modify the LED Display Support 19
- 5.10 How Do I Add a Self Test? 20
- 5.11 Are There Any Preexisting Test Routines? 20
- 5.12 How Do I Add a Command? 20
- 5.13 Are Documents Provided with the Source? 21
- 5.14 Telnet Access to RedBoot CLI 21
- 5.15 How to Update the ROM Image 21
- 5.16 How to Place RAM Mode Update Image in Flash for Future Use 23
- 5.17 How Do I Add My Custom Source to the Build Tree? 23
  - 5.17.1 How to Make Changes to the Source Tree into an eCos Package 23

5.17.2 Why Do I Need to Create a Package for Modifications to the Source Tree?..... 24
5.17.3 Package Content 24
5.17.4 Testing the Package 25
5.18 Passing Information Between RedBoot* and Linux 25
5.19 How to Change the Default Values Used for fconfig 25
5.19.1 Configuration Option Basics 26
5.19.2 Example: Changing the Default Boot Script in Source Code..... 26
5.19.3 Example: Change the Boot Script in the RedBoot* CDL 27
5.20 General eCos File Type Questions and Answers..... 27

A Overview of the Source Tree 28

A.1 eCos and RedBoot* 28
A.2 eCos Packages..... 28
A.3 Primary eCos Packages 28

B Build Script 30

B.4 Usage 30
B.5 What Does the Build Sequence Do? 30
B.6 build-redboot Script Source 32

C Makefile 35

Revision History

Date	Revision	Description
September 2004	002	Updated product branding.
February 2004	001	Initial release.

This page is intentionally left blank.

1.0 Introduction

The bootloader is a fundamental software component for most computer systems; it provides the initialization sequence for the processor and hardware components so that the system is operational. Primarily intended to boot Linux, RedHat* RedBoot* is the bootloader provided for the Intel® IXDP425 / IXCDP1100 Development Platform. RedBoot is provided in both binary and source form, and is developed and maintained by RedHat; for further information regarding RedBoot, see http://sources.redhat.com/redboot/.

Note that potential issues arise when product development shifts from the IXDP425 / IXCDP1100 platform to a prototype/custom baseboard. This shift will likely involve modification of the RedBoot source code to support the custom board design. So a fundamental question is:

> *"What does RedBoot need to do (or be modified to do) in order to support my custom board and the product that I am intending to ship?"*

In many cases, the default bootloader source configuration is acceptable; there may be no need to modify RedBoot. But, the default configuration may not be acceptable for your final product.

This application note is intended to provide guidance on issues that may arise when customizing (aka "productizing") Redboot. Main topics covered in this document include:

- What is customizing?
- Getting started
- Installing and building RedBoot
- A tour of the RedBoot source tree:
  - — What needs to be modified, reconfigured, added, and removed when moving to a custom board
  - — Where are modifications required?
- Typical customizing options — How-to's

1.1 Audience

This document is intended for software engineers (using the RedBoot bootloader) who are developing software and board support packages (BSPs) for custom hardware based upon the IXP42X product line and IXC1100 control plane processors. The application note references the IXDP425 / IXCDP1100 platform, but can apply to the Coyote* Gateway Reference Design (GRG), or the Motorola* PrPMC1100* platforms.

This document is NOT intended as a training guide for developing eCos* or RedBoot extensions. It is merely a guide to help an experienced software developer become more familiar with issues when modifications to RedBoot are required.

1.2 Related Documents

Document	Document Number
Intel® IXP400 Software Programmer's Guide	252539
Intel® XScale™ Microarchitecture Programmer's Reference Manual	273473
Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Datasheet	252479
Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor Developer's Manual	252480
*Intel® IXP400 Software - RedHat * Boot-loader v1.92 Software - Software Release Notes*	N/A
Intel® IXP400 Software Release 1.5 Software Release Notes	N/A
Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor: Operating System Porting Guide	253548

- 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/ixp425swr1.htm
- For the latest eCos documentation, consult the online reference at
  http://sources.redhat.com/ecos/docs-latest/
- A generic RedBoot users guide is available on line at
  http://sources.redhat.com/ecos/docs-latest/ref/redboot.html

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

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

1.3 Acronyms

Acronym	Description
BASH	Bourne Again Shell – Linux command shell
BSP	Board Support Package
CDL	Component Definition Language
ece	eCos Configuration – file name extension
eCos	Embedded Configurable Operating System
epk	eCos Package – file name extension
FIS	Flash Image System
GPIO	General-Purpose Input Output
GRG	Generic Residential Gateway
HAL	Hardware Abstraction Layer
ISR	Interrupt Service Routine
LAN	Local Area Network
LSB	Least-Significant Byte
MAC	Multiply/Accumulate
MMU	Memory Management Unit
MSB	Most-Significant Byte
MTD	Memory Technology Device
NPE	Network Processing Engine
PCI	Peripheral Component Interconnect
TCL	Tool Command Language

2.0 What is Customizing?

2.1 Production Versus Development Bootloader

The bootloader as provided by Intel for the IXDP425 / IXCDP1100 platform is a development-platform bootloader. This means it is intended primarily to boot, load and execute images on the IXDP425 / IXCDP1100 platform and not necessarily intended for a shipping product. It can work just fine as production bootloader, but depending upon the requirements for the product in the field, there are features that you may not want customers to access. There may be behavior in the development bootloader that is not desired and could be considered inconsistent with a bootloader for use in a production product.

For example:

- Changing the MAC address for the NPE Ethernet ports
- Loading and running arbitrary images
- Modifying the boot script or enabling the debug stub
- Changing the bootloader flash image/configuration

- Obtaining an IP address by the bootloader
- Runtime menu availability/visibility

The following additional changes may need to be made to support your board:

- RAM and flash configuration
- Processor initialization steps — board setup
- Bootloader announcement (version and copyright notice) and boot delays.

2.2 Requirements to Consider

Other bootloader functionality that may be required to support product features include:

- Loading images in the field
- Update of the bootloader itself
- Pre-OS load support
- Manufacturing support (board and hardware tests).
- Maintaining separate development and production bootloaders
- Customizing the command set; it may be desirable to change the commands available to typical users
- Modifying the default configuration; the default, especially for the boot script, may need to be changed
- How/where is the MAC address for the NPE Ethernet ports stored? The Coyote gateway platform keeps its MAC address in the fconfig space. The MAC is then managed by extending the command interface. This is a model for storing the MAC address in a location other that I^2C.

If your (Linux-based) application can update itself then you do not need the above-mentioned capability in the bootloader; this may actually be the cleanest way to handle updates as it turns the process into a user-space managed application that you can control. If you must have redundant signed images, then the bootloader, for use and loading, needs to know which images to manage and how to manage those images.

2.3 Licensing

RedBoot is licensed under the eCos license, which affects what you can do, how you release and redistribute the source code and related customizations.

- The eCos license is available on line at:
  http://sources.redhat.com/ecos/license-overview.html
- The NPE/ixp400 access library is covered by the Intel® IXP400 DSP Software license agreement and can be viewed on line at: http://www.intel.com/design/network/swsup/sla_425.htm
- A comprehensive collection of the relevant open source licenses is available at:
  http://www.opensource.org/licenses/

3.0 Getting Started

3.1 Prerequisites

It is recommended that you complete the following:

- Get and install SourceNavigator*; this will allow you easily search, read, study sections of the source tree that you may need to modify. SourceNavigator is available at: http://sourcenav.sourceforge.net/
- Get the RedBoot release notes, source code and the NPE epk file (eCos Package containing the NPE Ethernet driver and portions of the Intel® IXP400 Software to support running the driver in eCos). These items are available online at:

  http://www.intel.com/design/network/products/npfamily/ixp425swr1.htm
- Set up the toolchain and source per the release notes.
- Set up the build-redboot script; see Appendix B, “Build Script.”
- Build by running the build-redboot script (or suitable replacement).
- Verify your binary build loads and runs on the IXDP425. See Chapter 5.0, “How to Update the ROM Image.”
- Archive.

The fundamental goal of the prerequisites is to make sure that, before proceeding, you can compile and build a functional baseline version based upon unmodified source. No matter what procedure you use, this verifies that you have a system setup to start modifying RedBoot.

Note: A useful and highly recommended book is — *Embedded Software Development with eCos* by Anthony J. Massa; for further information, see Section 1.2 of this application note. The book provides the fundamentals on eCos and RedBoot that you will need if you are intending to perform significant updates to RedBoot.

3.2 Development Steps

The sequence of development steps depends upon production-bootloader requirements. Once RedBoot is running and loading images for your custom platform, then you have a development bootloader. So, what are the differences between a development bootloader, production images and the associated configuration? You need to consider what to customize, e.g., the product requirements. Customization activities typically include:

- Determining the changes necessary to support your custom hardware
- Determining the requirements for the bootloader as shipped in the final product
- Planning the development
- ‘Porting’ RedBoot to your custom hardware
- Making product-specific modifications
- Implementing and testing

3.3 First Step — Modify the Version

Start small by changing the version displayed. To change the version, see the ‘how-to’ in Chapter 5.0, “How to Modify the Version String Reported.”

- Build
- Verify
- Archive

For an overview of the RedBoot source tree, see Appendix A, “Overview of the Source Tree”.

For information on building RedBoot, see Appendix B, “Build Script”.

4.0 Items to Address When Customizing for a New Board

Before starting, reference the Intel® IXP42X Product Line of Network Processors and IXC1100 Control Plane Processor*: Operating System Porting Guide*. This document provides details on what needs to be addressed in the Linux kernel for a new platform and contains relevant information that can be helpful in modifying RedBoot for your platform.

4.1 Moving from the IXDP425 / IXCDP1100 platform to a Custom Board

The number of changes needed to the HAL (Hardware Abstraction Layer) depend upon how close the baseboard is to the IXDP425 / IXCDP1100 platform or Coyote gateway platform. The primary change that most boards will need to make is to update the memory. You may need to consider the items needed to port Linux to your platform; note, however, the RedBoot changes are confined to:

- CDL and configuration files
- SDRAM
- Flash
- Interrupts
- GPIO
- PCI
- Other peripherals
- Debug port number (default is 9000)

4.2 Where to Make Modifications in the Source Tree

Note: Before making changes to the source or source tree, make a back-up copy. Modify the copy, then use patch files or epk files to manage your changes. Or use a revision control system like CVS to track your changes. For an overview of the source tree, see Appendix A, “Overview of the Source Tree”.

The primary location to make modifications is to the HAL package. Choose a board already defined in the HAL structure and copy the contents to a new directory under the HAL. For example, if your board is named “new425”, set up the HAL support initially using the commands

```
cd hal/arm/xscale/
mkdir new425
cp -a ixdp425 new425
```

When creating a new board, be sure to update the handling of the board name by the `${BOARD}` variable in the build script. See Appendix B, “What Does the Build Sequence Do?,”

Rename the file hal_arm_xscale_ixdp425.cdl to hal_arm_xscale_new425.cdl. Modify and update the following elements as necessary:

- CDL and configuration files
- SDRAM
- Flash
- Interrupts
- GPIO
- PCI
- Other peripherals
- Debug port number (default is 9000)

4.3 Updating the Build Sequence

Depending upon the nature of the changes, there is at least one place in the build sequence that may need to be updated. You may decide that the easiest manner to update the RedBoot source and configuration files is to copy or patch files. Before starting, review Section B.2, “What Does the Build Sequence Do?” on page 30 and review the build script source in Appendix B, “Build Script.”

Note: It is recommended that the generated files not be modified; tracking the generated file content and patching may not be easily maintainable.

4.3.1 Add Packages Created

If a epk file was created to contain your changes, then the package must be added to the ecode.db.; the ecosadmin command is used for this. If your package is “new425.epk”, then the package is added using the command

```
ecosadmin add new425.epk
```

Place this command just after the existing command used to add the NPE epk file.

4.3.2 Changing the BOARD Variable

If the HAL was copied from an existing board and updates made, then the BOARD variable used in the script can be set to the name of your board. See Section B.2, “What Does the Build Sequence Do?” on page 30 for details and use of the `${BOARD}`variable.

4.3.3 Apply Patches

If you are modifying small sections of code that do not affect generated source modules, then the changes should be applied to the baseline code before the `ecosconfig tree` command.

If you determine that the easiest way to apply changes is to modify generated code, then apply a patch with your changes just after the `ecosconfig tree` command.

4.3.4 Rebuilding Notes

Modifying any template files (cdl, ecm, ece) affects generated code. So, in general, you must perform the configuration and tree generation steps again to have the changes take place in the source. The build-redboot script cleans the configuration so that this is not an issue.

Note: It is NOT necessary to change the NPE support for each new board. So, if you DO NOT update the NPE support, then will need to modify the build script to NOT use the BOARD variable. The build will need the baseline NPE support. So update the build script commands to replace

```
ecosconfig add ${BOARD}
```

with

```
ecosconfig add ixdp425_npe
```

5.0 How-To's

5.1 Obtaining RedBoot from the Public CVS Repository

RedBoot sources can be obtained from the public repository at http://sources.redhat.com/ecos/boards/ixdp425.html

You can obtain the RedBoot sources online using CVS:

```
% cvs -z3 -d :pserver:anoncvs@sources.redhat.com:/cvs/ecos co -D "2003-06-22
00:00:00 GMT" packages net
```

This command gets the exact modules used the build the RedBoot binaries that Intel provides (not including the NPE support library)

To get the most recent source tree use the command:

```
csv -z3 -d :pserver:anoncvs@sources.redhat.com:/cvs/ecos co ecos ecos-host packages
net
```

This will get the eCos and eCos-host package source as well.

Alternatively, you can obtain weekly snapshots of the CVS repository from

http://www.ecoscentric.com/devzone/snapshots.shtml

5.2 How to Modify the Version String Reported

There are several 'components' that support displaying the version; see information in the file:

```
packages/redboot/current/src/main.c.
```

The version is reported in function `do_version()`. You can add/remove information as desired. For more detailed customizing, formulating and formatting of the version string, see the file:

```
packages/redboot/current/src/version.c.
```

This contains a confusing logic sequence that concludes with the "RedHat Certified Release" message. The important thing is that the baseline version is in the define

```
CYGDAT_REDBOOT_CUSTOM_VERSION
```

This is in the file `packages/redboot/current/cdl/redboot.cdl`, the CDL option allows "`cdl_option CYGDAT_REDBOOT_CUSTOM_VERSION`" to be specified in an ecm or ecc file. This can then be used to extend the version display. This is the most convenient way to add a custom version identifier. The macro `CYGACC_CALL_IF_MONITOR_VERSION()` is an in-line routine. There are two locations that can be found using SourceNavigator. The routine is in the file

```
package/hal/common/current/include/hal_if.h
```

The file

```
build/install/include/cyg/hal/hal_if.h
```

is a generated file that uses the primary file. The configuration item, `HAL_PLATFORM_EXTRA`, allows for indicating additional platform detail in the version. This is defined in the file

```
packages/hal/arm/xscale/ixdp425/current/cdl/
hal_arm_xscale_ixdp425.cdl
```

When the tree is configured, the define is generated using the CDL file to take the configuration content and write the define with the value into the file

```
build/install/include/pkgconf/hal_arm_xscale_ixdp425.h
```

5.3 Where to Modify Diagnostic/Console UART Use

By default, RedBoot supports two console ports for the IXDP425 / IXCDP1100 platform:

- UART0 is the common console port for Linux and RedBoot.
- UART1 is labeled "Console" on the IXDP425.

For Linux development it is desirable to connect to a single UART for console use — use UART0. RedBoot has an interesting feature in that ANY traffic on UART0 or UART1 will cause RedBoot console control to changed to that UART. So if traffic appears on UART1 while booting, RedBoot can switch control to that port. Depending on the traffic, it can cause the boot sequence to halt (e.g., a ctrl-C character received will halt the boot script from running). This can lead to a problem in deployment.

To modify RedBoot to select a single or NO UART as a console, modify the file

`packages/hal/arm/xscale/${BOARD}/current/cdl//hal_arm_xscale_ixsp425.cdl`

Find the cdl_options that define the value for CYGSEM_HAL_IXP425_PLF_USES_UARTn and change the default value.

To see how this controls the console output (also referred to as diagnostic output), see the file

`packages/hal/arm/xscale/ixp425/current/src/ixp425_diag.c`

5.4 Where to Adjust the Platform Initialization Steps

Platform initialization consists of:

- Setting up the processor for proper operation
- Initializing the Chip Select (CS) for new/updated components on the expansion bus – flash
- Initializing the SDRAM
- Initializing the PCI bus
- Setup MMU
- Setup for Big-endian (BE) or Little-endian (LE)

The primary macros for platform initialization and start-up code is in the files

`packages/hal/arm/xscale/{BOARD}/current/include/hal_platform_setup.h.`

`packages/hal/arm/xscale/{BOARD}/current/include/hal_platform_extras.h.`

The initialization steps for the board are performed by the code in the file

`packages/hal/arm/xscale/{BOARD}/current/src/$(BOARD}_misc.c.`

5.4.1 How to Adjust for Alternate Flash Parts and Size

Adjustment for flash parts are made in the device package support code. This can be found in

`packages/devs/flash/arm`

Each of the boards that use the Strataflash has a BOARD named directory. The file

`${BOARD)/current/include/$(BOARD}_strataflash.inl`

contains the flash device definitions, including number of devices, size and base address.

In addition, you may need to modify the file

`/packages/hal/arm/xscale/ixdp425/current/include/ixdp425.h`

Find the line that contains the define IXP425_EXP_CS0_INIT. The current definition is provided below. Modify the value of the define for the flash organization that you are using. The define sets the initialization value for the chip select of the expansion bus.

```
// CS0 (flash optimum timing)
#define IXP425_EXP_CS0_INIT \
(EXP_ADDR_T(3) | EXP_SETUP_T(3) | EXP_STROBE_T(15) | EXP_HOLD_T(3)
| \
```

```
 EXP_RECOVERY_T(15) | EXP_SZ_16M | EXP_WR_EN | EXP_BYTE_RD16 |
EXP_CS_EN)
```

Macros are used to set the define value. The macros set expansion bus control bit values properly and are named per the ixp425 register set definitions. The definition of the macros can be found in the file

`/packages/hal/arm/xscale/ixp425/current/include/hal_ixp425.h`

5.4.2 How to Adjust for Alternate SDRAM Size

RedBoot must know the RAM size. To adjust the RAM size, see the file

`/packages/hal/arm/xscale/ixdp425/current/include/ixdp425.h`

Also note that the values must match the setup in the page table in file

`ixdp425/current/include/hal_platform_extras.`

The default value is 256M SDRAM on the IXDP425. This is set by the defines as indicated.

```
#define SDRAM_SIZE 0x10000000 // 256MB
#define IXP425_SDRAM_CONFIG_INIT (SDRAM_CONFIG_CAS_3 |
SDRAM_CONFIG_4x32Mx16)
#define IXP425_SDRAM_REFRESH_CNT 0x081
#define IXP425_SDRAM_SET_MODE_CMD SDRAM_IR_MODE_SET_CAS3
```

The IXP425_SDRAM defines are used to set up the SDRAM configuration registers during platform initialization.

5.4.3 Where to Adjust for PCI Changes

See the function hal_plf_pci_init(). This uses the defines that are in the file

`/packages/hal/arm/xscale/ixdp425/current/include/ixdp425.h`

Modify the defines as necessary to update the PCI support.

5.4.4 Where to Reconfigure the GPIO Assignments

See the file `/packages/hal/arm/xscale/ixdp425/current/include/ixdp425.h` for the defines used to set the GPIO numbers.

```
#define GPIO_EEPROM_SDA 7
#define GPIO_EEPROM_SCL 6
#define GPIO_ENET1_INT_N 5
#define GPIO_ENET0_INT_N 4
#define GPIO_HSS0_INT_N 3
#define GPIO_HSS1_INT_N 2
#define GPIO_DSL_INT_N 1
```

5.5 Simple Method of Adding an Interrupt Service Routine (ISR)

An application may configure a normal C function to be an ISR. The address of the function can be added into the vector table at the correct index and enable the interrupt at its source.

Refer to http://sources.redhat.com/ecos/docs-latest/ref/ixdp425.html#AEN6568 for the interrupt numbers assigned to the IXDP425 / IXCDP1100 platform.

5.6 How Can I Restrict Access to Commands?

There are many ways to restrict access to commands, including:

- Hide them (no help); comment-out using `#ifdef -#endif` conditional code
- Add a feature to enable menus when in particular mode only
- Use a spare GPIO pin to detect a mode
- Set a flag in the fconfig space to enable/disable extended menus

5.7 How to Set the Architecture ID for Proper Linux Boot Support

Linux kernels after 2.4.18-rmk6 require that the bootloader pass the machine ID to the kernel decompressor. Some kernels have been modified to work around this requirement as not all ARM platform have been able to comply. For a summary of the ARM Linux kernel boot requirements, see http://www.arm.linux.org.uk/developer/booting.php.

The machine ID can be changed by modifying the CDL file

```
arm/xscale/{BOARD}/current/cdl/hal_ar,m_xscale_{BOARD}.cdl
```

The define_proc script contains a sequence to write the define

```
#define HAL_PLATFORM_MACHINE_TYPE nnn,
```

where nnn is the registers Machine ID number.

The current machine types supported by RedBoot are shown in Table 1.

Table 1. ARM Machine Type IDs

Machine ID	Machine Name
254	IXDP425 / IXCDP1100 platform
260	Motorola PrPMS1100 (an IXC1100–based board)
290	Coyote gateway platform

See http://www.arm.linux.org.uk/developer/machines/ for instructions on how to register the machine ID. Refer to linux/arch/arm/boot/compressed/head-xscale.S for the Linux code that uses the machine ID. The machine IDs supported by the kernel are defined in linux/arch/arm/tools/mach-types.

Note: You can modify the Linux kernel source so that the need for the machine ID is eliminated. But you may limit the ability to configure the kernel in a manner that is unique to your board -NOT RECOMMENDED. But modifying the kernel source in this manner may make sense for a customized system rather than for a general-purpose Linux kernel and bootloader.

5.8 Do I Need to Use the EEPROM?

This is specific to designs based closely on the IXDP425 platform; therefore, using the EEPROM depends on your board design. If you plan to use EEPROM connected via I^2C in your system, then the support is already enabled in RedBoot and is used to store the MAC address for the NPEs. So if you only plan to use the EEPROM for storing the MAC addresses, then you may want to evaluate alternative locations for storing the MAC address. e.g., in flash instead of EEPROM.

What are the trade-offs?

1. With EEPROM, you have BOM costs and GPIO consumed. This is the default RedBoot support.
2. Flash – uses the configuration space in RedBoot (extended). RedBoot must be modified to use flash for this. A model for how to do this is already in the Coyote gateway platform support package, but is NOT a configurable option, so modifications are required in non-Coyote gateway platform systems.

If you do use the EEPROM, you can reclaim the GPIO used to access EEPROM.

If you want to modify the location (offset) at which the MAC address is stored, the EEPROM is support is provided in the file

```
packages/hal/arm/xscale/IXDP425/current/src/ixdp425_misc.c
```

Note: To store the MAC address in the flash, the `ixdp425_misc.c` module needs to be updated, removing the existing EEPROM support and updating the code to extend the platform configuration in the fconfig structure. Refer to the GRG source for a model on how to modify your source to add this feature.

5.9 How to Remove/Modify the LED Display Support

In the init process macro the macro DISPLAY is used. This is in the file:

```
packages/hal/arm/xscale/${BOARD}/current/include/hal_platform_setup.h
```

The DISPLAY macro is defined in:

```
packages/hal/arm/xscale/${BOARD}/current/include/ixdp425.h.
```

Use an `#ifdef - #endif` block to remove the code generated in a manner similar to how the delay is removed. The function HEX_DISPLAY should be removed as it is available for use but not used in the current source tree.

The expansion bus does not need to be setup to use CS2 to access the LED displays if they are not on your board. So the define

```
#define IXP425_EXP_CS2_INIT
```

must not be used. This is due to the `platform_setup` macro applying the CS3 init in the define if it is available. This is in the file:

```
hal_platform_setup.h
```

5.10 How Do I Add a Self Test?

In the file

```
packages/redboot/current/src/main.c
```

see the declaration for `bist()`:

```
// Builtin Self Test (BIST)

externC void bist()
```

This is called when RedBoot is starting up.

Note: It may not be appropriate to add a significant amount of functionality here as this can slow down the boot and load time for your application. You need to consider the power-on self test requirements and how these relate/affect the amount of time that the system requires to boot and become operational.

5.11 Are There Any Preexisting Test Routines?

Yes, they are for alternate platforms that are Intel XScale core–based, but can be used and added to the source tree. An example of this is the RAM test for the 80321 board. The RAM tests and algorithms employed are still valid. The menus and physical location/size are fixed. It should take a minimal level of effort to 'migrate' (copy code module, and update) the desired tests routines into the IXDP425 / IXCDP1100 platform source tree.

See the online reference for the `diag` command at:

http://sources.redhat.com/ecos/docs-latest/ref/iq80310.html#AEN6307

Note: The source for the `diag` command is available from the on-line CVS repository in the source tree:

```
packages/hal/xscale/iq80321/current/src/diag
```

5.12 How Do I Add a Command?

See main.c for the declaration of a few of the key commands. Review the `cmd` structure in packages/redboot/current/include/redboot.h. You can add a command to most any 'main -level' file as long as the command declarations follow the pattern in main.c. The command structure is then added to the main command list as part of the build. The basic steps include:

- Write the command function;

- Create the `cmd` structure wrapper.

See the file `main.c` and how the version command is defined and implemented; use this as an example of how to add a simple command.

5.13 Are Documents Provided with the Source?

Yes. The main entry point for the docs is the file docs/redboot/html/redboot.html.

Also see the files

`docs/redboot/html/ixdp425.html for the ixdp425`

`docs/redboot/html/getting-started-with-redboot.html`

5.14 Telnet Access to RedBoot CLI

Note: This section provides a hint and a warning about a feature.

If RedBoot is configured for GDB access, this port can also be used for Telnet access to the RedBoot CLI, prior to loading images. This allows you to connect to a device when a console port is not provided externally. This is also useful when the serial port hardware is not present in the production build of the board. To access this, use the Telnet command and specify a port of 9000. For example, for a board with an IP address of 192.168.200.201, the command is:

```
telnet 192.168.200.201 9000
```

Note: To help mitigate this risk of unauthorized access, it is easy to reassign the port number; in addition, RedBoot will only respond to requests for a short period of time during boot.

When the Telnet session connects, the "RedBoot >" prompt and commands are available. This is a good reason to add a command-enable sequence as all commands are available. Also depending upon the configuration, the delay from RedBoot init complete to starting to loading images may be short. So there is a timing aspect to connecting to the RedBoot prompt via Telnet.

For some products, using Telnet to connect to RedBoot is a way to initiate firmware updates.

5.15 How to Update the ROM Image

Updating the image that runs from ROM must be done using an image that runs from RAM instead of running from flash. Why? The flash partition — that needs to be updated — is also being accessed for instructions as RedBoot runs out of flash in the ROM configuration. So RedBoot must be running from RAM to be able to update the flash partition used by the ROM-mode RedBoot.

The basic sequence of steps is:

- Change the `IMAGE_TYPE` to RAM in the build script/makefile.
- Build a RAM image.
- Load the RAM image and run.
- Load the ROM image, unlock the flash, use fis to store the new image.

- Reset

For the IXDP425 / IXCDP1100 platform, the update sequence is captured and the commands entered are identified.

```
+Ethernet eth0: MAC address 00:02:b3:3c:15:de
IP: 192.168.200.200/255.255.255.0, Gateway: 0.0.0.0
Default server: 192.168.200.254, DNS server IP: 0.0.0.0

RedBoot(tm) bootstrap and debug environment [ROM]
Red Hat certified release, version 1.92 - built 23:01:46, May  2 2003

Platform: IXDP425 Development Platform (XScale)
Copyright (C) 2000, 2001, 2002, Red Hat, Inc.

RAM: 0x00000000-0x10000000, 0x0001f880-0x0ffd1000 available
FLASH: 0x50000000 - 0x51000000, 128 blocks of 0x00020000 bytes each.
== Executing boot script in 3.000 seconds - enter ^C to abort
^C
RedBoot> load -v redboot_RAM.srec
Using default protocol (TFTP)
Entry point: 0x00020040, address range: 0x00020000-0x0005701c
RedBoot> go 0x20040
+Ethernet eth0: MAC address 00:02:b3:3c:15:de
IP: 192.168.200.200/255.255.255.0, Gateway: 0.0.0.0
Default server: 192.168.200.254, DNS server IP: 0.0.0.0

RedBoot(tm) bootstrap and debug environment [RAM]
Non-certified release, version 1.92 - built 15:58:38, Apr  8 2003

Platform: IXDP425 Development Platform (XScale)
Copyright (C) 2000, 2001, 2002, 2003 Red Hat, Inc.

RAM: 0x00000000-0x10000000, 0x0006d5e0-0x0ffd1000 available
FLASH: 0x50000000 - 0x51000000, 128 blocks of 0x00020000 bytes each.
== Executing boot script in 3.000 seconds - enter ^C to abort
^C
RedBoot> load -v redboot_ROM.srec -b 0x01600000
Using default protocol (TFTP)
Address offset = 0xb1600000
Entry point: 0x01600040, address range: 0x01600000-0x0163d1dc
RedBoot> fis list
Name              FLASH addr  Mem addr    Length      Entry point
RedBoot           0x50000000  0x50000000  0x00040000  0x00000000
RedBoot config    0x50FC0000  0x50FC0000  0x00001000  0x00000000
FIS directory     0x50FE0000  0x50FE0000  0x00020000  0x00000000
zimage            0x50040000  0x01600000  0x000C0000  0x01600000
ramdisk           0x50100000  0x00800000  0x001E0000  0x00800000
RedBoot> fis create RedBoot -b 0x01600000
An image named 'RedBoot' exists - continue (y/n)? y
... Erase from 0x50000000-0x50040000: ..
... Program from 0x01600000-0x01640000 at 0x50000000: ..
... Unlock from 0x50fe0000-0x51000000: .
... Erase from 0x50fe0000-0x51000000: .
... Program from 0x0ffdf000-0x0ffff000 at 0x50fe0000: .
... Lock from 0x50fe0000-0x51000000: .
RedBoot>
```

5.16 How to Place RAM Mode Update Image in Flash for Future Use

It is advantageous to have a corresponding RAM image already available in the FIS so that the ROM image can be updated. Use the commands shown below to save a RAM-mode RedBoot image to flash.

```
RedBoot> load -v redboot_RAM.srec
Using default protocol (TFTP)
Entry point: 0x00020040, address range: 0x00020000-0x0005701c

RedBoot> fis list
Name              FLASH addr  Mem addr    Length      Entry point
RedBoot           0x50000000  0x01600000  0x00040000  0x01600040
RedBoot config    0x50FC0000  0x50FC0000  0x00001000  0x00000000
FIS directory     0x50FE0000  0x50FE0000  0x00020000  0x00000000
zimage            0x50040000  0x01600000  0x000C0000  0x01600000
ramdisk           0x50100000  0x00800000  0x001E0000  0x00800000

RedBoot> fis create RedBoot-RAM -b 0x20000 -l 0x40000
... Erase from 0x502e0000-0x50320000: ..
... Program from 0x00020000-0x00060000 at 0x502e0000: ..
... Unlock from 0x50fe0000-0x51000000: .
... Erase from 0x50fe0000-0x51000000: .
... Program from 0x0ffdf000-0x0ffff000 at 0x50fe0000: .
... Lock from 0x50fe0000-0x51000000: .

RedBoot> fis list
Name              FLASH addr  Mem addr    Length      Entry point
RedBoot           0x50000000  0x01600000  0x00040000  0x01600040
RedBoot config    0x50FC0000  0x50FC0000  0x00001000  0x00000000
FIS directory     0x50FE0000  0x50FE0000  0x00020000  0x00000000
zimage            0x50040000  0x01600000  0x000C0000  0x01600000
ramdisk           0x50100000  0x00800000  0x001E0000  0x00800000
RedBoot-RAM       0x502E0000  0x00020000  0x00040000  0x00020040
RedBoot>
```

The fis list command shows that the RAM image is stored. To run this images use the commands

```
RedBoot> fis load RedBooT-RAM
RedBoot> go
```

5.17 How Do I Add My Custom Source to the Build Tree?

It depends upon the nature of the added functionality. It can be added anywhere, but the best way to add source to the tree is via an eCos package file.

5.17.1 How to Make Changes to the Source Tree into an eCos Package

To make an eCos package:

- Make a copy of the tree
- Remove the unmodified files
- Create the files pkadd.db and pksadd.txt and place into the top directory of the source tree
- Tar gzip using a command such as:

```
tar czvf mypackage.epk source-tree/
```

Note: An eCos package is simply a tar.gz file renamed with an .epk extension and a few key files added to the root directory of the file. For details on exactly what ecosadmin does with the epk file, refer to the source in `ecosadmin.tcl`.

For a more detailed instructions, see the online eCos documentation at http://sources.redhat.com/ecos/docs-latest/cdl-guide/package.distrib.html.

5.17.2 Why Do I Need to Create a Package for Modifications to the Source Tree?

1. Licensing – you may be required to supply the source updates for this board.
2. It is fairly easy to do.
3. It makes for a consistent source integration and build mechanism.

5.17.3 Package Content

There are two key files that are in an eCos package file — pkadd.db and pkadd.txt.

The pkgadd.db file is significant in that this file contain a description and instructions for adding the component(s) in the package to the ecos component database. This is not as difficult as it sounds.

pkgadd.txt content can consist of a short message identifying the package. Any licensing notice can be placed here as ecosadmin will ask for y/m acceptance if this file exists. If no notice or license is desired it can be left out.

Note: For details on the section content in the example pkgadd.db file, see page 264-66 in the Massa book referred to in Section 1.2.

For example, a new425 board is defined. This board is based on the Coyote gateway platform.

```
package CYGPKG_DEVS_FLASH_NEW425 {
    alias          { "FLASH memory support for NEW425" flash_NEW425 }
    directory      devs/flash/arm/NEW425
    script         flash_NEW425.cdl
    hardware
    description "
        This package contains hardware support for FLASH memory
        on the NEW425 Reference Board."
}

package CYGPKG_DEVS_ETH_ARM_NEW425_I82559 {
    alias           { "NEW425 / Intel 82559 ethernet driver"
                      devs_eth_arm_NEW425_i82559 }
    hardware
    directory      devs/eth/arm/NEW425/i82559
    script         NEW425_i82559_eth_driver.cdl
    description    "Ethernet driver for NEW425 with Intel 82559 PCI NIC."
}

package CYGPKG_HAL_ARM_XSCALE_NEW425 {
    alias          { "NEW425 Reference Board"
                     hal_arm_xscale_NEW425 }
    directory         hal/arm/xscale/NEW425
```

```
    script              hal_arm_xscale_NEW425.cdl
    hardware
    description "
        The NEW425 HAL package provides the support needed to run eCos on an
        NEW425 Reference Board"
}

target NEW425 {
    alias { "Intel NEW425 Wireless SOHO router Reference Board" grg }
    packages {      CYGPKG_HAL_ARM
                    CYGPKG_HAL_ARM_XSCALE_CORE
                    CYGPKG_HAL_ARM_XSCALE_IXP425
                    CYGPKG_HAL_ARM_XSCALE_NEW425
                    CYGPKG_IO_PCI
                    CYGPKG_DEVS_ETH_INTEL_I82559
                    CYGPKG_DEVS_ETH_ARM_NEW425_I82559
                    CYGPKG_DEVS_FLASH_STRATA
                    CYGPKG_DEVS_FLASH_NEW425
                }
    description "
        The NEW425 target provides the packages needed to run
        eCos on an Intel NEW425 Reference Board"
}
```

5.17.4 Testing the Package

Use the following ecosadmin command to verify the package is installs properly:

```
ecosadmin add mypackage.epk
```

5.18 Passing Information Between RedBoot* and Linux

The Linux kernel can recognize, read and write MTD partitions. RedBoot can create and initialize partitions. A simple structure can then be used to share information. Also, Linux can read the fconfig partition.

Linux can also recognize partitions created and maintained by RedBoot. In the kernel config this is controlled in the CONFIG_MDT_ REDBOOT_PARTS define. This is enabled by default in most kernels that support the IXDP425 / IXCDP1100 platform.

This allow the user mode applications to access via the `/dev/mtd` and the `/dev/mtdblock0`. These are typically assigned to contain a flash file system. By defining a structure common to RedBoot and the Linux kernel and applications, a mailbox-like system can be set up.

5.19 How to Change the Default Values Used for fconfig

The fconfig command allows the RedBoot configuration to be set up/initialized and changed. The fconfig command allows modifying the boot script. ANY value in fconfig can be set to a default value. You can also extend the fconfig structure and add your own values.

The fconfig structure elements and defined configuration items are in

`packages/redboot/current/include/flash_config.h`

`packages/redboot/current/src/flash.c`

5.19.1 Configuration Option Basics

All configuration values are identified by the macro

```
RedBoot_config_option
```

The arguments to the macro provide the initial values for the following:

- prompt message displayed when interactively setting the option value
- option data variable name
- option data name tag
- enabled state (true/false)
- tconfig type (see flash_config.h for config types)
- default value

Note: Use SourceNavigator to find all the configuration variables as they are spread around the source tree; they are defined in the context of the supporting code and source module.

5.19.2 Example: Changing the Default Boot Script in Source Code

Note: This changes the boot script in generated code.

Follow these steps to change the default (empty) boot script to a new value that loads a kernel image from flash into RAM and executes.

In the file `flash.c`, find the definition for "Boot Script"; look for:

```
RedBoot_config_option("Boot Script",
                      boot_script_data
                      "boot_script", true,
                      CONFIG_SCRIPT,
                      " "
   );
```

Change the default value by editing the last line of the macro. Change the following from

```
" "
```

to

```
"fis load -r -v -b 0x01600000 zimage\nexec"
```

Save the file, rebuild and load onto your board. Use the command

```
fconfig init
```

to re-initialize the configuration date in flash. This will set all the values to the default value. Use the command

```
fconfig -l
```

to list the configuration and verify the boot script contains the desired value.

Note: This modification can be coupled with a modification to the "Run script and boot" and "Boot script timeout" configuration values.

5.19.3 Example: Change the Boot Script in the RedBoot* CDL

Note: This example shows how to change the boot script in the CDL file so that ALL generated code will use the same boot script.

Edit the file

`packages/redboot/current/cdl/redboot.cdl`

Find the cdl_option entry for

`CYGDAT_REDBOOT_DEFAULT_BOOT_SCRIPT`

Change

`default_value`

to the desired boot script string.

Regenerate the RedBoot source tree.

5.20 General eCos File Type Questions and Answers

Q: What is an .ecc file?
A: An ecc file an eCos configuration file. This specifies the packages to load, the template used, and the option settings.

Q: What is an .ect file
A: An ect file is an eCos template file

Q: When and what updates the ecos.db file?
A: The ecosadmin.tcl program updates the ecos.db file. The add command is used in the build script to add a package to the build.

Q: What is an ecm file?
A: ecm – This contains eCos configuration macros. The file is imported to provide the minimal setup of packages for the HAL and the product. This is the file where the "CYGNUM_HAL_*" variables are set to alter RedBoot; features can be enabled/disabled. See page 188, 253 in the Massa book referred to in Section 1.2.

Q: What is a CDL file?
A: CDL – Component Definition Language file. The CDL files contain code that is used to describe package components. This is the basis for eCos configurability. Each package must have at least one CDL file.

See the following online documentation for eCos:

- *The eCos Component Writer's Guide*, http://sources.redhat.com/ecos/docs-latest/cdl-guide/overview.html
- *The CDL Language*, http://sources.redhat.com/ecos/docs-latest/cdl-guide/language.html

Appendix A Overview of the Source Tree

A complete and detailed tour of the eCos source tree can be found in the Massa book referred to in Section 1.2. This appendix provides pointers to elements of the RedBoot source tree (a subset of the eCos source tree) needed for customizing RedBoot, and includes the following general information:

- Basic introduction to eCos and eCos package files
- What the primary packages are and what they are for
- Key source code modules
- Where the prime modules are located in the source tree

Note: In perusing a new source tree, a key question is "Where is main?". *main.c* is the source module that provides a summary of what RedBoot is doing and what facilities are available. main.c is located in the source file

```
packages/redboot/current/src/main.c
```

A.1 eCos and RedBoot*

RedBoot is an eCos-based application. There is a comprehensive summary of RedBoot in the Massa book (pp. 185-206), referred to in Section 1.2.

For a short summary on eCos and the eCos file types, see Section 5.20, "General eCos File Type Questions and Answers" on page 27.

A.2 eCos Packages

eCos is an organized set of packages that are configured by the ecosconfig application. Since RedBoot is an eCos application, the source tree follows the eCos source tree layout. Each package has a common structure under the package name:

```
<packagename>/current
    -cdl
    -doc
    -include
    -misc
    -src
```

Each package has its own configuration 'database' in the CDL file. The CDL file is a text file that contain statements that follow the Component Description Language syntax. See http://sources.redhat.com/ecos/docs-latest/cdl-guide/language.html for a guide to the language and file structure.

A.3 Primary eCos Packages

The primary packages relevant to RedBoot in the eCos source tree include:

- HAL

- RedBoot

A.3.1 HAL — Hardware Abstraction Layer

The HAL is essentially the BSP configuration items. The HAL package provides all the processor- and platform-specific code so that eCos will run on the platform. This includes platform initialization/startup code, configuration for memory, interrupts, GPIO, PCI bus, the expansion bus, etc. The HAL source is located in the directory

```
packages/hal/arm/
```

The following subdirectories contain specific HAL code for the IXP425 network processor–based boards:

xscale/ixp425 – the primary IXP42X product line and IXC1100 control plane processors support code

xscale/ixdp425 – provides specific IXDP425 / IXCDP1100 platform support

xscale/grg – provides specific Coyote gateway platform support

xscale/prpmc1100 – provides specific Motorola Computer Group PrPMC1100* board support.

Note: HAL code for other boards based on Intel XScale® Core processors is available if you obtain the source from the CVS repository rather than using the source zip file. See Chapter 5.0, "Obtaining RedBoot from the Public CVS Repository."

A.3.2 RedBoot*

This is the eCos package that provides the application functionality for RedBoot. The src and include directories contain the code that defines the functionality available via the command prompt as well as the networking and filesystem support.

Appendix B Build Script

The build-redboot script is a BASH command script to automate the RedBoot build commands identified in the *Intel® IXP400 DSP Software: Red Hat* Boot-Loader 1.92 Software Release Notes*, 7/2/03. This script requires that toolchain and source are set up and installed per the release notes.

A command line argument allows the target to be specified — IXDP425 / IXCDP1100 platform, Coyote gateway platform, or PrPMC1100 platform.

Copy and paste the entire script into a file. Then change the script to set the exec flag using the command:

```
chmod +x
```

which makes it executable.

B.1 Usage

The build-redboot script expects to be run in a directory above the directory where the source code is installed and will check for the source code directory redboot-intel-xscale-030618.

To run, issue the command

```
build-redboot
```

Note: This assumes the script is in the path. If you created the script in the current directory, use the command

```
./build-redboot
```

The build is interactive in that you will be asked to accept the license agreement for the NPE enabling source. Beyond that, it runs to completion. The script defaults to build RedBoot for the IXDP425 / IXCDP1100 platform; the platform can be specified on the command line.

For the Coyote gateway platform, use:

```
build-redboot grg
```

For the PrPMC1100 platform, use:

```
build-redboot prpcm1100
```

B.2 What Does the Build Sequence Do?

The build sequence uses the command ecosconfig to set up the source tree to create RedBoot. This uses templates, packages and CDL files defined in the ece file to set up and then generate the tree. Once the tree is generated a simple make is all that is required to build the configured RedBoot.

Any changes to packages or CDL files require that you regenerate the tree; therefore, to make development easier, identify the packages, templates and CDL file, modify them, and then use these to control the configuration without editing the source code. Source code will have to be patched, modified and/or added to address all the requirements of a customized RedBoot.

A primary tool used in building RedBoot is the host application, *ecosconfig*. ecosconfig manages the content of the ecos.ecc file. Note that ALL the initialization steps build the eCos configuration file. This is a CDL file that is the 'master' collection of the eCos component hierarchy that define the application to be built — in this case, RedBoot.

Note: The build script provides the commands to rebuild ecosconfig as the first step in building RedBoot. Once ecosconfig is compiled for your platform it does not need to be recompiled.

The build script annotation applies to any RedBoot build sequence typically provided in the README with the source distribution. Refer to Appendix B, "build-redboot Script Source," for the complete build script source or Appendix C, "Makefile," for a makefile.

The first commands are BASH script variable commands

```
BUILD_ECOSCONFIG="y"
RBSRC="redboot-intel-xscale-030618"
NPE_EPK_FILE="npe-1.1.epk"
IMAGE_TYPE="ROM"
```

The `${BOARD}` variable is set to the default of the IXDP425 / IXCDP1100 platform. Or can be specified in the command line.

Note: If you need a RAM-mode[1] image, set IMAGE_TYPE to RAM.

The main command sequence starts with

```
chmod +x ${ECOS_REPOSITORY}/ecosadmin.tcl
```

The above command changes the ecosadmin.tcl file (a TCL script) so that it can be executed.

The following command runs the ecosadmin.tcl script

```
${ECOS_REPOSITORY}/ecosadmin.tcl add ../../${NPE-EPK-FILE}
```

The ecosadmin.tcl script argument ADD causes the script to process the epk file, uncompressing the package content and placing the package into the source tree. This command has much functionality. The npe-1.1.epk file contains the NPE Ethernet support package required for RedBoot to use the NPE Ethernet ports.

The next command is

```
ecosconfig new ${BOARD} redboot
```

This creates a new target configuration ${BOARD} using the RedBoot template. The RedBoot template defines the baseline.

The next command is

```
ecosconfig import ${ECOS_REPOSITORY}/hal/arm/xscale/${BOARD}/
current/misc/redboot_${IMAGE_TYPE}.ecm
```

This command imports additional configuration information to set the RedBoot startup mode. The IMAGE_TYPE variable is either RAM or ROM and defines how the RedBoot image runs.

1. See http://sources.redhat.com/ecos/docs-latest/ref/startup-mode.html for more information on RedBoot startup modes.

The next command is

```
ecosconfig add intel_npe
```

This command adds the intel_npe package to the configuration database

The next command is

```
ecosconfig add ${BOARD}_npe
```

This adds the board specific npe package to the configuration database.

The next command is

```
ecosconfig tree
```

This creates the build tree using the target configuration. The CDL files are consulted and the includes are generated. These are all created under the build directory. An example is the version defines that are configured in the file `packages/hal/arm/xscale/ixdp425/current/cdl/hal_arm_xscale_ixdp425.cdl`. When the tree is generated, the CDL is consulted and the defines are written into the file `build/install/include/pkgconf/hal_arm_xscale_ixdp425.h`

The next command (should be obvious by now!)

```
make
```

The starts the RedBoot build and may take several minutes to complete.

B.3 build-redboot Script Source

```
#!/bin/bash
#
#
# Disclaimer
# ----------
# This script is distributed in the hope that it will be useful, but without any
# warantee, without even the implied warantee of merchantabilty or fitness for
# a particular purpose
#
#
# All copyrights are owned by their owners, unless specifically noted otherwise.
# Use of a term in this document should not be regarded as affecting the
# validity of any trademark or service mark.
#
# Naming of particular products or brands should not be seen as endorsements.
#
# You are strongly recommended to take a backup of your system before major
# installation and backups at regular intervals.
#
#
# Revision History
# -----------------
# 20030812 - Released for general use
#

#set +x

BUILD_ECOSCONFIG="y"
RBSRC="redboot-intel-xscale-030618"
NPE_EPK_FILE="npe-1.1.epk"
# change IMAGE_TYPE to ROM for images that you load into flash
```

```
# Or use RAM for images that are for testing or update
IMAGE_TYPE="ROM"


if [ -n "$1" ]; then
    case $1 in
        "ixdp425" | "grg" | "prpmc1100")
            BOARD=$1
        ;;

        *)
            echo "$0: error"
            echo "Invalid build target"
            echo "If specified, board must be ixdp425, grg or prpmc1100"
            exit 1
        ;;
    esac
else
    BOARD="ixdp425"
fi

echo ""
echo "-----------------------------------"
echo "Building RedBoot for the ${BOARD}"
echo "-----------------------------------"
echo ""

if [ ! -d ./${RBSRC} ]; then
    echo "RedBoot source not installed in ${RBSRC}"
fi


cd ./${RBSRC}

export PATH=/opt/redhat/xscale/bin:${PATH}
export TOPDIR=`pwd`
export ECOS_REPOSITORY=${TOPDIR}/packages
export VERSION=current
echo "TOPDIR          = ${TOPDIR}"
echo "ECOS_REPOSITORY = ${ECOS_REPOSITORY}"


chmod +x ${ECOS_REPOSITORY}/ecosadmin.tcl
${ECOS_REPOSITORY}/ecosadmin.tcl add ../../${NPE_EPK_FILE}

if [ -d ${TOPDIR}/build ]; then
    echo " Cleaning up old build..."
    rm -rf ${TOPDIR}/build
fi

mkdir ${TOPDIR}/build
cd ${TOPDIR}/build


#-------------------------------------------------------------
# you only need to build ecosconfig once for your platform
# Once it is built , move it to the tool chain bin
# directory, and set  BUILD_ECOSCONFIG ="n"
#
# This section builds it and sets it up for use by the
# rest of the script.  One you build and move to bin
# ecoscfg can be found in the PATH, so ECOSCFG_PATH
# is set to empty

if [ ${BUILD_ECOSCONFIG} = "y" ]; then
    echo ""
    echo "--------------------------------"
```

```
    echo "Building and setup of ecosconfig..."
    echo "-----------------------------------"
    echo ""

    ../host/configure
    make
    cp tools/configtool/standalone/common/ecosconfig .
    ECOSCFG_PATH="./"

else
    ECOSCFG_PATH=""
fi

echo ""
echo "-----------------------------------"
echo " Building RedBoot..."
echo "-----------------------------------"
echo ""

${ECOSCFG_PATH}ecosconfig new ${BOARD} redboot
${ECOSCFG_PATH}ecosconfig import ${ECOS_REPOSITORY}/hal/arm/xscale/${BOARD}/
current/misc/redboot_${IMAGE_TYPE}.ecm
${ECOSCFG_PATH}ecosconfig add intel_npe
${ECOSCFG_PATH}ecosconfig add ${BOARD}_npe
${ECOSCFG_PATH}ecosconfig tree

make


echo ""
echo "------------------------------------------------------------"
echo " RedBoot Build compelete -  for ${IMAGE_TYPE} based images"
echo " Binaries for the ${BOARD} are available at: "
echo "  ${TOPDIR}/build/install/bin "
echo ""
ls ${TOPDIR}/build/install/bin -l
echo ""
```

Appendix C Makefile

This makefile is an example method of building RedBoot for the IXDP425 / IXCDP1100 platform. This could be extended to other platforms by changing the BOARD. The makefile ‘install’ target will set up and install the toolchain.

```
make install

make ecosconfig

make
```

Note: When cutting and pasting the following example, remember that ‘make’ uses tabs for indenting the target recipe lines.

```
#
# This script is distributed in the hope that it will be useful, but without any
# warantee, without even the implied warantee of merchantabilty or fitness for
# a particular purpose
#
# All copyrights are owned by their owners, unless specifically noted otherwise.
# Use of a term in this document should not be regarded as affecting the
# validity of any trademark or service mark.
#
export TOPDIR=$(CURDIR)
PROOT=/opt/redhat/xscale-030422
PATH +=:${PROOT}/H-i686-pc-linux-gnulibc2.2/bin
BOARD=ixpd425
NPE_EPK_FILE=npe-1.1.epk
#INFOPATH=${PROOT}/info
export PATH
export ECOS_REPOSITORY=${TOPDIR}/packages
#export VERSION=current


all: npe
    cd ${TOPDIR}/build; \
    ./ecosconfig new ${BOARD} redboot; \
    ./ecosconfig import ${ECOS_REPOSITORY}/hal/arm/xscale/ixdp425/current/misc/
redboot_ROM.ecm; \
    ./ecosconfig add intel_npe;\
    ./ecosconfig add ${BOARD}_npe;\
    ./ecosconfig tree; \
    make

npe:
    chmod +x ${ECOS_REPOSITORY}/ecosadmin.tcl
    ${ECOS_REPOSITORY}/ecosadmin.tcl add ../${NPE_EPK_FILE}

ecosconfig:
    rm -rf ${TOPDIR}/build
    mkdir ${TOPDIR}/build
    cd ${TOPDIR}/build; \
    ../host/configure; \
    make; \
    cp tools/configtool/standalone/common/ecosconfig .


# tool chain installation
install:
    echo "Installing tool chain for RedBoot..."
    cd ../ ; \
```

```
uncompress < i686-pc-linux-gnulibc2.2-x-xscale-elf.tar.Z | tar xpf - Install ;\
    ./Install --tape=i686-pc-linux-gnulibc2.2-x-xscale-elf.tar.Z binaries
echo "Installing tool chain for RedBoot complete."
```