

Getting Started with eCosTM

i386 PC edition May 2001

Copying terms

The contents of this manual are subject to the Red Hat eCos Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.redhat.com/

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is eCos - Embedded Configurable Operating System, released September 30, 1998.

The Initial Developer of the Original Code is Red Hat. Portions created by Red Hat are Copyright© 1998, 1999, 2000, 2001 Red Hat, Inc. All Rights Reserved.

Trademarks

 $Java^{TM}$, Sun, and $Solaris^{TM}$ are trademarks and registered trademarks of Sun Microsystems, Inc.

SPARC is a registered trademark of SPARC International, Inc.

UNIXTM is a trademark of The Open Group.

Microsoft, Windows NT, Windows 95, Windows 98 and Windows 2000 are registered trademarks of Microsoft Corporation.

Linux is a registered trademark of Linus Torvalds.

Intel is a registered trademark of Intel Corporation.

eCos™ is a trademark of Red Hat, Inc.

Red Hat is a registered trademark of Red Hat, Inc.

300-400-1010049-03

Contents

Getting Started with eCos TM	1
Copying terms	2
Trademarks	2
Foreword	7
Documentation Roadmap	12
Getting Started with eCos	12
eCos User's Guide	12
eCos Reference Manual	13
Part I: Release Notes	15
Notation and Conventions	16
GDB and GCC Command Notation	16
Directory and File System Conventions	17
Version Conventions	17
Release Overview	18
Hardware Abstraction	18
Embedded Kernel	19
Configurability	19
μITRON and Other Operating Systems	
ISO C Library	
Serial Device Drivers	
Monitor Image	21
Tests and Examples	21
GNU Tools and their Documentation	
eCos Documentation	22
Package Contents	23
eCos Net Release	23

	System Requirements	24
	Required	24
	Recommended	25
	Reporting Problems	26
	How to Report Problems	26
P	Part II: Installation Guide	30
	Software Installation	31
	Software Installation on Windows	31
	Software Installation on UNIX	32
	Target Setup	36
	Connecting To A Target Via Serial	36
	Connecting To A Target Via Ethernet	37
	Connecting To A Simulator Target	37
	Connecting To A Synthetic Target	37
	i386 PC Hardware Setup	39
	i386/Linux Synthetic Target Setup	39
	Running Applications on the Target	
P	Part III: Programming Tutorial	43
	Programming with eCos	44
	Configuring and Building eCos from Source	
	eCos Start-up Configurations	
	Using the Configuration Tool on Windows	
	Using ecosconfig on UNIX	
	Architectural Notes	
	Test Suites	66
	Using the Configuration Tool	67
	Using the command line	
	Testing Filters	68
	Building and Running Sample Applications	
	eCos Hello World	69
	A Sample Program with Two Threads	70
	More Features — Clocks and Alarm Handlers	74
	A Sample Program with Alarms	74

Appendixes				
Appendix 1: Real-time characterization	80			
Sample numbers:	80			
Appendix 2: eCos Licensing				
RED HAT ECOS PUBLIC LICENSE				
Version 1.1	83			
Appendix 3: The eCos Copyright Assignment Form, Re	vision 1.190			
Index	95			

_		~	C . 1	* . 1	
h	n	Getting	Started	with	AL OS

Foreword

Welcome to the latest release of Red Hat eCos(TM) - the Embedded Configurable Operating System.

What's New?

In the fourth major public release of eCos, starting with version 1.4, we have added a wealth of new features, enhancements, and have further extended the target platform coverage.

Major new elements include:

- ⁿ Package management that supports the extension of eCos functionality via third party add-on packages.
- A standardized configuration save file format that is human readable and editable, and compatible between both GUI and command line configuration tools.
- ⁿ Enhanced web based help and component documentation system integrated into the GUI configuration tool.
- The Component Definition Language (CDL) has been radically revised and has now been implemented as a TCL extension for maximum flexibility. CDL is now fully documented in the Component Writers Guide.
- Template support for straightforward control of multiple configuration elements, which can be used to provide easy access to standard eCos configurations such as a debug stub boot ROM.
- Best of all, the source of the new configuration tools and underlying libCDL technology has been open sourced under the GNU Public License (GPL).

A companion beta version of the eCos TCP/IP stack has been released in conjunction with this release. The stack is derived from the OpenBSD source base and provides UDP, TCP, ICMP, BOOTP and DHCP protocol support on an IPv4 standards base. Device driver support for Cirrus Logic EP72xx evaluation boards, Motorola MBX, and StrongARM EBSA285 with Intel 82559 is included. The stack and ethernet core support are distributed as configurable eCos packages. Platform ethernet device drivers are distributed with the associated platform HAL, but are naturally not useful without the ethernet core support package.

A PCI bus support library has also been added that provides generic PCI bus based device initialization, discovery, and configuration. The library has been ported to both the VR4300 DDB-VRC4373 and StrongARM EBSA285 development boards.

eCos also contains support for the POSIX Specification (ISO/IEC 9945-1)[POSIX]. This support follows EL/IX level 1 in the functionality supplied [ELIX].

POSIX support is divided between the POSIX and the FILEIO packages. The POSIX package provides support for threads, signals, synchronization, timers and message queues. The FILEIO package provides support for file and device I/O. The two packages may be used together or separately, depending on configuration.

Also supplied with eCos is RedBoot; the standard bootstrap and debugging environment for embedded systems from Red Hat. It provides a wide set of tools for downloading and executing programs on embedded target systems, as well as tools for manipulating the target system's environment.

RedBoot's capabilities include:

- Serial and network (Ethernet) based debugging
- n FLASH management
- ⁿ Simple command line interface, available via serial or Ethernet
- n Configurable and extensible, specifically adapted to the target environment

New architectures and platforms added in this release include:

- n ARM Thumb
- n ARM9
- n Cirrus Logic CL-PS7111 and EP72xx
- n Cogent CMA222 and CMA230 ARM boards
- n Hitachi SH3
- n Intel StrongARM
- n Intel x86 PC
- n Matsushita AM33
- n Motorola MBX evaluation board
- n NEC MIPS VR4300
- n NEC CEB-V850SA1

For further details of all the changes see the NEWS file in the eCos sources.

Red Hat is dedicated to continued enhancement and maintenance of the eCos system. Developers can look forward to upcoming releases that further expand the architectural and board coverage, extend the functionality of the TCP/IP stack, add a Linux version of the GUI configuration tool, and add major new features such as a Linux/Posix compatibility layer based on the upcoming EL/IX standard - see

http://sources.redhat.com/elix/

for more details.

eCos in a Nutshell

eCos is an open source, configurable, portable, and royalty-free embedded real-time operating system. The following text expands on these core aspects that define eCos.

eCos is provided as an open source runtime system supported by the Red Hat GNUPro and GNU open source development tools. Developers have full and unfettered access to all aspects of the runtime system. No parts of it are proprietary or hidden, and you are at liberty to examine, add to, and modify the code as you deem necessary. These rights are granted to you and protected by the Red Hat eCos Public License (RHEPL). It also grants you the right to freely develop and distribute applications based on eCos. You are not expected or required to make your embedded applications or any additional components that you develop freely available, although we do require that you make publicly available any modifications to the eCos code itself. Red Hat of course welcomes all contributions back to eCos such as board ports, device drivers and other components, as this helps the growth and development of eCos, and is of benefit to the entire eCos community.

One of the key technological innovations in eCos is our configuration system. The configuration system allows the application writer to impose their requirements on the run-time components, both in terms of their functionality and implementation, whereas traditionally the operating system has constrained the application's own implementation. Essentially, this enables eCos developers to create their own application-specific operating system and makes eCos suitable for a wide range of embedded uses. Configuration also ensures that the resource footprint of eCos is minimized as all unnecessary functionality and features are removed. The configuration system also presents eCos as a component architecture. This provides a standardized mechanism for component suppliers to extend the functionality of eCos and allows applications to be built from a wide set of optional configurable run-time components. Components can be provided from a variety of sources including: the standard eCos release; commercial third party developers; open source contributors; or additional optional components from Red Hat.

The royalty-free nature of eCos means that you can develop and deploy your application using the standard eCos release without incurring any royalty charges. In addition, there are no up-front license charges for the eCos runtime source code and associated tools. We provide, without charge, everything necessary for basic embedded applications development.

eCos is designed to be portable to a wide range of target architectures and target platforms including 16, 32, and 64 bit architectures, MPUs, MCUs and DSPs. The eCos kernel, libraries and runtime components are layered on the Hardware Abstraction Layer (HAL), and thus will run on any target once the HAL and relevant device drivers have been ported to the target's processor architecture and board. Currently eCos supports a large range of different target architectures (ARM, Hitachi SH3, Intel x86, MIPS, Matsushita AM3x, Intel StrongARM, NEC V850, Motorola PowerPC, and SPARC) including many of the popular variants of these architectures and evaluation boards. Many new ports are in development and will be released as they become available.

eCos has been designed to support applications with real-time requirements, providing features such as full preemptability, minimal interrupt latencies, and all the necessary synchronization primitives, scheduling policies, and interrupt handling mechanisms needed for these type of applications. eCos also provides all the functionality required for general embedded application support including device drivers, memory management, exception handling, C, math libraries, etc. In addition to runtime support, the eCos system includes all the tools necessary to develop embedded applications, including eCos software configuration and build tools, and GNU based compilers, assemblers, linkers, debuggers, and simulators.

To get the most out of eCos you should visit the eCos open source developers site:

http://sources.redhat.com/ecos/

The site is dedicated to the eCos developer community and contains a rich set of resources including news, FAQ, online documentation, installation guide, discussion and announcement mailing lists, online problem report form, and runtime and development tools downloads. We also support anonymous CVS and WEBCVS access to provide you with direct access to the very latest eCos source base. Complementing the open source developers site is an eCos product site, featuring news, press releases, details of our commercial engineering and support services, products, and third party partner offerings. This is located at

http://www.redhat.com/embedded/technologies/ecos/

We have released eCos as open source software because we believe that this is the most effective software development model, and that it provides the greatest benefit to the embedded developer community as a whole. As part of this endeavor, we seek the input and participation of eCos developers in its continuing evolution. Participation can take many forms including:

- providing us with feedback on how eCos might be made more useful to you by taking part in the ongoing mailing list discussions and by submitting problem reports covering bugs, documentation issues, and missing features
- n contributing bug fixes and enhancement patches
- n contributing new code including device drivers, board ports, libraries, and other runtime components

Our long term aim is to make eCos a rich and ubiquitous standard infrastructure for the development of deeply embedded applications. This will be achieved in part by Red Hat's own efforts, but also with the assistance of the eCos developer community cooperating to improve eCos for all. I would like to take this opportunity to extend our thanks to the many eCos developers who have already contributed feedback, ideas, patches, and code that have augmented and improved this release.

On behalf of the eCos team, welcome to the eCos developer community.

Paul Beskeen, Director of Engineering, eCos November 2000



Documentation Roadmap

Getting Started with eCos

Release Notes

Description of this release.

Installation Guide

Hardware and software installation instructions, including instructions on how to execute some prebuilt tests to verify the installation.

Programming Tutorial

A tutorial that gets you started running programs with **eCos**.

Appendixes

Extra information about the licensing terms for **eCos**.

eCos User's Guide

The eCos Configuration Tool

A description of all features of the Configuration Tool.

Programming concepts and techniques

An explanation of the **eCos** programming cycle, and a description of some debugging facilities that **eCos** offers.

Configuration and the Package Repository

Information on how to configure **eCos** manually, including a reference on the ecosconfig command, memory layouts, and information on how to manage a package repository using the **eCos Package Administration Tool**.

eCos Reference Manual

Preliminaries

An overview of the **eCos** kernel and configurability system.

Kernel APIs

In-depth description of **eCos**'s native C kernel API, the μ ITRON API, the ISO standard C library, and the **eCos** Hardware Abstraction Layer (HAL). Important considerations are given for programming the **eCos** kernel. The semantics for each kernel function are described, including how they are affected by configuration.

eCos Device Drivers

A description of the philosophy behind **eCos** device drivers, as well as a presentation of the C language API for using the current device drivers.

The ISO Standard C and Math Libraries

eCos comes with an implementation of the ISO C library specification. This section gives details about the implementation, lists the few functions that are not yet implemented, and gives a complete reference for configuring the C library.

Part I: Release Notes

This release of eCos supports the following architectures:

- _n i386 PC and compatibles
- n Linux i386—synthetic Linux target

and supports the following target platforms:

- n Standard PC motherboard
- ⁿ Linux (i386) synthetic Linux target (i386 and compatibles)

This release also supports the following host operating systems:

- n UNIX[™]. Redhat Linux[™] and Solaris[™] are the only tested UNIX variants.
- ⁿ Microsoft® Windows NT®, Windows 95®, Windows 98®, Windows 2000®. Note that support for Windows 95, 98 and 2000 is still "beta".

2

Notation and Conventions

Since there are many supported target architectures, notation conventions are used in this manual to avoid repeating instructions that are very similar.

GDB and GCC Command Notation

Cross-development commands like gcc and gdb will be shown without prefixed information about the platform for which you are cross-compiling. You need to add the necessary prefix before you execute the commands, so instead of simply typing gcc and gdb, as illustrated in the various example in this manual, use:

i386-elf-gcc and i386-elf-gdb for standard PC i686-pc-linux-gnu-gcc and i686-pc-linux-gnu-gdb for Synthetic Linux.

Note that the GCC cross compiler generates executable files with the .exe suffix on Windows, but not on UNIX. The suffix .exe will be omitted from executable file names, so you will see hello instead of hello.exe.

Directory and File System Conventions

The default directory for installing eCos on Windows (usually C:/Program Files/Red Hat/eCos) is different from that on UNIX (usually /usr/local/ecos-1.4.x). Since many command line examples in the tutorials use these paths, this default (base) directory will be cited as *BASE DIR*.

Windows and UNIX have a similar file system syntax, but the MS-DOS command interpreter on Windows uses the backslash character (\) as a path separator, while UNIX and POSIX shells (including the Cygwin bash shell for windows) use the forward slash (/).

This document will use the POSIX shell convention of forward slashes throughout.

Version Conventions

This manual outlines the features of eCos version 1.4.x. The initial release version was 1.4, and additional 1.4 releases will incorporate one or more additional numbers, represented in this manual by "x". Please note the exact version number of the version that you are using, because it is incorporated in certain file paths.

3

Release Overview

The Embedded Configurable Operating System (eCos) software consists of a set of tools and a run-time environment for developing embedded applications. It is a configurable, open source framework that allows you to build a run-time system that closely matching the needs of your application.

eCos is aimed at embedded software developers who use architectures with tight memory constraints, who want a portable framework for developing their applications.

If you want to start programming eCos immediately, see "Part II: Installation Guide" on page 30 and "Part III: Programming Tutorial" on page 43.

Hardware Abstraction

eCos includes a Hardware Abstraction Layer (HAL) that hides the specific features of each supported CPU and platform, so that the kernel and other run-time components can be implemented in a portable fashion.

The eCos HAL has now been ported to numerous architectures, and to one synthetic target, Linux i386. Notes on porting the HAL to new platforms are provided in the eCos Reference Manual under Kernel porting notes in The eCos Hardware Abstraction Layer section.

Embedded Kernel

The core of eCos is a full-featured, flexible, and configurable embedded kernel.

The kernel provides, among other features, multi-threading, a choice of schedulers, a full set of synchronization primitives, memory allocation primitives, and thread manipulation functions (see the *eCos Reference Manual* for the full kernel API).

The kernel is designed such that some parts of it can be changed or replaced without affecting other kernel components.

The following is a partial list of kernel features:

- n choice of memory allocation algorithm
- n choice of scheduling algorithm
- n a rich set of synchronization primitives
- n timers, counters, and alarms
- n interrupt handling
- n exception handling
- n cache control
- n thread support
- n kernel support for multi-threaded debugging with GDB
- n trace buffers
- ⁿ infrastructure and instrumentation

The kernel API and configuration are described in the eCos Reference Manual.

Configurability

The eCos kernel and other components can be configured in great detail at compile time, which avoids the need to add unwanted code to the library to be linked with your application code. There is no performance penalty for configuration.

Configuration is fine-grained, so that very small details of eCos' behavior can be tuned by selecting different configuration options.

eCos is organized as a component architecture, with a language to describe the constraints between the components and individual configuration options. These constraints are necessary to resolve inconsistent configurations, such as disabling the code which handles the real-time clock, while enabling per-thread timers.

The designer of a component or general-purpose library should write configurable code using a component definition language (CDL). Once that has been done there is no additional burden on the end user (i.e. an embedded systems developer), who will be able to use eCos' graphical **Configuration Tool** to configure the kernel and basic libraries without needing to understand how the configuration infrastructure works.

A tutorial explaining how to configure eCos is located in "Configuring and Building eCos from Source" on page 49. The eCos *User's Guide* has more detailed information on running the **Configuration Tool** and CDL.

μITRON and Other Operating Systems

eCos' configurability is the key to simulating different operating systems by using compatibility layers on top of eCos' kernel, because the semantics of basic kernel functions can be configured to match the semantics of other operating systems.

The specification for the μ ITRON operating system has been implemented on top of eCos. μ ITRON is configured by selecting appropriate options in the kernel (a real-time clock, the mlqueue scheduler, and no timeslicing); and writing a thin layer to map the μ ITRON system calls.

The μ ITRON port implements the complete μ ITRON 3.02 "Standard functionality" (level S) specification, as well as some of the "Extended" (level E) functions. The μ ITRON implementation is described in more detail in the *eCos Reference Manual*.

ISO C Library

The ISO C and math library shipped with eCos was written to be configurable and tightly integrated with the kernel and the HAL.

By carefully selecting configuration options in the C library, you can significantly reduce the size of the final executable image.

Serial Device Drivers

eCos provides serial device drivers for all supported eCos platforms, with the exception of the i386 Linux synthetic target and most simulator platforms. The serial drivers provide an API (documented in the *eCos Reference Manual*) to control serial ports directly. The standard I/O library can be configured to use them as a transport layer.

Monitor Image

RedBoot

The new standard bootstrap and debugging environment for Red Hat embedded systems is RedBoot, a configurable and extensible command line application which provides serial and network debugging and FLASH management. Based on the eCos HAL, RedBoot supports eCos, GNUPro applications and embedded Linux systems on a wide range of platforms, including ARM, MIPS, MN10300, PowerPC, SHx, v850 and x86.

RedBoot is recommended for the i386 PC, ARM PID/EPI Dev7/EPI Dev9, ARM Evaluator-7T, Cirrus Logic EP72xx, Intel StrongARM, Intel XScale, PMC-Sierra Ocelot, and Motorola PPC MBX860 platforms.

Redboot provides a GDB stub allowing debugging with the GDB debugger, and is supplied as a standalone application with the eCos distribution. See also the *RedBoot User's Guide*.

No monitor image is required for the synthetic Linux target.

Tests and Examples

Test suites are included for every portion of eCos shipped in this release. These are brief programs that test the behavior of most system calls and libraries in eCos. "Test Suites" on page 66 describes how to build and run these test suites.

The last chapters of "Part III: Programming Tutorial" on page 43 provide examples that guide you the steps required for running eCos applications, starting from a "Hello world" program and then moving on to more complex programs that use additional kernel features.

GNU Tools and their Documentation

Red Hat's GNUPro Toolkit, which includes the GCC and G++ compilers and the GDB debugger, is needed to build eCos applications. It is bundled with the CD-ROM distribution of the eCos Developer's Kit, and is also available on the net at

http://sources.redhat.com/ecos/

Online HTML versions of the full GNUPro documentation are included with eCos, as well as a specific GNUPro tools reference guide for your hardware architecture, customized for use with eCos. The full GNUPro documentation can also be found on the web at:

http://www.redhat.com/support/manuals/gnupro.html

NOTE The Linux synthetic i386 target is an exception, as there is (currently) no GNUPro manual. However, the GNUPro source archive contains documentation for the tools. This documentation is usually also included as part of a default Red Hat Linux installation, accessible with the **info** program.

eCos Documentation

The eCos documentation set includes *Getting Started with eCos*, the *eCos User's Guide*, the *eCos Reference Manual*, and a *GNUPro Reference Manual* for your specific architecture.

For users of the eCos Net releases, these are available online in HTML format at

http://sources.redhat.com/ecos/

4

Package Contents

eCos Net Release

The eCos Net Release consists of the archive files for GNUPro and eCos, which are located on the Red Hat eCos web site:

http://sources.redhat.com/ecos/

The eCos Net Release, because it is digitally distributed only, does not provide images for the various development boards. However, the images for the supported hardware platforms are included in the distribution, so you can burn your own Flash ICs to work with eCos.

HTML versions of the GNUPro and eCos manuals are included in the distribution, and are also available online.

5

System Requirements

Required

- n Standard Intel™ architecture PC running Linux (tested on Red Hat Linux distributions 5.0-7.0), and English or Japanese versions of Microsoft Windows NT version 4.0 (service pack 3 or above must be installed), Windows 95, Windows 98, or Windows 2000. Other versions of Red Hat distributions, or Linux distributions from other vendors should work as well.
- Windows NT users must install Internet Explorer 4.0 or later, since this will ensure correct operation of the **Configuration Tool**.

 SunTM workstation running Solaris 2.5.1 or later for the SPARCTM.

 Support for any platform except for Windows NT 4.0, Solaris 2.5.1 and Linux is beta. In particular, rebuilding the GNUPro compiler toolchain is only supported and tested on Windows NT 4.0, Solaris 2.5.1, and Red Hat Linux.
- Enough disk space for the installed distribution. The eCos installation process will detail the various components of eCos and the GNUPro toolkit that can be installed, and their disk space requirements.
- 64MB of RAM and a 350MHz or faster Pentium processor.
 If you are downloading the eCos Net Release distribution from Red Hat's sources.redhat.com site, you will also need space to store that image and to compile GNUPro and eCos from source.

If you will be using an x86 board, you will also need:

A standard PC motherboard with an i386 or better processor and a 3.5" floppy disk drive. Optionally, a color text-mode-only display card with monitor or keyboard may also be provided. Any standard PC will provide all the hardware required to run eCos.

If serial debugging is to be used:

ⁿ A null modem cable to connect the COM1 port on the target motherboard to a serial port on the host system.

If network debugging is to be used:

- Suitable network interface card on the development PC and connecting cables
- An Intel i82559 based PCI ethernet card (for example an Intel EtherExpress Pro 10/100) may be installed for use by the eCos network stack or for network debugging.

If you will be using the Linux synthetic target, you will also need:

ⁿ An x86 PC with an installed Linux distribution (tested with Red Hat Linux distributions 5.0 - 7.0).

Recommended

ⁿ A Pentium II computer and 64MB or more of RAM are recommended for best build performance.

The system has been tested only in the recommended configuration above, although other configurations are expected to work.

6

Reporting Problems

Reporting bugs and other problems is very important: it allows Red Hat to solve your problem quickly, and improves the eCos product. The effort you make in reporting problems is appreciated.

To submit a problem report, please use the web interface. If you have a CD distribution of the eCos Developer's Kit, you should use the address:

http://support.cygnus.com

You will need a login name and an ID, provided by your administrator.

If you are using the eCos Net release you should use the address http://sources.redhat.com/ecos/problemreport.html

Known Bugs in eCos and GNUPro

Before filing bug reports, however, please read the README provided with this release. It describes known problems and possible workarounds in eCos or with the GNUPro Toolkit. The file is at the base of the distribution.

How to Report Problems

For documentation discussing methods of reporting on, editing and querying, see the following *Accessing Red Hat Web Support to Report Problems*, or *Additional Options* in this chapter.

This documentation serves only as a guide and it is not meant to supercede the Help documentation on the Web Support site. We have tried to make our software as trouble-free as possible. If you do encounter problems, we'd like to diagnose and fix the problem as quickly as possible.

Accessing Red Hat Web Support to Report Problems

If you have a CD distribution, use the following instructions to access the Red Hat Support website.

- Use the following URL in your web browser's address or location dialog box.

 http://support.cygnus.com
- Click on the Case Management System icon, enter your ID and password, and the Welcome page will be displayed.
 - Access the Welcome page at any time by using the Welcome link (in the navigation bar on the left side of each Web Support page).
 - If you have the CD distribution, your details will have been entered in the database, and will be displayed on the Welcome page. If you wish to alter these details, select the Profile link in the navigation bar on the left side of the page.
- Use the links included in the navigation bar on the left side of the page to perform any of the following Red Hat Web Support activities.
 - New Case (see Submitting a support request, and the Red Hat Support website)
 - guery Case (see Additional options, and the Red Hat Support website)
 - r Add Notes (see Additional options, and the Red Hat Support website)
 - r Find Solutions (see Additional options, and the Red Hat Support website)
 - r Profile (see Additional options, and the Red Hat Support website)
 - Fig. Help documentation see Additional options, and the Red Hat Support website)
 - r Close Case (see Additional options, and the Red Hat Support website)

Submitting a Support Request

Use the following instructions to submit a support request, once you have a valid ID established.

- $_{\mbox{\tiny n}}$ Click on New Case to create a new reported problem case.
 - The New Case page allows you to complete the creation of a new case. If there is more than one site, select the site relating to your problem.
 - r Click on Use This Site ID button to display a list of the relevant products.

- Select a product from the list and then click on the Create Case for Selected Product button.
 - Each customer has a valid list of parts of Red Hat products for which they can submit problem reports. These components are part of the Web Support database.
- Type a brief description of the case in the Case Title field. You can enter up to 80 characters in this field.
- Select a case type from the Type drop-down menu that best describes the case.
- Select a customer severity level from the Severity drop-down menu that best describes how severe you view this problem.
- Select a case priority level from the Priority drop-down menu that best describes the priority of this case to Red Hat.
- Type a complete description of your case in the Problem Description field.
- ^r Use the scrollbars to scroll text in this field. You can add up to 30 kilobytes of text in this field.
- Click on the Create Case button at the bottom of the page to create the case in the Red Hat Web Support database. Alternatively, clear the input fields on the New Case page, using the Clear button.

After you create your case, the Case Details page displays, which includes the Case ID number that the support database assigns to your case.

To create a new case for a different site and/or part, click the New Case link in the navigation bar; then use the previous instructions.

Additional Options

The following documentation discusses the other features for the Red Hat Web Support site. Red Hat has a database to help in determining when problems developed, tracking the problems case from their first report through analysis and resolution. The database can also be used for correlation with other products as well as to other related problems.

- n Click on Query Case to find an existing problem case in our database.
 - You may examine problem cases in the Red Hat Web Support database, searching by solution ID or by entering keywords and/or a key phrase. There are options on this page enabling you to control how your search works.
 - At this point, view a problem case's details, check its status, add notes or close a problem.
- ⁿ Click on Add Notes to add additional data to an existing case in our database.

- ⁿ Click on Find Solutions to search for problem solutions in the database. The search will provide a list of the current problem cases in the Red Hat Web Support database.
- Click on Profile to change your profile information and/or your Web Support password in our database. A Profile page wil be displayed.
- Click on Help for questions about using the Web Support page. The online help documentation for the Web Support site supercedes this guide; it is not meant to supercede the more updated Help documentation for the Web Support site.
- Click on Close Case link to close a case. Closing a case brings the problem to its resolution.

Updating your profile

Clicking on Profile allows you to enter the following details.

- n Your contact name
- ⁿ The primary phone number where Red Hat Support can contact you
- _n FAX number Red Hat Support can use to send you information
- n Your e-mail address
- Your site ID, used to identify your primary site in the Web Support database (a Red Hat representative will provide this information)
- n Your site name

Part II: Installation Guide

7

Software Installation

Software Installation on Windows

If you have a CD distribution of the eCos Developer's Kit, you have received the eCos software and its supporting utilities on a single CD-ROM for installation on a PC-compatible computer running Windows NT 4.0, Windows 95, Windows 98 or Windows 2000. If you use NT you must apply the NT 4.0 Service Pack 3 or above before installing eCos. Support is only for Windows NT 4.0. Installations on other Windows platforms are beta.

The following components are provided on the eCos CD-ROM:

- n eCos source code
- n Prebuilt eCos libraries and tests
- n eCos documentation
- n Red Hat GNUPro compiler toolchain for eCos source code compilation
- ⁿ Red Hat Cygwin environment: this product provides a POSIX compatibility layer on top of Windows NT, and supports the GNUPro tools on Windows NT.
- Documentation for the GNUPro tools, including a Reference Manual for the particular evaluation board being used to run eCos.

If you have obtained the Net release of eCos for Windows, you will have the distribution in a self-extracting archive. Apart from the difference in medium, the installation procedure for eCos itself will be the same as for the CD-ROM-based distribution.

The software installation process involves a number of installation utilities. Some familiarity with Windows is assumed.

- 1. Invoke the file Setup.exe on the CD-ROM. This will start the installation procedure. If you have the autorun feature enabled, Windows will run Setup.exe automatically when the CD-ROM is inserted into the drive.
- 2. The setup program will offer to install the GNU user tools. Click **OK**.
- 3. You will be prompted for a file path in which to install the GNU user tools. The default will be in the /cygnus/gnupro/i686-cygwin32/i686-cygwin32 hierarchy (usually on drive C). It will then offer to install the source code and documentation for the GNU user tools. It is recommended that you install the documentation, but not the source code, unless you are interested in modifying or recompiling the GNU user tools.
- **4.** At this point the setup program will begin installing eCos. Click **OK**.
- 5. The default path offered for eCos installation will be in the /Program Files/Red Hat hierarchy (usually on drive C). You may change this path, and indeed you will need to change it if that partition does not have sufficient free disk space available. It is recommended that you accept the default selection of software components for installation.
- **6.** You will be asked to select the program folder under which the eCos menu items will be placed. The default folder name is Red Hat eCos.
- 7. The installation should finish normally, offering to show you the README file that contains any last minute information and a list of known problems detected after this document was printed. Once the installation is finished, you can start eCos or view the online documentation by selecting Start -> Programs -> Red Hat eCos, and then choosing an option within this folder, e.g Configuration Tool, Package Administration Tool, etc.

At this point you are ready to configure and build a customized eCos kernel as described in "Configuring and Building eCos from Source" on page 49.

NOTE The order of directories in the PATH is very important, and build failures may result if the PATH is not set correctly. If you are having difficulties in building eCos, please make sure you have set the PATH exactly as above.

Software Installation on UNIX

Installation and build instructions for the eCos Net release are available on the Red Hat eCos web site

http://sources.redhat.com/ecos/

Installing the eCos Developer's Kit under Linux

Users of the eCos Developer's Kit under Red Hat Linux should use the following instructions, for most of which you will normally need to be the root user.

1. The CD-ROM must be "mounted" before installation can proceed. Execute the command:

```
# mount /dev/cdrom/ /mnt/cdrom
```

Install the eCos repository from the RPM file ecos14x.rpm (where x or xx are final digits of the current version number), located in the root directory of the CD-ROM using the following command:

```
# rpm -i /mnt/cdrom/ecos14x.rpm
```

Note that root privileges are required to perform this installation. On completion, the eCos repository may be found in the directory /opt/ecos/ecos-1.4.x.

2. Extract the eCos development tools from the compressed tar archive tool-bin.tgz, located in the root directory of the CD-ROM, using the following commands:

```
# mkdir /usr/cygnus
# cd /usr/cygnus
# qunzip -c < /mnt/cdrom/tool-bin.tqz | tar xvf -</pre>
```

3. On completion, the eCos development tools may be found in the directory /usr/cygnus/ecos-DEVTOOLSVERSION. The source code for the development tools may optionally be installed in the same way:

```
# gunzip -c < /mnt/cdrom/tool-src.tgz | tar xvf -</pre>
```

4. Add the eCos host tools and development tools to the front of your path. Under Linux, you should modify the PATH environment variable as follows. Using sh, ksh, or bash:

```
$ PATH=/opt/ecos/ecos-1.4.x/tools/bin:/usr/cygnus/\
DEVTOOLSVERSION/H-i686-pc-linux-gnu/bin:$PATH
$ export PATH
```

Using csh or tcsh: Note that csh also requires the shell command "rehash" after modifying the path for the path change to take effect.

```
$ setenv PATH /opt/ecos/ecos-1.4.x/tools/bin:/usr/cygnus/\
DEVTOOLSVERSION/H-i686-pc-linux-qnu/bin:$PATH
```

Set the ECOS_REPOSITORY environment variable as follows. Using sh, ksh or bash:

```
$ ECOS_REPOSITORY=/opt/ecos/ecos-1.4.x/packages
$ export ECOS_REPOSITORY
```

Using csh or tcsh:

```
$ setenv ECOS REPOSITORY /opt/ecos/ecos-1.4.x/packages
```

At this point you are ready to configure and build a customized eCos kernel as shown in "Configuring and Building eCos from Source" on page 49.

NOTE The order of directories in the PATH is very important, and build failures may result if the PATH is not set correctly. If you are having difficulties in building eCos, please make sure you have set the PATH exactly as above.

Installing the eCos Developer's Kit under Solaris

Users of the eCos Developer's Kit under Solaris should use the following instructions, which assume that the CD-ROM is available at /cdrom/cdrom0.

1. Extract the eCos repository from the compressed tar archive ecos14x.taz (where x or xx are the final digits of the version number), located in the root directory of the CD-ROM using the following commands:

```
# mkdir /usr/local
# cd /usr/local
# zcat < /cdrom/cdrom0/ecos14x.taz | tar xvf -</pre>
```

On completion, the eCos repository may be found in the directory $/usr/local/\end{var}$

2. Extract the eCos development tools from the compressed tar archive toolbin.taz, located in the root directory of the CD-ROM, using the following commands:

```
# mkdir /usr/cygnus
# cd /usr/cygnus
# zcat < /cdrom/cdrom0/tool-bin.taz | tar xvf -</pre>
```

On completion, the executable files of the eCos development tools may be found in the directory /usr/cygnus/ecos-DEVTOOLSVERSION/H-host-triplet/bin. The source code for the development tools may optionally be installed in the same way:

```
# zcat < /cdrom/cdrom0/tool-src.taz | tar xvf -</pre>
```

3. Add the eCos host tools, development tools and any native tools supporting the eCos build process to the front of your path. Under Solaris you should modify the PATH environment variable as follows.

Using sh, ksh, or bash:

```
$ PATH=/usr/local/ecos-1.4.x/tools/bin:/usr/xpg4/bin:\
/usr/ucb:$PATH
$ export PATH
```

Using csh or tcsh:

Note that csh also requires the shell command "rehash" after modifying the path for the path change to take effect.

```
% setenv PATH /usr/local/ecos-1.4.x/tools/bin:/usr/xpg4/bin:\
/usr/ucb:$PATH
```

4. Set the ECOS REPOSITORY environment variable as follows:

Using sh, ksh or bash:

```
$ ECOS_REPOSITORY=/usr/local/ecos-1.4.x/packages
$ export ECOS_REPOSITORY
Using csh or tcsh:
% setenv ECOS REPOSITORY /usr/local/ecos-1.4.x/packages
```

At this point you are ready to configure and build a customized eCos kernel as shown in "Configuring and Building eCos from Source" on page 49.

NOTE The order of directories in the PATH is very important, and build failures may result if the PATH is not set correctly. If you are having difficulties in building eCos, please make sure you have set the PATH exactly as above.



Target Setup

Connecting To A Target Via Serial

While eCos supports a variety of targets, communication with all the targets happens in one of four ways. These are descibed in general below.

The descriptions are followed by descriptions of each target, providing specific details of how to set up the target (if hardware) and the necessary communication information (such as baud rate for hardware targets, or special connection options for simulator targets).

Most targets will have eCos GDB stubs or CygMon installed. These normally wait for GDB to connect at 38400 baud, using 8 data bit, no parity bit and 1 stop-bit (no hardware flow control). Check the section for your target to ensure it uses this speed. If not, adjust the following instructions accordingly.

The following instructions depend on your having selected the appropriate serial port on the host. That is, the serial port which connects to the target's (primary) serial port. On Linux this could be /dev/ttyso, while the same port on Windows would be named COM1, or /dev/ttya on Solaris. Substitute the proper serial port name in the below.

Connect to the target by issuing the following commands in GDB console mode:

```
(gdb) set remotebaud 38400 (gdb) target remote /dev/ttyS0
```

In Insight, connect by opening the **File->Target Settings** window and enter:

Target: Remote/Serial Baud Rate: 38400 Port: /dev/ttyS0 Set other options according to preference, close the window and select **Run->Connect to target**.

Connecting To A Target Via Ethernet

Some targets allow GDB to connect via Ethernet - if so, it will be mentioned in the section describing the target. Substitute the target's assigned IP address or hostname for <hostname> in the following. The <port> is the TCP port which the eCos GDB stub or CygWin is listening on. It is also listed in the section describing the target.

Connect to the target by issuing the following command in GDB console mode:

```
(qdb) target remote <hostname>:<port>
```

In Insight, connect by opening the **File->Target Settings** window and enter:

```
Target: Remote/TCP
Hostname: <hostname>
Port: <port>
```

Set other options according to preference, close the window and select **Run->Connect to target**.

Connecting To A Simulator Target

GDB connects to all simulator targets using the same basic command, although each simulator may require additional options. These are listed in the section describing the target, and should be used when connecting.

Connect to the target by issuing the following command in GDB console mode:

```
(gdb) target sim [target specific options]
```

In Insight, connect by opening the **File->Target Settings** window and enter:

```
Target: Simulator
Options: [target specific options]
```

Set other options according to preference, close the window and select **Run->Connect to target**.

Connecting To A Synthetic Target

Synthetic targets are special in that the built tests and applications actually run as native applications on the host. This means that there is no target to connect to. The test or application can be run directly from the GDB console using:

```
(qdb) run
```

or from Insight by pressing the **Run** icon. There is therefore no need to connect to the target or download the application, so you should ignore GDB "target" and "load" commands in any instructions found in other places in the documentation.

i386 PC Hardware Setup

For information about setting up the PC to run with RedBoot, consult the RedBoot User's Guide. If using serial debugging, the serial line runs at 38400 baud 8-N-1 and should be connected to the debug host using a null modem cable. If ethernet debugging is required, an i82559 compatible network interface card, such as an Intel EtherExpress Pro 10/100, should be installed on the target PC and connected to the development PC running GDB. When RedBoot is configured appropriately to have an IP address set, then GDB will be able to debug directly over TCP/IP to the target PC.

i386/Linux Synthetic Target Setup

When building for the synthetic Linux target, the resulting binaries are native Linux applications with the HAL providing suitable bindings between the eCos kernel and the Linux kernel.

NOTE: Please be aware that the current implementation of the Linux synthetic target does not allow thread-aware debugging.

These Linux applications cannot be run on a Windows system. However, it is possible to write a similar HAL emulation for the Windows kernel if such a testing target is desired.

Tools

For the synthetic target, eCos relies on features not available in native compilers earlier than gcc-2.95.1. It also requires version 2.9.5 or later of the GNU linker. If you have gcc-2.95.1 or later and ld version 2.9.5 or later, then you do not need to build new tools. eCos does not support earlier versions. You can check the compiler version using gcc -v or egcs -v, and the linker version using 1d -v.

If you have native tools that are sufficiently recent for use with eCos, you should be aware that by default eCos assumes that the tools i686-pc-linux-gnu-gcc, i686-pc-linux-gnu-ar, i686-pc-linux-gnu-ld, and i686-pc-linux-gnu-objcopy are on your system and are the correct versions for use with eCos. But instead, you can tell eCos to use your native tools by editing the configuration value "Global command prefix" (CYGBLD_GLOBAL_COMMAND_PREFIX) in your eCos configuration. If left empty (i.e. set to the empty string) eCos will use your native tools when building.

If you have any difficulties, it is almost certainly easiest overall to rebuild the tools as described on:

http://sources.redhat.com/ecos/getstart.html

9

Running Applications on the Target

At this point you should have installed the eCos software on your system (see "Software Installation" on page 31), and connected to a hardware target (see "Target Setup" on page 36).

To verify both that a hardware target is properly set up, and that the GDB commands used to connect to the target (hardware, simulator or synthetic) work properly on your system, you will now be guided through downloading and executing a prebuilt eCos test. The procedure is exactly the same when you want to download and run applications or tests that you have built yourself.

On Windows you must have the bash command line interpreter running with some environment variables that are useful for eCos work. If you have purchased the eCos Developer's Kit, you can select this by selecting Start->Programs->Red Hat eCos>eCos Development Environment. If you are using the eCos Net Release, you should set the environment variables as shown in the GNUPro Toolkit Reference Manual. On Linux, simply open a new shell window.

You will need to change directory to the prebuilt tests that are provided in the eCos installation, as follows:

```
for the i386 PC target:
   $ cd BASE_DIR/prebuilt/pc/tests/kernel/v1_4_x/tests
for the i386-based Linux synthetic target:
   $ cd BASE_DIR/prebuilt/linux/tests/kernel/v1_4_x/tests
```

To execute the thread gdb test case on the desired target,

1. Run GDB in command line mode using the following command, remembering to substitute the appropriate name for the architecture's gdb, eg. <target>-gdb:

```
$ gdb -nw thread gdb
```

GDB will display a copyright banner and then display a prompt (gdb).

2. Connect to the target according to the instructions given earlier (in "Target Setup" on page 36) - via serial or ethernet to hardware targets, or directly, for simulator and synthetic targets.

Depending on the target type, you will be notified about a successful connection, and possibly see some output informing you of the current program counter of the target.

3. Download the test - effectively loading the test case executable into the memory of the target - by typing this command:

```
(qdb) load
```

Again, depending on the target, you may see some output describing how much data was downloaded, and at what speed.

4. Start the test case running. For hardware targets this is done with the continue command, while run must be used on simulators and synthetic targets:

```
(gdb) continue
or
(gdb) run
```

You should now see a number of text messages appear, such as:

```
PASS:<GDB Thread test OK>
```

NOTE eCos has no concept of the application exiting. All eCos test cases complete and then run in a continuous tight loop. To return control to GDB you must stop the application.

The usual method of stopping an application is with **Ctrl+C**, but **Ctrl+C** may not work on your platform for the prebuilts. First, make default tests and check that they work the same way as prebuilts, then modify your config to enable GDB stubs (if applicable) and break support, so that a **Ctrl+C** character will interrupt the application.

Another way to stop the application is by means of a breakpoint. Before running the application, breakpoint <code>cyg_test_exit()</code> to stop an eCos test case at its end.

The usual method of stopping an application is with Ctrl+C, but Ctrl+C may not work on your platform for the prebuilts. First, make default tests and check that they work the same way as prebuilts, then modify your config to enable GDB stubs (if applicable) and break support, so that a Ctrl+C character will interrupt the application.

Another way to stop the application is by means of a breakpoint. Before running the application, breakpoint <code>cyg_test_exit()</code> to stop an eCos test case at its end.

The full functionality of GDB is now available to you, including breakpoints and watchpoints. Please consult the GNUPro GDB documentation for further information.

Part III: Programming Tutorial

10

Programming with eCos

The remaining chapters of this manual comprise a simple tutorial for configuring and building eCos, building and running eCos tests, and finally building three stand-alone example programs which use the eCos API to perform some simple tasks.

You will need a properly installed eCos system, with the accompanying versions of the GNUPro tools. On Windows you will be using the bash command line interpreter that comes with Cygwin, with the environment variables set as described in the GNUPro documentation.

The Development Process

Most development projects using eCos would contain some (or most) of the following:

eCos Configuration

eCos is configured to provide the desired API (the inclusion of libc, uitron, and the disabling of certain undesired funtions, etc.), and semantics (selecting scheduler, mutex behavior, etc.). See "Configuring and Building eCos from Source" on page 49.

It would normally make sense to enable eCos assertion checking at this time as well, to catch as many programming errors during the development phase as possible.

Note that it should not be necessary to spend much time on eCos configuration initially. It may be important to perform fine tuning to reduce the memory footprint and to improve performance later when the product reaches a testable state.

Integrity check of the eCos configuration

While Red Hat strive to thoroughly test eCos, the vast number of configuration permutations mean that the particular configuration parameters used for your project may not have been tested. Therefore, we advise running all the eCos tests after the project's eCos configuration has been determined. See "Test Suites" on page 66.

Obviously, this should be repeated if the configuration changes later on in the development process.

Application Development - Target Neutral Part

While your project is probably targeting a specific architecture and platform, possibly custom hardware, part of the application development may be possible to perform using simulated or synthetic targets.

There are two primary reasons for doing this:

- It may be possible by this means to perform application development in parallel with the design/implementation of the target hardware, thus providing more time for developing and testing functionality, and reducing time-to-market.
- The build-run-debug-cycle may be faster when the application does not have to be downloaded to a target via a serial interface. Debugging is also likely to be more responsive when you do not have to to communicate with a stub via serial. It also removes the need for manually or automatically resetting the target hardware.

This is approach is possible because all targets (including simulators and synthetic ones) provide the same basic API: that is, kernel, libc, libm, uitron, infra, and to some extent, HAL and IO.

Synthetic targets are especially suitable as they allow you to jury-rig simulations of elaborate devices by interaction with the host system, where an IO device API can hide the details from the application. When switching to hardware later in the development cycle, the IO driver is properly implemented. While this is possible to do, and has been done, it is not specifically documented or supported by Red Hat. It may become so later.

Therefore, select a simulator or synthetic target and use it for as long as possible doing application development. That is, configure for the selected target, build eCos, build the application and link with eCos, run and debug. Repeat the latter two steps.

Obviously, at some time you will have to switch to the intended target hardware, for example when adding target specific feature support, for memory footprint/performance characterization, and for final tuning of eCos and the application.

Application Development - Target Specific Part

Repeat the build-run-debug-cycle while performing final tuning and debugging of application. Remember to disable eCos assertion checking, as it reduces performance.

It may be useful to switch between this and the previous step repeatedly through the development process; use the simulator/synthetic target for actual development, and use the target hardware to continually check memory footprint and performance. There should be little cost in switching between the two targets when using two separate build trees.

Using the SUBSETROMRAM startup type

NOTE This facility is not available for all boards.

The SUBSETROMRAM startup type combines the ROM and ROMRAM startup types, allowing named functions/rodata to be copied to RAM during system initialization. This provides for a user configurable trade off between RAM footprint and performance.

This startup type works by putting named sections in a special .rom2ram section at link-time. This means that both eCos and application functions/rodata can be relocated to RAM.

Configure the system with CYG_HAL_STARTUP set to SUBSETROMRAM and build eCos as usual. Before it's possible to link eCos to the application (or eCos tests) you have to create the file <code>install/include/subsetromram.spec</code> (and run make again to get the linker script updated).

The file subsetromram.spec is included by the linker script and must contain a single definition of the macro SUBSETROMRAM_SPEC. It is basically a space separated list of section names that should be put in the .rom2ram section (see the ld manual for details about how to write the match expressions):

These section names are the result of the compiler options -ffunction-sections and -fdata-sections, and are constructed by prefixing the function/variable name with '.text.' or '.rodata.'. Note that for C++ member functions, the mangled name is used (use the c++filt tool to demangle the names).

You can find these section names by running objdump with the -h option, listing the hunks (sections) in either the eCos library file (libtarget.a) or the application object(s).

As an example, move some of the functions of the kernel/dhrystone test into RAM:

n Configure eCos, and build the library [CLI tool approach]

```
$ ecosconfig new mpc
```

```
<edit ecos.ecc, changing CYG_HAL_STARTUP to SUBSETROMRAM>
$ ecosconfig tree
$ make -j
```

ⁿ Create the subsetromram.spec file with an empty list:

#define SUBSETROMRAM SPEC

- Build the dhrystone test:
 - \$ make && make -C kernel/current/ tests TESTS=tests/dhrystone
- n Observe that the .rom2ram section is empty:

n Then find the section names of interest:

```
$ sh-elf-objdump -h kernel/current/tests/dhrystone.o|grep text  
0    .text 00000000 00000000 00000000 00000034 2**0  
3    .text.dtime 00000074 00000000 00000000 000000040 2**4  
6    .text.main 000006a0 00000000 00000000 000006f0 2**4  
7    .text.Proc_1 00000000 00000000 00000000 00000660 2**4  
8    .text.Proc_2 00000034 00000000 00000000 00000660 2**4  
9    .text.Proc_3 00000034 00000000 00000000 00000660 2**4  
10    .text.Proc_4 0000002c 00000000 00000000 00000000 2**4  
11    .text.Proc_5 00000020 00000000 00000000 0000060 2**4  
12    .text.Proc_6 00000084 00000000 00000000 00000f10 2**4  
13    .text.Proc_7 00000010 00000000 00000000 00000f20 2**4  
14    .text.Proc_8 00000078 00000000 00000000 00000fd0 2**4  
15    .text.Func_1 00000020 00000000 00000000 00000fd0 2**4  
16    .text.Func_2 00000088 00000000 00000000 00001070 2**4  
17    .text.Func_3 0000016 00000000 00000000 00001100 2**4
```

ⁿ And create the subsetromram.spec file with the list of functions that should be moved to RAM (a random subset of the functions):

- n Rebuild the test.
 - \$ make && make -C kernel/current/ tests TESTS=tests/dhrystone
- n Observe that the .rom2ram section is now non-empty:

n And that the desired functions appear in RAM:

```
$ sh-elf-nm install/tests/kernel/current/tests/dhrystone
[...]
   8c000200 A __rom2ram_start
   8c000230 ? _Proc_4
   8c000230 ? _Proc_5
   8c000250 ? _Proc_6
   8c0002e0 ? _Func_2
   8c000370 ? _Func_3
   8c000390 ? __rom2ram_end
[...]
```

11

Configuring and Building eCos from Source

This chapter documents the configuration of eCos, using the ARM PID board as an example. The process is the same for any of the other supported targets: you may select a hardware target (if you have a board available), any one of the simulators, or a synthetic target (if your host platform has synthetic target support).

At the end of the chapter is a section describing special issues for this architecture which may affect the way you should configure eCos for your target.

eCos Start-up Configurations

There are various ways to download an executable image to a target board, and these involve different ways of preparing the executable image. In the eCos Hardware Abstraction Layer (HAL package) there are configuration options to support the different download methods. The following table summarizes the ways in which an eCos image can be prepared for different types of download.

Table 1: Configuration for various download methods

Download method	HAL configuration
Burn hardware ROM	ROM start-up
Download to ROM emulator	ROM start-up
Download to board with CygMon or GDB stub ROM	RAM start-up
Download to simulator without CygMon or GDB stub ROM	ROM start-up
Download to simulator with CygMon	RAM start-up
Download to simulator ignoring devices	SIM configuration
Run synthetic target	RAM start-up

CAUTION You cannot run an application configured for RAM start-up on the simulator directly: it will fail during start-up. You can only download it to the simulator if:

> you are already running CygMon (or a GDB stub) in the simulator, as described in the GNUPro documentation

NOTE

Configuring eCos' HAL package for simulation should rarely be needed for real development; binaries built with such a kernel will not run on target boards at all. The main use for a "simulation" configuration is if you are trying to work around problems with the device drivers or with the simulator.

If your chosen architecture does not have simulator support, then the combinations above that refer to the simulator do not apply. Similarly, if your chosen platform does not have CygMon or GDB stub ROM support, the combinations listed above that use CygMon or GDB stub ROMs do not apply.

The debugging environment for most developers will be either a hardware board or the simulator, in which case they will be able to select a single HAL configuration.

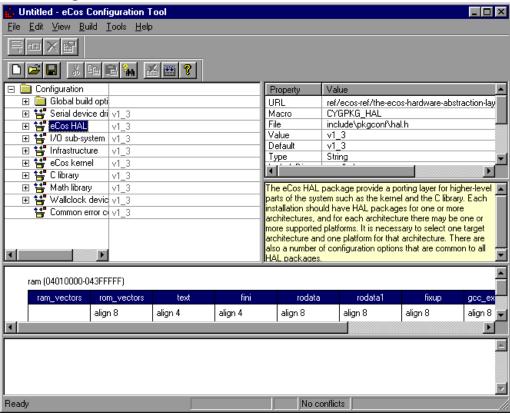
More information on the interactions between CygMon, the simulators, and GDB's thread-aware debugging features is available in the GNUPro Reference Manual for your specific architecture.

Using the Configuration Tool on **Windows**

Note that the use of the **Configuration Tool** is described in detail in the *eCos User's* Guide.

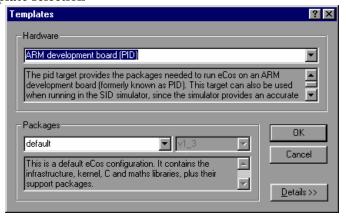
The **Configuration Tool** (see Figure 1) has five main elements: the *configuration window*, the *properties window*, the *short description window*, the *memory layout window*, and the *output window*.

Figure 1: Configuration Tool



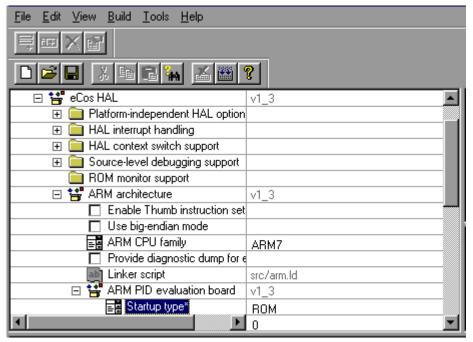
Start by opening the templates window via **Build->Templates**. Select the desired target (see Figure 2).

Figure 2: Template selection



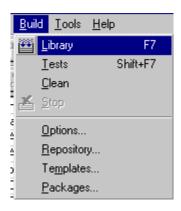
Make sure that the configuration is correct for the target in terms of endianness, CPU model, Startup type, etc. (see Figure 3).

Figure 3: Configuring for the target



Next, select the **Build->Library** menu item to start building eCos (see Figure 4).

Figure 4: Selecting the build Library menu item



The *Save As* dialog box will appear, asking you to specify a directory in which to place your save file. You can use the default, but it is a good idea to make a subdirectory, called *ecos-work* for example.

Figure 5: Build dialog



The first time you build an eCos library for a specific architecture, the **Configuration Tool** may prompt you for the location of the appropriate build tools (including make and gcc) using a **Build Tools** dialog box (as shown in Figure 6, page 54). You can select a location from the drop down list, browse to the directory using the **Browse** button, or type in the location of the build tools manually.

Figure 6: Build tools dialog



The **Configuration Tool** may also prompt you for the location of the user tools (such as cat and ls) using a User Tools dialog box (as shown in Figure 7, page 55). As with the **Build Tools** dialog, you can select a location from the drop down list, browse to the directory using the **Browse** button, or type in the location of the user tools manually.

Figure 7: User tools dialog



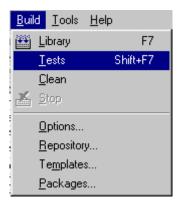
When the tool locations have been entered, the **Configuration Tool** will configure the sources, prepare a build tree, and build the *libtarget.a* library, which contains the eCos kernel and other packages.

The output from the configuration process and the building of *libtarget.a* will be shown in the output window.

Once the build process has finished you will have a kernel with other packages in *libtarget.a*. You should now build the eCos tests for your particular configuration.

You can do this by selecting **Build** -> **Tests**. Notice that you could have selected **Tests** instead of **Library** in the earlier step and it would have built *both* the library and the tests, but this would increase the build time substantially, and if you do not need to build the tests it is unnecessary.

Figure 8: Selecting the build tests menu item



"Test Suites" on page 66 will guide you through running one of the test cases you just built on the selected target, using GDB.

Using ecosconfig on UNIX

On UNIX systems the **Configuration Tool** is not yet available, but it is still possible to configure and build a kernel by editing a configuration file manually and using the *ecosconfig* command.

The following instructions assume that the PATH and ECOS_REPOSITORY environment variables have been setup correctly as described in the section "Software Installation on UNIX" on page 32.

Before invoking *ecosconfig* you need to choose a directory in which to work. For the purposes of this tutorial, the default path will be BASE_DIR/ecos-work. Create this directory and change to it by typing:

```
$ mkdir BASE_DIR/ecos-work
$ cd BASE DIR/ecos-work
```

To see what options can be used with *ecosconfig*, type:

```
$ ecosconfig --help
```

The available packages, targets and templates may be listed as follows:

```
$ ecosconfig list
```

Here is sample output from *ecosconfig* showing the usage message.

Table 2: Getting help from ecosconfig

```
$ ecosconfig --help
Usage: ecosconfig [ qualifier ... ] [ command ]
  commands are:
                                                : list repository contents
   new TARGET [ TEMPLATE [ VERSION ] ]
                                               : create a configuration
    target TARGET
                                               : change the target hardware
                                              : change the template
    template TEMPLATE [ VERSION ]
    add PACKAGE [ PACKAGE ... ]
                                              : add package(s)
   remove PACKAGE [ PACKAGE ... ]
                                              : remove package(s)
   version VERSION PACKAGE [ PACKAGE ... ]
                                              : change version of package(s)
    export FILE
                                               : export minimal config info
    import FILE
                                               : import additional config info
    check
                                               : check the configuration
                                               : resolve conflicts
   resolve
                                               : create a build tree
   tree
  qualifiers are:
                                               : the configuration file
    --config=FILE
    --prefix=DIRECTORY
                                               : the install prefix
    --srcdir=DIRECTORY
                                               : the source repository
                                               : disable conflict resolution
    --no-resolve
    --version
                                               : show version and copyright
```

Table 3: ecosconfig output — list of available packages, targets and templates

```
$ ecosconfig list
```

```
Package CYGPKG CYGMON (CygMon support via eCos):
aliases: cygmon
versions: v1 3 x
Package CYGPKG DEVICES WALLCLOCK DALLAS DS1742 (Wallclock driver
for Dallas 1742):
aliases: devices wallclock ds1742 device wallclock ds1742
versions: v1 3 x
Package CYGPKG DEVICES WALLCLOCK SH3 (Wallclock driver for SH3 RTC
module):
aliases: devices wallclock sh3 device wallclock sh3
versions: v1 3 x
Package CYGPKG DEVICES WATCHDOG ARM AEB (Watchdog driver for ARM/
AEB board):
aliases: devices watchdog aeb device watchdog aeb
versions: v1 3 x
Package CYGPKG DEVICES WATCHDOG ARM EBSA285 (Watchdog driver for
ARM/EBSA285 board):
aliases: devices watchdog ebsa285 device watchdog ebsa285
versions: v1 3 x
Package CYGPKG DEVICES WATCHDOG MN10300 MN10300 (Watchdog driver
for MN10300 chip):
aliases: devices watchdog mn10300 device watchdog mn10300
versions: v1_3_x
Package CYGPKG DEVICES WATCHDOG SH SH3 (Watchdog driver for the
Hitachi SH3 chip):
aliases: devices watchdog sh3 device watchdog sh3
versions: v1 3 x
Package CYGPKG DEVS ETH ARM EBSA285 (Intel EBSA285 with PRO/100+
ethernet driver):
aliases: devs eth arm ebsa285 ebsa285 eth driver
versions: v1 \overline{3} x
Package CYGPKG DEVS ETH ARM EDB7XXX (Cirrus Logic ethernet driver):
aliases: edb7xxx eth driver
versions: v1 3 x
Package CYGPKG DEVS ETH CF (PCMCIA (Compact Flash) ethernet
drivers):
aliases: cf eth drivers
versions: v1 3 x
Package CYGPKG DEVS ETH POWERPC QUICC (QUICC ethernet driver):
aliases: quicc eth driver
versions: v1 3 x
Package CYGPKG DEVS FLASH ASSABET (FLASH memory support for Intel
SA1110 (Assabe\overline{t}):
aliases: flash assabet
versions: v1 3 x
Package CYGPKG DEVS FLASH EBSA285 (FLASH memory support for
StrongARM EBSA-285):
aliases: flash ebsa285
versions: v1 3 x
Package CYGPKG DEVS FLASH EDB7XXX (FLASH memory support for Cirrus
Logic EDB7xxx):
aliases: flash edb7xxx
versions: v1 3 x
Package CYGPKG DEVS PCMCIA ASSABET (SA11x0/Assabet PCMCIA & Compact
Flash support):
aliases: pcmcia assabet assabet pcmcia assabet cf io assabet pcmcia
versions: v1 3 x
Package CYGPKG ERROR (Common error code support):
aliases: error errors
```

```
versions: v1 3 x
Package CYGPKG FS RAM (RAM Filesystem):
aliases: ramfs ram fs fs ram
versions: v1 3 x
Package CYGPKG HAL (eCos common HAL):
aliases: hal hal common
versions: v1 3 x
Package CYGPKG HAL ARM (ARM common HAL):
aliases: hal arm arm hal arm arch hal
versions: v1 3 x
Package CYGPKG HAL ARM AEB (ARM evaluation board (AEB-1)):
aliases: hal arm aeb arm aeb hal
versions: v1 3 x
Package CYGPKG HAL ARM CMA230 (Cogent CMA230/222 board):
aliases: hal arm cma230 arm cma230 hal
versions: v1 3 x
Package CYGPKG HAL ARM EBSA285 (Intel EBSA285 StrongARM board):
aliases: hal arm ebsa285 arm ebsa285 hal
versions: v1 3 x
Package CYGPKG HAL ARM EDB7XXX (Cirrus Logic development board):
aliases: hal arm edb7xxx arm edb7xxx hal
versions: v1 3 x
Package CYGPKG HAL ARM PID (ARM development board (PID)):
aliases: hal arm pid arm pid hal
versions: v1 3 x
Package CYGPKG HAL ARM SA11X0 (Intel SA11X0 Chipset):
aliases: hal arm sallx0
versions: v1 3 x
Package CYGPKG HAL ARM SA11X0 ASSABET (Intel SA1110 Assabet eval
board):
aliases: hal arm sal1x0 assabet
versions: v1 3 x
Package CYGPKG HAL ARM SA11X0 BRUTUS (Intel SA1100 Brutus eval
board):
aliases: hal arm sallx0 brutus
versions: v1 3 x
Package CYGPKG HAL I386 (i386 common HAL):
aliases: hal i386 i386 hal i386 arch hal
versions: v1 3 x
Package CYGPKG HAL I386 LINUX (Linux synthetic target):
aliases: hal i386 linux
versions: v1 3 x
Package CYGPKG HAL I386 PC (i386 PC target):
aliases: hal \overline{1386} pc
versions: v1 3 x
Package CYGPKG HAL MIPS (MIPS common HAL):
aliases: hal mips mips hal mips arch hal
versions: v1 3 x
Package CYGPKG HAL MIPS SIM (MIPS simulator):
aliases: hal mips sim mips sim hal
versions: v1 3 x
Package CYGPKG HAL MIPS TX39 (TX39 chip HAL):
aliases: hal tx39 tx39 hal tx39 arch hal
versions: v1 3 x
Package CYGPKG HAL MIPS TX39 JMR3904 (Toshiba JMR-TX3904 board):
aliases: hal t\bar{x}39 \ \bar{j}mr3904 \ tx\bar{3}9 \ jmr3904 \ hal
versions: v1 3 x
Package CYGPKG HAL MIPS VR4300 (VR4300 chip HAL):
aliases: hal vr4300 vr4300 hal vr4300 arch hal
```

```
versions: v1 3 x
Package CYGPKG HAL MIPS VR4300 VRC4373 (NEC VRC4373 board):
aliases: hal vrc4373 vrc4373 hal
versions: v1 3 x
Package CYGPKG HAL MN10300 (MN10300 common HAL):
aliases: hal mn10300 mn10300 hal mn10300 arch hal
versions: v1 3 x
Package CYGPKG HAL MN10300 AM31 (MN10300 AM31 variant HAL):
aliases: hal mn10300 am31 mn10300 am31 hal
versions: v1 3 x
Package CYGPKG HAL MN10300 AM31 SIM (MN10300 simulator):
aliases: hal mn10300 sim mn10300 sim hal
versions: v1 3 x
Package CYGPKG HAL MN10300 AM31 STDEVAL1 (Matsushita stdeval1
board):
aliases: hal mn10300 stdeval1 mn10300 stdeval1 hal
versions: v1 3 x
Package CYGPKG HAL MN10300 AM33 (MN10300 AM33 variant HAL):
aliases: hal mn10300 am33 mn10300 am33 hal
versions: v1 3 x
Package CYGPKG HAL MN10300 AM33 STB (Matsushita STB board):
aliases: hal mn10300 am33 stb mn10300 am33 stb hal
versions: v1_3_x
Package CYGPKG HAL POWERPC (PowerPC common HAL):
aliases: hal powerpc powerpc hal powerpc arch hal
versions: v1 3 x
Package CYGPKG HAL POWERPC COGENT (Cogent CMA286/287 board):
aliases: hal powerpc cogent powerpc cogent hal
versions: v1 3 x
Package CYGPKG HAL POWERPC FADS (Motorola MPC8xxFADS board):
aliases: hal powerpc fads powerpc fads hal
versions: v1 3 x
Package CYGPKG HAL POWERPC MBX (Motorola MBX860/821 board):
aliases: hal powerpc mbx powerpc mbx hal
versions: v1_3_x
Package CYGPKG HAL POWERPC MPC8xx (PowerPC 8xx variant HAL):
aliases: hal mpc8xx mpc8xx hal mpc8xx arch hal
versions: v1 3 x
Package CYGPKG HAL POWERPC PPC60x (PowerPC 60x variant HAL):
aliases: hal ppc60x ppc60x hal ppc60x arch hal
versions: v1 3 x
Package CYGPKG HAL POWERPC SIM (PowerPC simulator):
aliases: hal powerpc sim powerpc sim hal
versions: v1 3 x
Package CYGPKG HAL QUICC (Motorola MBX860/821 QUICC support):
aliases: hal_quicc quicc_hal quicc
versions: v1_3_x
Package CYGPKG HAL SH (SH common HAL):
aliases: hal sh sh hal sh arch hal
versions: v1 3 x
Package CYGPKG HAL SH EDK7708 (Hitachi SH7708 board):
aliases: hal sh edk sh edk hal
versions: v1 3 x
Package CYGPKG HAL SH SH7708 CQ7708 (CqREEK SH7708 board):
aliases: hal sh cq sh cq hal
versions: v1 3 x
Package CYGPKG HAL SPARCLITE (SPARClite common HAL):
aliases: hal sparclite sparclite hal sparclite arch hal
versions: v1 3 x
```

```
Package CYGPKG HAL SPARCLITE SIM (SPARClite simulator):
aliases: hal sparclite sim sparclite sim hal
versions: v1 3 x
Package CYGPKG HAL SPARCLITE SLEB (Fujitsu MB86800-MA01 board):
aliases: hal sparclite sleb sparclite sleb hal
versions: v1 3 x
Package CYGPKG HAL V85X (NEC V85x common HAL):
aliases: hal v85x nec arch hal
versions: v1 3 x
Package CYGPKG HAL V85X V850 (NEC V850 variant HAL):
aliases: hal_v85x_v850 nec_v850_hal
versions: v1 3 x
Package CYGPKG HAL V85X V850 CEB (Cosmo CEB-V850/SA1 board):
aliases: hal v85x v850 ceb nec v850 ceb hal
versions: v1 3 x
Package CYGPKG INFRA (Infrastructure):
aliases: infra
versions: v1 3 x
Package CYGPKG IO (I/O sub-system):
aliases: io
versions: v1 3 x
Package CYGPKG IO DSP BLOCK (DSP (block) device drivers):
aliases: dsp_block io_dsp_block
versions: v1 3 x
Package CYGPKG IO ETH DRIVERS (Common ethernet support):
aliases: net drivers eth drivers
versions: v1 3 x
Package CYGPKG IO FILEIO (File IO):
aliases: fileio io file
versions: v1 3 x
Package CYGPKG IO FLASH (Generic FLASH memory support):
aliases: flash
versions: v1 3 x
Package CYGPKG IO PCI (PCI configuration library):
aliases: io pci
versions: v\overline{1} 3 x
Package CYGPKG IO PCMCIA (PCMCIA & Compact Flash support):
aliases: pcmcia io pcmcia cf
versions: v1 3 x
Package CYGPKG IO SERIAL (Serial device drivers):
aliases: serial io serial
versions: v1 3 x
Package CYGPKG IO SERIAL ARM AEB (ARM AEB-1 serial device drivers):
aliases: devs serial arm aeb aeb serial driver
versions: v1 \overline{3} x
Package CYGPKG IO SERIAL ARM CMA230 (Cogent ARM/CMA230 serial
device drivers):
aliases: devs serial arm cma230 cma230 serial driver
versions: v1 \overline{3} x
Package CYGPKG IO SERIAL ARM EBSA285 (Intel EBSA285 serial driver):
aliases: devs serial arm ebsa285 ebsa285 serial driver
versions: v1 3 x
Package CYGPKG IO SERIAL ARM EDB7XXX (ARM EDB7XXX serial device
drivers):
aliases: devs serial arm edb7xxx edb7xxx serial driver
versions: v1 3 x
Package CYGPKG IO SERIAL ARM PID (ARM PID serial device drivers):
aliases: devs serial arm pid pid serial driver
versions: v1 3 x
```

```
Package CYGPKG IO SERIAL ARM SA11X0 (Intel StrongARM SA11x0 serial
driver):
aliases: devs serial arm sallx0 sallx0 serial driver
versions: v1 3 x
Package CYGPKG IO SERIAL GENERIC 16X5X (16x5x compatible serial
device drivers):
aliases: devs serial generic 16x5x 16x5x serial driver
versions: v1 \overline{3} x
Package CYGPKG IO SERIAL I386 PC (PC serial device drivers):
aliases: devs serial i386 pc pc serial driver
versions: v1_{\overline{3}}x
Package CYGPKG IO SERIAL LOOP (Loop serial device drivers):
aliases: devs serial loop loop serial driver
versions: v1 3 x
Package CYGPKG IO SERIAL MIPS VRC4373 (VRC4373 serial device
drivers):
aliases: devs serial mips vrc4373 vrc4373 serial driver
versions: v1 3 x
Package CYGPKG IO SERIAL MN10300 (MN10300 serial device drivers):
aliases: devs serial mn10300 mn10300 devs serial mn10300
mn10300 serial driver
versions: v1_3_x
Package CYGPKG IO SERIAL POWERPC COGENT (Cogent PowerPC serial
device drivers):
aliases: devs serial powerpc cogent cogent serial driver
versions: v1 3 x
Package CYGPKG IO SERIAL POWERPC QUICC SMC (PowerPC QUICC/SMC
serial device drivers):
aliases: devs serial quicc smc quicc smc serial driver
devs serial powerpc quicc smc devs serial powerpc quicc
quicc serial driver
versions: v1 3 x
Package CYGPKG IO SERIAL SH CQ7708 (SH3 cg7708 serial device
drivers):
aliases: devs_serial_sh3_cq7708 devs serial sh cq7708
cq7708 serial driver
versions: v1 3 x
Package CYGPKG IO SERIAL SH EDK7708 (SH3 EDK7708 serial device
drivers):
aliases: devs serial sh3 edk7708 devs serial sh edk7708
edk7708 serial driver
versions: v1 3 x
Package CYGPKG IO SERIAL SH SCI (SH SCI serial device drivers):
aliases: devs serial sh sci sci serial driver
versions: v1 \overline{3} x
Package CYGPKG IO SERIAL SH SCIF (SH SCIF serial device drivers):
aliases: devs serial sh scif scif serial driver
versions: v1 \overline{3} x
Package CYGPKG IO SERIAL SPARCLITE SLEB (SPARClite SLEB serial
device drivers):
aliases: devs serial sparclite sleb sleb serial driver
versions: v1 3 x
Package CYGPKG IO SERIAL TX39 JMR3904 (TX39 JMR3904 serial device
drivers):
aliases: devs serial tx39 jmr3904 jmr3904 serial driver
devs serial mips jmr3904 tx3904 serial driver
versions: v1 3 x
Package CYGPKG IO SERIAL V85X V850 (NEC V850 serial device
drivers):
```

```
aliases: devs serial v85x v850 v850 serial driver
versions: v1 \overline{3} x
Package CYGPKG IO WALLCLOCK (Wallclock device framework):
aliases: wallclock io wallclock devices wallclock device wallclock
versions: v1 3 x
Package CYGPKG IO WATCHDOG (Watchdog IO device):
aliases: watchdog io watchdog
versions: v1 3 x
Package CYGPKG ISOINFRA (ISO C and POSIX infrastructure):
aliases: isoinfra
versions: v1_3_x
Package CYGPKG KERNEL (eCos kernel):
aliases: kernel
versions: v1 3 x
Package CYGPKG LIBC (C library):
aliases: libc clib clibrary
versions: v1 3 x
Package CYGPKG LIBC I18N (ISO C library internationalization):
aliases: libc 118n
versions: v1 3 x
Package CYGPKG LIBC SETJMP (ISO C library non-local jumps):
aliases: libc setjmp
versions: v1 3 x
Package CYGPKG LIBC SIGNALS (ISO C library signals):
aliases: libc signals
versions: v1 3 x
Package CYGPKG LIBC STARTUP (ISO environment startup/termination):
aliases: libc startup
versions: v1 3 x
Package CYGPKG LIBC STDIO (ISO C library standard input/output
functions):
aliases: libc stdio
versions: v1 \overline{3} x
Package CYGPKG LIBC STDLIB (ISO C library general utility
functions):
aliases: libc stdlib
versions: v1 \overline{3} x
Package CYGPKG LIBC STRING (ISO C library string functions):
aliases: libc string
versions: v1 3 x
Package CYGPKG LIBC TIME (ISO C library date/time functions):
aliases: libc \overline{t}ime
versions: v1 \overline{3} x
Package CYGPKG LIBM (Math library):
aliases: libm mathlib mathlibrary
versions: v1 3 x
Package CYGPKG MEMALLOC (Dynamic memory allocation):
aliases: memalloc malloc
versions: v1 3 x
Package CYGPKG POSIX (POSIX compatibility):
aliases: posix
versions: v1 3 x
Package CYGPKG REDBOOT (RedBoot, the Red Hat bootstrap):
aliases: cygmon
versions: v1 3 x
Package CYGPKG UITRON (uITRON compatibility):
aliases: uitron
versions: v1 3 x
Target aeb (ARM evaluation board (AEB-1)):
```

```
aliases: aeb1
Target am31 sim (MN10300 AM31 minimal simulator):
aliases:
Target assabet (Intel StrongARM SA1110 board):
aliases: assabet
Target brutus (Intel StrongARM SA1100 board):
aliases: brutus
Target ceb v850 (Cosmo CEB-V850/SA1 board):
aliases: CEB
Target cma230 (Cogent CMA230/222 board):
aliases: cma222
Target cma28x (Cogent CMA286/287 board):
aliases: cma286 cma287
Target cq7708 (CqREEK SH7708 board):
aliases: cq7708
Target ebsa285 (Intel EBSA285 StrongARM board):
aliases: ebsa
Target edb7xxx (Cirrus Logic development board):
aliases: edb7211 eb7xxx eb7211
Target fads (Motorola MPC8xxFADS board):
aliases:
Target jmr3904 (Toshiba JMR-TX3904 board):
aliases: jmr tx39
Target linux (Linux synthetic target):
aliases:
Target mbx (Motorola MBX860/821 board):
aliases: mbx860 mbx821
Target pc (i386 PC target):
aliases:
Target pid (ARM development board (PID)):
aliases: PID
Target psim (PowerPC simulator):
aliases: ppc sim powerpc sim
Target sh7708 (Hitachi EDK/SH7708 board):
aliases: edk7708
Target sleb (Fujitsu MB86800-MA01 board):
aliases:
Target sparclite sim (SPARClite simulator):
aliases: sl sim sparcl sim
Target stb (Matsushita STB board):
aliases:
Target stdeval1 (Matsushita stdeval1 board):
aliases:
Target tx39 sim (TX39 minimal simulator):
aliases:
Target vrc4373 (NEC VRC4373 board):
aliases:
Template all:
versions: v1 3 x
Template cygmon:
versions: v1 3 x
Template cygmon no kernel:
versions: v1 3 x
Template default:
versions: v1 3 x
Template elix:
versions: v1 3 x
Template kernel:
versions: v1 3 x
```

```
Template minimal:
versions: v1_3_x
Template posix:
versions: v1_3_x
Template redboot:
versions: v1_3_x
Template stubs:
versions: v1_3_x
Template uitron:
versions: v1_3_x
```

For detailed information about how to edit the ecos.ecc file, see the *CDL Writer's Guide* and *Editing an eCos Savefile* in the *eCos User's Guide*.

Selecting a Target

To configure for a listed target, type:

```
$ ecosconfig new <target>
```

For example, to configure for the ARM PID development board, type:

```
$ ecosconfig new pid
```

Then edit the generated file, ecos.ecc, setting the options as required for the target (endianess, CPU model, Startup type, etc.)

Create a build tree for the configured target by typing:

```
$ ecosconfig tree
```

You can now run the command *make* or *make tests*, after which you will be at the same point you would be after running the **Configuration Tool** on Windows— you can start developing your own applications, following the steps in "Building and Running Sample Applications" on page 69.

The procedure shown above allows you to do very coarse-grained configuration of the eCos kernel: you can select which packages to include in your kernel, and give target and start-up options. But you cannot select components within a package, or set the very fine-grained options.

To select fine-grained configuration options you will need to edit the configuration file ecos.ecc in the current directory and regenerate the build tree.

CAUTION

You should follow the manual configuration process described above very carefully, and you should read the comments in each file to see when one option depends on other options or packages being enabled or disabled. If you do not, you might end up with an inconsistently configured kernel which could fail to build or might execute incorrectly.

Architectural Notes

There are no notes for this architecture.

12

Test Suites

The eCos kernel and other packages have test suites that rigorously exercise the available features and confirm correct execution. The tests are run on many different possible configurations, but the high number of configuration permutations makes it impossible to test them all. The use of test suites is particularly important for embedded systems, where software robustness is a priority. All eCos software is tested prior to shipping, but if you define your own configuration, you will probably want to verify that the test cases work for it.

This release includes test suites for the eCos kernel, kernel C API, C library, µITRON compatibility, and device driver packages. The use of the test suites is similar for all packages. The tests are supplied as source code for building with your specific eCos configurations. The test case source code is located under the base source directory BASE_DIR/packages/:

- n compat/uitron/v1_4_x/tests
- $_{n}$ hal/common/v1_4_x/tests
- n io/serial/v1_4_x/tests
- n io/wallclock/v1_4_x/tests
- n devs/watchdog/v1_4_x/tests
- n kernel/v1_4_x/tests
- n language/c/libc/v1_4_x/tests
- $_{n} \quad language/c/libm/v1_4_x/tests$

There may be additional tests found in other packages.

Each test suite consists of a number of test cases which can be executed individually. A test case may involve one or more individual tests of the package's features. Successful completion of each test within the test case is reported as a line of text that is sent to the diagnostic channel (usually the serial port) for display on a terminal or terminal emulator.

Each test case runs only once and usually requires target hardware to be reset on completion. Note that certain test cases may not terminate immediately, especially if they involve delays and run on a target simulator.

Using the Configuration Tool

Using the eCos Configuration Tool it is possible to automate the downloading and execution of tests with the appropriately configured eCos packages. To do so, compile and link the test cases by using the **Build->Tests** menu item, after which the tests can be downloaded and executed by selecting **Tools->Run Tests**.

When a test run is invoked, a resizable property sheet is displayed, comprising three tabs: **Executables**, **Output** and **Summary**.

Three buttons appear on the property sheet itself: **Run/Stop**, **Close** and **Properties**.

The **Run** button is used to initiate a test run. Those tests selected on the Executables tab are run, and the output recorded on the **Output** and **Summary** tabs. During the course of a run, the **Run** button changes to **Stop**. This button may be used to interrupt a test run at any point.

See the *eCos User's Guide* for further details.

Using the command line

At the moment, there is no tool for automating testing on Linux, so you will have to run the tests manually.

It may also be necessary to run tests by hand if the automated tool finds any failing tests: it may be necessary to diagnose the problem by debugging the test.

Build the tests by typing 'make tests' in the root of the build directory. This will cause the tests to be built and installed under <install-path>/tests/.

Running the test manually is done simply by invoking GDB, connecting to the target, downloading the test, optionally setting some breakpoints, and then running the test. All this was covered in "Target Setup" on page 36.

Testing Filters

While most test cases today run solely in the target environment, some packages may require external testing infrastructure and/or feedback from the external environment to do complete testing.

The serial package is an example of this. It is the first package to require external testing infrastructure, but it will certainly not be the last.

Since the serial line is also used for communication with GDB, a filter is inserted in the communication pathway between GDB and the serial device which is connected to the hardware target. The filter forwards all communication between the two, but also listens for special commands embedded in the data stream from the target.

When such a command is seen, the filter stops forwarding data to GDB from the target and enters a special mode. In this mode the test case running on the target is able to control the filter, commanding it to run various tests. While these tests run, GDB is isolated from the target.

As the test completes (or if the filter detects a target crash) the communication path between GDB and the hardware target is re-established, allowing GDB to resume control.

In theory, it is possible to extend the filter to provide a generic framework for other target-external testing components, thus decoupling the testing infrastructure from the (possibly limited) communication means provided by the target (serial, JTAG, Ethernet, etc).

Another advantage is that the host tools will not need to know about the various testing environments required by the eCos packages, since all contact with the target will continue to happen via GDB.

It remains to be seen if it will be possible, or sensible, to implement all target-external testing infrastructure via filters.

13

Building and Running Sample Applications

The example programs in this tutorial are included, along with a *Makefile*, in the *examples* directory of the eCos distribution. The first program you will run is a *hello world*-style application, then you will run a more complex application that demonstrates the creation of threads and the use of <code>cyg_thread_delay()</code>, and finally you will run one that uses clocks and alarm handlers.

The *Makefile* has two variables you will need to adjust: *PKG_INSTALL_DIR* and *XCC*.

Edit the Makefile, setting *PKG_INSTALL_DIR* to the install tree previously created by ecosconfig and uncommenting the relevant *XCC* line for your architecture.

eCos Hello World

The following code is found in the file hello.c in the *examples* directory:

eCos hello world program listing

```
/* this is a simple hello world program */
#include <stdio.h>
int main(void)
{
  printf("Hello, eCos world!\n");
  return 0;
}
```

To compile this or any other program that is not part of the eCos distribution, you can follow the procedures described below. Type this explicit compilation instruction (assuming your current working directory is also where you built the eCos kernel):

```
$ gcc -g -IBASE_DIR/ecos-work/install/include hello.c -LBASE_DIR/
ecos-work/install/lib -Ttarget.ld -nostdlib
```

The compilation instruction above contains some standard GCC options (for example, -g enables debugging), as well as some mention of paths (-IBASE_DIR/ecos-work/install/include allows files like cyg/kernel/kapi.h to be found, and - LBASE_DIR/ecos-work/install/lib allows the linker to find -Ttarget.ld).

The executable program will be called a .out.

NOTE Some target systems require special options to be passed to gcc to compile correctly for that system. Please examine the Makefile in the examples directory to see if this applies to your target.

You can now run the resulting program in the simulator using GDB the way you ran the test case. The procedure will be the same, but this time run "gdb" specifying "-nw a.out" on the command line:

```
$ gdb -nw a.out
```

For targets other than the synthetic linux target, you should now run the usual GDB commands described earlier. Once this is done, typing the command "run" at the (gdb) prompt ("continue" for real hardware) will allow the program to execute and print the string "Hello, eCos world!" on your screen.

On the synthetic linux target, you may use the "run" command immediately - you do not need to invoke simulator macros, nor the "load" command.

A Sample Program with Two Threads

Below is a program that uses some of eCos' system calls. It creates two threads, each of which goes into an infinite loop in which it sleeps for a while (using cyg_thread_delay()). This code is found in the file twothreads.c in the examples directory.

eCos two-threaded program listing

```
#include <cyg/kernel/kapi.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

/* now declare (and allocate space for) some kernel objects,
like the two threads we will use */
cyg_thread thread_s[2];/* space for two thread objects */
```

```
char stack[2][4096];/* space for two 4K stacks */
/* now the handles for the threads */
cyg handle t simple threadA, simple threadB;
/* and now variables for the procedure which is the thread */
cyg thread entry t simple program;
/* and now a mutex to protect calls to the C library */
cyg mutex t cliblock;
/* we install our own startup routine which sets up threads */
void cyg user start(void)
printf("Entering twothreads' cyg user start() function\n");
cyq mutex init(&cliblock);
cyg thread create(4, simple program, (cyg addrword t) 0,
 "Thread A", (void *) stack[0], 4096,
 &simple threadA, &thread s[0]);
cyg_thread_create(4, simple_program, (cyg_addrword_t) 1,
 "Thread B", (void *) stack[1], 4096,
 &simple threadB, &thread s[1]);
cyg thread resume (simple threadA);
cyg thread resume (simple threadB);
/* this is a simple program which runs in a thread */
void simple program(cyg addrword t data)
int message = (int) data;
int delay;
printf("Beginning execution; thread data is %d\n", message);
cyg thread delay(200);
for (;;) {
delay = 200 + (rand() % 50);
/* note: printf() must be protected by a
call to cyg mutex lock() */
cyg mutex lock(&cliblock); {
printf("Thread %d: and now a delay of %d clock ticks\n",
 message, delay);
cyg mutex unlock(&cliblock);
cyg thread delay(delay);
```

When you run the program (by typing run at the (**gdb**) prompt) the output should look like this:

```
Starting program: BASE_DIR/examples/twothreads.exe Entering twothreads' cyg user start() function
```

```
Beginning execution; thread data is 0
Beginning execution; thread data is 1
Thread 0: and now a delay of 240 clock ticks
Thread 1: and now a delay of 225 clock ticks
Thread 0: and now a delay of 234 clock ticks
Thread 0: and now a delay of 231 clock ticks
Thread 1: and now a delay of 24 clock ticks
Thread 0: and now a delay of 249 clock ticks
Thread 0: and now a delay of 202 clock ticks
Thread 0: and now a delay of 202 clock ticks
Thread 0: and now a delay of 235 clock ticks
```

NOTE When running in a simulator the delays might be quite long. On a hardware board (where the clock speed is 100 ticks/second) the delays should average to about 2.25 seconds. In simulation, the delay will depend on the speed of the processor and will almost always be much slower than the actual board. You might want to reduce the delay parameter when running in simulation.

Figure 9, page 73 shows how this multitasking program executes. Note that apart from the thread creation system calls, this program also creates and uses a *mutex* for synchronization between the <code>printf()</code> calls in the two threads. This is because the C library standard I/O (by default) is configured not to be thread-safe, which means that if more than one thread is using standard I/O they might corrupt each other. This is fixed by a mutual exclusion (or *mutex*) lockout mechanism: the threads do not call <code>printf()</code> until <code>cyg_mutex_lock()</code> has returned, which only happens when the other thread calls <code>cyg_mutex_unlock()</code>.

You could avoid using the mutex by configuring the C library to be thread-safe (by selecting the component CYGSEM_LIBC_STDIO_THREAD_SAFE_STREAMS). Keep in mind that if the C library is thread-safe, you can no longer use printf() in cyg user start().

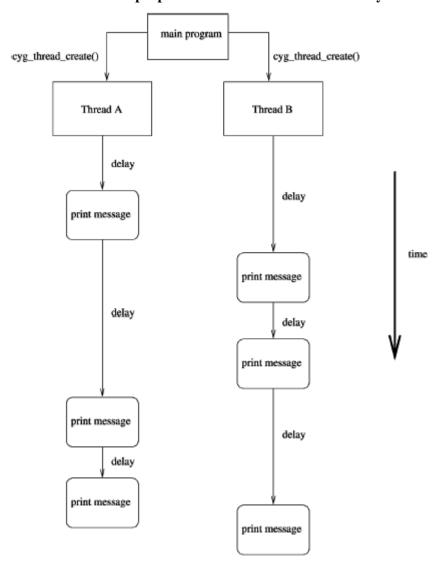


Figure 9: Two threads with simple print statements after random delays

14

More Features — Clocks and Alarm Handlers

If a program wanted to execute a task at a given time, or periodically, it could do it in an inefficient way by sitting in an infinite loop and checking the real-time clock to see if the proper amount of time has elapsed. But operating systems usually provide system calls which allow the program to be interrupted at the desired time.

eCos provides a rich timekeeping formalism, involving *counters*, *clocks*, *alarms*, and *timers*. The precise definition, relationship, and motivation of these features is beyond the scope of this tutorial, but these examples illustrate how to set up basic periodic tasks.

Alarms are events that happen at a given time, either once or periodically. A thread associates an alarm handling function with the alarm, so that the function will be invoked every time the alarm "goes off".

A Sample Program with Alarms

simple-alarm.c (in the examples directory) is a short program that creates a thread that creates an alarm. The alarm is handled by the function test_alarm_func(), which sets a global variable. When the main thread of execution sees that the variable has changed, it prints a message.

Table 4: A sample program that creates an alarm

```
/* this is a very simple program meant to demonstrate
a basic use of time, alarms and alarm-handling functions
in eCos */
#include <cyg/kernel/kapi.h>
#include <stdio.h>
#define NTHREADS 1
#define STACKSIZE 4096
static cyg handle t thread[NTHREADS];
static cyg thread thread obj [NTHREADS];
static char stack[NTHREADS][STACKSIZE];
static void alarm prog( cyg addrword t data );
/* we install our own startup routine which sets up
threads and starts the scheduler */
void cyg_user_start(void)
cyg thread create (4, alarm prog, (cyg addrword t) 0,
 "alarm thread", (void *) stack[0],
 STACKSIZE, &thread[0], &thread obj[0]);
cyg_thread_resume(thread[0]);
/* we need to declare the alarm handling function (which is
defined below), so that we can pass it to
cyg_alarm_initialize() */
cyg alarm t test alarm func;
/* alarm prog() is a thread which sets up an alarm which is then
handled by test alarm func() */
static void alarm_prog(cyg_addrword_t data)
cyg handle t test counterH, system clockH, test alarmH;
cyg tick count t ticks;
cyg alarm test alarm;
unsigned how many alarms = 0, prev alarms = 0, tmp how many;
system_clockH = cyg_real_time_clock();
cyg_clock_to_counter(system_clockH, &test_counterH);
cyg_alarm_create(test_counterH, test_alarm_func,
 (cyg_addrword_t) &how_many_alarms,
 &test alarmH, &test alarm);
cyg_alarm_initialize(test_alarmH, cyg_current_time()+200, 200);
/* get in a loop in which we read the current time and
print it out, just to have something scrolling by */
for (;;) {
ticks = cyg current time();
printf("Time is %llu\n", ticks);
/* note that we must lock access to how_many_alarms, since the
alarm handler might change it. this involves using the
annoying temporary variable tmp how many so that I can keep the
```

```
critical region short */
cyg_scheduler_lock();
tmp_how_many = how_many_alarms;
cyg_scheduler_unlock();
if (prev_alarms != tmp_how_many) {
  printf(" --- alarm calls so far: %u\n", tmp_how_many);
  prev_alarms = tmp_how_many;
}
  cyg_thread_delay(30);
}

/* test_alarm_func() is invoked as an alarm handler, so
  it should be quick and simple. in this case it increments
  the data that is passed to it. */
void test_alarm_func(cyg_handle_t alarmH, cyg_addrword_t data)
{
  ++*((unsigned *) data);
}
```

When you run this program (by typing run at the (**gdb**) prompt) the output should look like this:

```
Starting program: BASE DIR/examples/simple-alarm.exe
Time is 0
Time is 30
Time is 60
Time is 90
Time is 120
Time is 150
Time is 180
Time is 210
 --- alarm calls so far: 1
Time is 240
Time is 270
Time is 300
Time is 330
Time is 360
Time is 390
Time is 420
  --- alarm calls so far: 2
Time is 450
Time is 480
```

When running in a simulator the delays might be quite long. On a hardware board (where the clock speed is 100 ticks/second) the delays should average to about 0.3 seconds (and 2 seconds between alarms). In simulation, the delay will depend on the speed of the processor and will almost always be much slower than the actual board. You might want to reduce the delay parameter when running in simulation.

Here are a few things you might notice about this program:

It used the cyg_real_time_clock(); this always returns a handle to the default system real-time clock.

- Alarms are based on counters, so the function cyg_alarm_create() uses a counter handle. The program used the function cyg_clock_to_counter() to strip the clock handle to the underlying counter handle.
- Once the alarm is created it is initialized with cyg_alarm_initialize(), which sets the time at which the alarm should go off, as well as the period for repeating alarms. It is set to go off at the current time and then to repeat every 200 ticks.
- The alarm handler function test_alarm_func() conforms to the guidelines for writing alarm handlers and other *delayed service routines*: it does not invoke any functions which might lock the scheduler. This is discussed in detail in the *eCos Reference Manual*, in the chapter *Requirements for programs*.
- There is a *critical region* in this program: the variable *how_many_alarms* is accessed in the main thread of control and is also modified in the alarm handler. To prevent a possible (though unlikely) race condition on this variable, access to *how_many_alarms* in the principal thread is protected by calls to cyg_scheduler_lock() and cyg_scheduler_unlock(). When the scheduler is locked, the alarm handler will not be invoked, so the problem is averted.

Appendixes

Appendix 1: Real-time characterization

For a discussion of real-time performance measurement for eCos, see the *eCos Users' Guide*.

Sample numbers:

Board: PC

CPU: 433MHz Celeron

```
Startup, main stack
                                        124 size 2912
                            : stack used
Startup
                   : Interrupt stack used 280 size 4108
Startup
                   : Idlethread stack used
                                        62 size 2048
eCos Kernel Timings
Notes: all times are in microseconds (.000001) unless otherwise stated
Reading the hardware clock takes 8 'ticks' overhead
... this value will be factored out of all other measurements
Clock interrupt took 6.75 microseconds (8 raw clock ticks)
Testing parameters:
  Clock samples:
                          32
  Threads:
                          64
  Thread switches:
                         128
  Mutexes:
  Mailboxes:
  Semaphores:
                         32
  Scheduler operations:
  Counters:
  Alarms:
                             Confidence
          Min
                 Max
                        Var Ave Min Function
    Ave
 -----
        _____
   3.93
         1.68
                8.38
                       0.93 68% 3% Create thread
          0.00
                 3.35 0.84 59% 59% Yield thread [all suspended]
   0.71
        0.00
                  5.03 0.84 64% Suspend [suspended] thread
   0.65
```

0.63 0.00 1.68 0.79 62% 62% Resume thread

0.76 0.39 1.34 0.68 0.92 0.63 0.84 0.73 0.58 1.26 0.86 1.44 4.45 1.62	0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.0	1.68 1.68 6.70 1.68 1.68 1.68 1.68 1.68 1.68 1.68 1.68	0.83 0.60 0.67 0.81 0.83 0.79 0.84 0.67 0.67 0.67 0.86 0.40 0.89	54% 76% 73% 59% 62% 100% 56% 65% 71% 98% 85% 96%	76% 25% 45% 62% 565% 26% 26% 14% 40%	Set priority Get priority Kill [suspended] thread Yield [no other] thread Resume [suspended low prio] thread Resume [runnable low prio] thread Suspend [runnable] thread Yield [only low prio] thread Suspend [runnable->not runnable] Kill [runnable] thread Destroy [dead] thread Destroy [runnable] thread Resume [high priority] thread Thread switch
0.41 0.48 0.59 0.45 0.45	0.00 0.00 0.00 0.00	1.68 1.68 1.68 1.68	0.61 0.69 0.76 0.65 0.65	75% 71% 64% 73% 73%	71% 64% 73%	Scheduler lock Scheduler unlock [0 threads] Scheduler unlock [1 suspended] Scheduler unlock [many suspended] Scheduler unlock [many low prio]
0.52 0.79 0.84 0.63 0.52 0.58 3.40	0.00 0.00 0.00 0.00 0.00 0.00 3.35	1.68 5.03 5.03 1.68 1.68 1.68 5.03	0.72 0.93 0.94 0.79 0.72 0.76	68% 96% 96% 68% 65%	59% 56% 62% 68% 65%	Init mutex Lock [unlocked] mutex Unlock [locked] mutex Trylock [unlocked] mutex Trylock [locked] mutex Destroy mutex Unlock/Lock mutex
0.99 0.47 0.79 0.42 0.79 0.37 0.73 0.73 0.68 0.68 0.63 0.63 0.63 0.73 3.25	0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00	1.68 1.68 5.03 1.68 1.68 1.68 3.35 1.68 3.35 1.68 1.68 1.68 1.68 1.68	0.81 0.68 0.93 0.63 0.83 0.57 0.87 0.88 0.88 0.87 0.79 0.81 0.44 0.79 0.87	59% 71% 96% 75% 53%% 59% 56% 59% 62% 59% 84% 59% 93%	71% 75% 75% 75% 75% 75% 75% 75% 75% 75% 75	Create mbox Peek [empty] mbox Put [first] mbox Peek [1 msg] mbox Put [second] mbox Peek [2 msgs] mbox Get [first] mbox Get [first] mbox Get [second] mbox Tryput [first] mbox Peek item [non-empty] mbox Tryget [non-empty] mbox Tryget [empty] mbox Tryget [empty] mbox Waiting to get mbox Waiting to put mbox Delete mbox Put/Get mbox
0.63 0.63 0.63 0.52 0.52 0.52 0.21 3.30	0.00 0.00 0.00 0.00 0.00 0.00 0.00	1.68 1.68 1.68 1.68 1.68 1.68 1.68 3.35	0.79 0.79 0.79 0.72 0.72 0.72 0.37 0.10	62% 62% 62% 68% 68% 68% 96%	62% 62% 68% 68% 68% 87%	Init semaphore Post [0] semaphore Wait [1] semaphore Trywait [0] semaphore Trywait [1] semaphore Peek semaphore Destroy semaphore Post/Wait semaphore
0.79 0.42 0.37 0.73 0.63	0.00 0.00 0.00 0.00	3.35 1.68 1.68 1.68	0.88 0.63 0.57 0.82 0.79	56% 75% 78% 56% 62%	75% 78% 56%	Create counter Get counter value Set counter value Tick counter Delete counter
0.89 0.84 0.52 0.89	0.00 0.00 0.00 0.00	3.35 1.68 1.68 3.35	0.89 0.84 0.72 0.89	96% 100% 68% 96%	50% 68%	Create alarm Initialize alarm Disable alarm Enable alarm

```
0.58
            0.00
                    1.68
                             0.76
                                     65% Delete alarm
                                    62% 62% Tick counter [1 alarm]
93% 3% Tick counter [many alarms]
    0.63
            0.00
                    1.68
                             0.79
    5.03
            3.35
                   6.70
                             0.10
    0.94
            0.00
                   1.68
                             0.82
                                     56% 43% Tick & fire counter [1 alarm]
   11.16
           10.06
                   11.73
                             0.76
                                     65% 34% Tick & fire counters [>1 together]
                   6.70
                                     90% 90% Tick & fire counters [>1 separately]
    5.19
            5.03
                             0.28
                    1.68
    0.01
            0.00
                             0.03
                                     99% 99% Alarm latency [0 threads]
                                    92% 92% Alarm latency [2 threads]
53% 45% Alarm latency [many threads]
    0.13
            0.00
                    1.68
                             0.24
    0.94
            0.00
                     3.35
                             0.85
                                     96% 96% Alarm -> thread resume latency
    1.75
            1.68
                     6.70
                             0.15
            0
                  368 (main stack: 1036) Thread stack used (1712 total)
All done, main stack
                         : stack used 1036 size 2912
All done
                     : Interrupt stack used
                                                368 size 4108
All done
                      : Idlethread stack used
                                                288 size 2048
```

Timing complete - 28520 ms total

PASS: <Basic timing OK>

EXIT:<done>

Appendix 2: eCos Licensing

RED HAT ECOS PUBLIC LICENSE Version 1.1

- 1. DEFINITIONS.
- 1.1. "Contributor" means each entity that creates or contributes to the creation of Modifications.
- 1.2. "Contributor Version" means the combination of the Original Code, prior Modifications used by a Contributor, and the Modifications made by that particular Contributor.
- 1.3. "Covered Code" means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof.
- 1.4. "Electronic Distribution Mechanism" means a mechanism generally accepted in the software development community for the electronic transfer of data.
- 1.5. "Executable" means Covered Code in any form other than Source Code.
- 1.6. "Initial Developer" means the individual or entity identified as the Initial Developer in the Source Code notice required by Exhibit A.
- 1.7. "Larger Work" means a work which combines Covered Code or portions thereof with code not governed by the terms of this License.
- 1.8. "License" means this document.
- 1.9. "Modifications" means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:
- A. Any addition to or deletion from the contents $\,$ of a file containing Original Code or previous $\,$ Modifications.
- B. Any new file that contains any part of the Original Code or previous Modifications.
- 1.10. "Original Code" means Source Code of computer software code which is described in the Source Code notice required by Exhibit A as Original Code, and which, at the time of its release under this License is not already Covered Code governed by this License.

- 1.11. "Source Code" means the preferred form of the Covered Code for making modifications to it, including all modules it contains, plus any associated interface definition files, scripts used to control compilation and installation of an Executable, or a list of source code differential comparisons against either the Original Code or another well known, available Covered Code of the Contributor's choice. The Source Code can be in a compressed or archival form, provided the appropriate decompression or dearchiving software is widely available for no charge.
- 1.12. "You" means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License or a future version of this License issued under Section 6.1. For legal entities, "You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of fifty percent (50%) or more of the outstanding shares or beneficial ownership of such entity.
- 1.13. "Red Hat Branded Code" is code that Red Hat distributes and/or permits others to distribute under different terms than the Red Hat eCos Public License. Red Hat's Branded Code may contain part or all of the Covered Code.
- 2. SOURCE CODE LICENSE.
- 2.1. The Initial Developer Grant. The Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:
- (a) to use, reproduce, modify, display, perform, sublicense and distribute the Original Code (or portions thereof) with or without Modifications, or as part of a Larger Work; and
- (b) under patents now or hereafter owned or controlled by Initial Developer, to make, have made, use and sell ("Utilize") the Original Code (or portions thereof), but solely to the extent that any such patent is reasonably necessary to enable You to Utilize the Original Code (or portions thereof) and not to any greater extent that may be necessary to Utilize further Modifications or combinations.
- 2.2. Contributor Grant. Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:
- (a) to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code or as part of a Larger Work; and
- (b) under patents now or hereafter owned or controlled by Contributor, to Utilize the Contributor Version (or portions thereof), but solely to the extent that any such patent is reasonably necessary to enable You to Utilize the Contributor Version (or portions thereof), and not to any greater extent that may be necessary to Utilize further Modifications or combinations.
- 3. DISTRIBUTION OBLIGATIONS.
- 3.1. Application of License. The Modifications which You create or to which You contribute are governed by the terms of this License, including without limitation Section 2.2. The Source Code version of Covered Code may be

distributed only under the terms of this License or a future version of this License released under Section 6.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or restricts the applicable version of this License or the recipients rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.5.

- 3.2. Availability of Source Code. Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available and to the Initial Developer; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party. You are responsible for notifying the Initial Developer of the Modification and the location of the Source if a contact means is provided. Red Hat will be acting as maintainer of the Source and may provide an Electronic Distribution mechanism for the Modification to be made available. You can contact Red Hat to make the Modification available and to notify the Initial Developer.
- 3.3. Description of Modifications. You must cause all Covered Code to which you contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by the Initial Developer and including the name of the Initial Developer in (a) the Source Code, and (b) in any notice in an Executable version or related documentation in which You describe the origin or ownership of the Covered Code.

3.4. Intellectual Property Matters

- (a) Third Party Claims. If You have knowledge that a party claims an intellectual property right in particular functionality or code (or its utilization under this License), you must include a text file with the source code distribution titled "LEGAL" which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If you obtain such knowledge after You make Your Modification available as described in Section 3.2, You shall promptly modify the LEGAL file in all copies You make available thereafter and shall take other steps (such as notifying appropriate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained.
- (b) Contributor APIs. If Your Modification is an application programming interface and You own or control patents which are reasonably necessary to implement that API, you must also include this information in the LEGAL file.
- 3.5. Required Notices. You must duplicate the notice in Exhibit A in each file of the Source Code, and this License in any documentation for the Source Code, where You describe recipients rights relating to Covered Code. If You created one or more Modification(s), You may add your name as a Contributor to the Source Code. If it is not possible to put such notice in a particular Source Code file due to its structure, then you must include such notice in a location (such as a relevant directory file) where a user would be likely to look for such a notice. You may choose to offer, and to charge a fee for, warranty,

support, indemnity or liability obligations to one or more recipients of Covered Code. However, You may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

- 3.6. Distribution of Executable Versions. You may distribute Covered Code in Executable form only if the requirements of Section 3.1-3.5 have been met for that Covered Code, and if You include a notice stating that the Source Code version of the Covered Code is available under the terms of this License, including a description of how and where You have fulfilled the obligations of Section 3.2. The notice must be conspicuously included in any notice in an Executable version, related documentation or collateral in which You describe recipients rights relating to the Covered Code. You may distribute the Executable version of Covered Code under a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable version does not attempt to limit or alter the recipients rights in the Source Code version from the rights set forth in this License. If You distribute the Executable version under a different license You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or any Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer. If you distribute executable versions containing Covered Code, you must reproduce the notice in Exhibit B in the documentation and/or other materials provided with the product.
- 3.7. Larger Works. You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.
- 4. INABILITY TO COMPLY DUE TO STATUTE OR REGULATION.

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Code due to statute or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; (b) cite the statute or regulation that prohibits you from adhering to the license; and (c) describe the limitations and the code they affect. Such description must be included in the LEGAL file described in Section 3.4 and must be included with all distributions of the Source Code. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it. You must submit this LEGAL file to Red Hat for review, and You will not be able use the covered code in any means until permission is granted from Red Hat to allow for the inability to comply due to statute or regulation.

5. APPLICATION OF THIS LICENSE.

This License applies to code to which the Initial Developer has attached the notice in Exhibit A, and to related Covered Code.

Red Hat may include Covered Code in products without such additional products becoming subject to the terms of this License, and may license such additional products on different terms from those contained in this License. Red Hat may

license the Source Code of Red Hat Branded Code without Red Hat Branded Code becoming subject to the terms of this License, and may license Red Hat Branded Code on different terms from those contained in this License. Contact Red Hat for details of alternate licensing terms available.

6. VERSIONS OF THE LICENSE.

- 6.1. New Versions. Red Hat may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number.
- 6.2. Effect of New Versions. Once Covered Code has been published under a particular version of the License, You may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by Red Hat. No one other than Red Hat has the right to modify the terms applicable to Covered Code beyond what is granted under this and subsequent Licenses.
- 6.3. Derivative Works. If you create or use a modified version of this License (which you may only do in order to apply it to code which is not already Covered Code governed by this License), you must (a) rename Your license so that the phrases "ECOS", "eCos", "Red Hat", "RHEPL" or any confusingly similar phrase do not appear anywhere in your license and (b) otherwise make it clear that your version of the license contains terms which differ from the Red Hat eCos Public License. (Filling in the name of the Initial Developer, Original Code or Contributor in the notice described in Exhibit A shall not of themselves be deemed to be modifications of this License.)

7. DISCLAIMER OF WARRANTY.

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS DISCLAIMER.

8. TERMINATION.

This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. All sublicenses to the Covered Code which are properly granted shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

9. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH

PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THAT EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

10. U.S. GOVERNMENT END USERS.

The Covered Code is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with only those rights set forth herein.

11. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by California law provisions (except to the extent applicable law, if any, provides otherwise), excluding its conflict-of-law provisions. With respect to disputes in which at least one party is a citizen of, or an entity chartered or registered to do business in, the United States of America: (a) unless otherwise agreed in writing, all disputes relating to this License (excepting any dispute relating to intellectual property rights) shall be subject to final and binding arbitration, with the losing party paying all costs of arbitration; (b) any arbitration relating to this Agreement shall be held in Santa Clara County, California, under the auspices of JAMS/EndDispute; and (c) any litigation relating to this Agreement shall be subject to the jurisdiction of the Federal Courts of the Northern District of California, with venue lying in Santa Clara County, California, with the losing party responsible for costs, including without limitation, court costs and reasonable attorneys fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License.

12. RESPONSIBILITY FOR CLAIMS.

Except in cases where another Contributor has failed to comply with Section 3.4, You are responsible for damages arising, directly or indirectly, out of Your utilization of rights under this License, based on the number of copies of Covered Code you made available, the revenues you received from utilizing such rights, and other relevant factors. You agree to work with affected parties to distribute responsibility on an equitable basis.

13. ADDITIONAL TERMS APPLICABLE TO THE RED HAT ECOS PUBLIC LICENSE.

Nothing in this License shall be interpreted to prohibit Red Hat from licensing under different terms than this License any code which Red Hat otherwise would have a right to license.

Red Hat and logo - This License does not grant any rights to use the trademark Red Hat, the Red Hat logo, eCos logo, even if such marks are included in the Original Code. You may contact Red Hat for permission to display the Red Hat and eCos marks in either the documentation or the Executable version beyond

that required in Exhibit B.

Inability to Comply Due to Contractual Obligation - To the extent that Red Hat is limited contractually from making third party code available under this License, Red Hat may choose to integrate such third party code into Covered Code without being required to distribute such third party code in Source Code form, even if such third party code would otherwise be considered "Modifications" under this License.

EXHIBIT A.

The contents of this file are subject to the Red Hat eCos Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.redhat.com

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is eCos - Embedded Configurable Operating System, released September 30, 1998.

The Initial Developer of the Original Code is Red Hat. Portions created by Red Hat are Copyright (C) 1998, 1999, 2000 Red Hat, Inc. All Rights Reserved.

EXHIBIT B.

Part of the software embedded in this product is eCos - Embedded Configurable Operating System, a trademark of Red Hat. Portions created by Red Hat are Copyright (C) 1998, 1999, 2000 Red Hat, Inc. (http://www.redhat.com) All Rights Reserved.

THE SOFTWARE IN THIS PRODUCT WAS IN PART PROVIDED BY RED HAT AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Appendix 3: The eCos Copyright Assignment Form, Revision 1.1

Rationale

This preamble describes how to use the standard **eCos** copyright assignment form. The rationale behind this assignment is to avoid any possible confusion over the legal ownership of **eCos**, and to indemnify Red Hat and all **eCos** users against copyright or patent claims on contributed code used within **eCos**. Red Hat would be especially vulnerable, but all users and their **eCos** based applications could be affected.

All contributions to **eCos** for which there are copyright assignments will be covered by the Red Hat **eCos** public license. The license provides a guarantee that the contribution will remain freely available to all.

This agreement gives Red Hat ownership of your changes but promises that you will retain the right to use your contributed changes as you see fit.

Because employers often can claim ownership over things that employees write, you may also have to get your employer to sign a disclaimer that says that they have no claim to the changes you are contributing.

Please read everything, and if you have any questions, email

ecos-assign@redhat.com

for help.

Thanks for your contribution to eCos!

How to assign copyright

The way to assign copyright to Red Hat is to sign an assignment contract. This is what makes Red Hat the legal copyright holder, so that Red Hat can register the copyright on the new version.

If you are employed as a programmer (even at a university), or have made an agreement with your employer or school that gives them ownership of the software you write, then Red Hat needs a signed letter from your employer disclaiming rights to the contributed software.

The disclaimer should be printed on the company's headed paper, and signed by an officer of the company, or someone authorized to license the company's intellectual property. Here is an example of wording that can be used for this purpose:

<INSERT COMPANY NAME> hereby disclaims all copyright interest in the changes and enhancements made by <INSERT YOUR NAME> to **eCos**, including any future revisions of these changes and enhancements.

<INSERT COMPANY NAME> affirms that it has no other intellectual property interest that would undermine this release, or the use of **eCos**, and will do nothing to undermine it in the future.

<INSERT SIGNATURE OF OFFICER OF COMPANY>,

<INSERT DATE>

<INSERT PRINTED NAME OF OFFICER OF COMPANY>

<INSERT TITLE OF OFFICER>

If your employer says they do have an intellectual property claim that could conflict with the use of the program, then please contact Red Hat to discuss possible next steps.

Below is the usual assignment contract. You need to edit and replace <INSERT NAME OF CONTRIBUTOR> with your full name. Please print a copy, sign, date, and mail it to:

Legal Department (eCos Assignments) Red Hat, Inc. 2600 Meridian Parkway NC 27713 USA

Don't forget to include the original signed copy of the employer's disclaimer.

Please try to print the whole first page of the form on a single piece of paper. If it doesn't fit on one printed page, put it on two sides of a single piece of paper, and attach the second page of the form. Please write the date using letters rather than numbers to avoid any confusion due to international day/month ordering conventions.

Note: This text is also available in the **eCos** software distribution, in the file assign.txt.

----- Cut Here ------

eCos ASSIGNMENT

For good and valuable consideration, receipt of which I acknowledge, I, <INSERT NAME OF CONTRIBUTOR>, hereby transfer to Red Hat, Inc. my entire right, title, and interest (including all rights under copyright) in my changes and enhancements to eCos, subject to the conditions below. These changes and enhancements are herein called the "Work". The Work hereby assigned shall also include any future revisions of these changes and enhancements hereafter made by me.

Upon thirty days prior written notice, Red Hat agrees to grant me non-exclusive rights to use the Work (i.e. just my changes and enhancements, not eCos as a whole) as I see fit; (and Red Hat's rights shall otherwise continue unchanged).

I hereby agree that if I have or acquire hereafter any patent or interface copyright or other intellectual property interest dominating the software enhanced by the Work (or use of that software), such dominating interest will not be used to undermine the effect of this assignment, i.e. Red Hat and the general public will be licensed to use, in that program and its derivative works, without royalty or limitation, the subject matter of the dominating interest. This license provision will be binding on my heirs, assignees, or other successors to the dominating interest, as well as on me.

I hereby represent and warrant that I am the sole copyright holder for the Work and that I have the right and power to enter into this contract. I hereby indemnify and hold harmless Red Hat, its officers, employees, and agents against any and all claims, actions or damages (including attorney's reasonable fees) asserted by or paid to any party on account of a breach or alleged breach of the foregoing warranty. I make no other express or implied warranty (including without limitation, in this disclaimer of warranty, any warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE).

[Print Name]
Date: [Please write using letters]
For Red Hat:

Agreed: [signature]

Date:

[Please print your name here]

[For the copyright registration, of what country are you a citizen?]

[In what year were you born?]

[Please write your email address here]

[Please write your address here, so we can mail a signed copy of the agreement back to you]

[Which files have you changed so far, and which new files have you written so far?]

Index

A-C

```
alarms 74, 77
  initializing 77
applications
  running on target 40
  sample 69
building from source 49
CDL 20
chips, supported 15
clocks 74, 76
command notation, GDB and GCC 16
component definition language (CDL) 20
configurability 19
Configuration Tool 19, 50
configuring from source 49
connecting to a simulator 37
connecting via Ethernet 37
connecting via serial 36
counters 77
CygWin 31
```

D-F

```
Delayed Service Routine (DSR) 77
delayed service routines 77
Developer's Kit
bundled with GNUPro Toolkit 22
device drivers
serial 21
directory and file system conventions 17
disk space requirements 24
DSR (Delayed Service Routine) 77
eCos Net Release 23
ecosconfig 56
available options 56
list of targets 56
embedded kernel 19
```

Ethernet connection 37
examples
ecosconfig list of available targets 56
ecosconfigl list of available options 56
hello world program 69
program that creates an alarm 75
two-threaded program 70
file system and directory conventions 17
G-I
G++ 22
GH+ 22 GCC 22
command notation 16
GDB 22
command notation 16
GNUPro compiler 31
GNUPro Toolkit 22
HAL 18
Hardware Abstraction Layer (HAL) 18
hardware setup 36
host operating systems, supported 15
I/O library 21
i386 hardware setup 39
installation instructions
Linux 33
Solaris 34
unix 32
Windows 31
ISO C library 20
J-L
kernel, embedded 19
kernel, real-time
features 19
libraries
ISO C 20
math 20
standard I/O 21
Linux
13Xb synthetic target

setup 39 system requirements 25 Linux installation 33

M-O

math library 20
measuring
sample numbers 80
mutex 72
Net release 31
Net site 22
overview of the release 18

P-R

packages 23
performance
sample numbers 80
problem reports, submitting 26
programming tutorial 43
Running Applications on the Target 40

S-U

sample programs hello.c 69 simple-alarm.c 74 twothreads.c 70 serial connection 36 serial device drivers 21 setup, target 36 simulator delays, as compared with hardware 72,76simulator, connecting to 37 Solaris installation 34 supported host operating systems 15 target microprocessors 15 target platforms 15 system performance sample numbers 80 system requirements 24 target

selecting with ecosconfig 64
supported microprocessors 15
supported platforms 15
target setup 36
test suites 66
UNIX
ecosconfig 56
UNIX installation instructions 32

V-Z

 $\begin{array}{c} {\rm Windows} \\ {\rm Configuration\ Tool\ 50} \\ {\rm Windows\ installation\ instructions\ 31} \end{array}$