

Application Note

USING eCOS[™] ON THE EP72XX DEVELOPMENT BOARDS





Note: Cirrus Logic assumes no responsibility for the attached information which is provided "AS IS" without warranty of any kind (expressed or implied).



TABLE OF CONTENTS

1.		3
2.	REQUIREMENTS	3
3.	LOADING THE GDB ROM IMAGE INTO ON-BOARD FLASH	3
4.	BUILD ECOS CONFIGURATION FOR YOUR APPLICATION PROGRAM	
	USING THE CONFIGURATION TOOL.	5
5.	COMPILE YOUR PROGRAM INTO AN EXECUTABLE USING GCC	8
6.	RUNNING THE BINARY EXECUTABLE	9
	6.1 Executing your New Program through the eCos Configuration Tool.	9
	6.2 Executing your New Program through gdb and the eCos Development Environment	10
	6.3 Executing your New Program through gdb using Multi-Ice™	10
7.	DOWNLOADING EXECUTABLE IMAGES INTO FLASH MEMORY (OPTIONAL)	12

LIST OF FIGURES

Figure 1. Selecting the Cirrus Pre-Built Template	. 5
Figure 2. Selecting Startup Type to Enable User Programs to Download	. 6
Figure 3. Setting ARM Architecture Options	. 6
Figure 4. Build Tools Dialog Box	.7
Figure 5. User Tools Dialog Box	.7
Figure 6. Build Library Menu Selection	.7
Figure 7. Build Tests Menu Selection	. 8
Figure 8. Run Tests Window	. 9
Figure 9. Dialog Box to Acknowledge that the Target Has Been Reset	. 9
Figure 10. Screen Shot of Executed Code1	10
Figure 11. Multi-ICE Dialog Box 1	11
Figure 12. Screen Shot of Multi-Ice-gdb-Server Starting1	11

Contacting Cirrus Logic Support

For a complete listing of Direct Sales, Distributor, and Sales Representative contacts, visit the Cirrus Logic web site at: http://www.cirrus.com/corporate/contacts/sales.cfm

Preliminary product information describes products which are in production, but for which full characterization data is not yet available. Advance product information describes products which are in development and subject to development changes. Cirrus Logic, Inc. has made best efforts to ensure that the information contained in this document is accurate and reliable. However, the information is subject to change without notice and is provided "AS IS" without warranty of any kind (express or implied). No responsibility is assumed by Cirrus Logic, Inc. for the use of this information, nor for infringements of patents or other rights of third parties. This document is the property of Cirrus Logic, Inc. and implies no license under patents, copyrights, trademarks, or trade secrets. No part of this publication may be copied, reproduced, stored in a retrieval system, or transmitted, in any form or by any means (electronic, mechanical, photographic, or otherwise) without the prior written consent of Cirrus Logic, Inc. Items from any Cirrus Logic website or disk may be printed for use by the user. However, no part of the printout or electronic files may be copied, reproduced, stored in a retrieval system, or transmitted, in any form or by any means (electronic, mechanical, photographic, or otherwise) without the prior written consent of Cirrus Logic, Inc. Furthermore, no part of this publication may be used as a basis for manufacture or sale of any items without the prior written consent of Cirrus Logic, Inc. The names of products of Cirrus Logic, Inc. or other vendors and suppliers appearing in this document may be trademarks or service marks of their respective owners which may be registered in some jurisdictions. A list of Cirrus Logic, Inc. trademarks and service marks can be found at http://www.cirrus.com.



1. INTRODUCTION

This document explains the steps involved in downloading and executing Cirrus Logic sample programs with the EP72XX Development Kit using eCos with or without Multi-Ice. Two Cirrus Logic sample programs were used to verify functionality of the EP72XX Development Kit using GNU tools to compile, load, and execute on the EP72XX Development Kit. The sample programs used were the keyboard and the screen sample programs and the development board used was the EDB7211-2.

The procedures contained in this document describe how to use the eCos and the GNU toolchain to run tests and your own applications on EP72XX Development Kit. It is assumed that eCos version 1.3 or higher has been installed on your Windows OS along with the GNU toolchain.

2. REQUIREMENTS

The following components are needed to successfully download and execute programs:

- 1) A host PC running Windows NT.
- 2) Program capable of downloading to on-board FLASH. (Contained on the Cirrus Logic demo CD ROM)
- 3) A Cirrus Logic EDB72xx Development Kit containing:
 - a) EDB72xx development board.
 - b) Null modem cable.
 - c) OrCad 7.2 and PDF board schematics.
 - d) Documentation on CD ROM.
 - e) 83-key QWERTY keyboard
 - f) LCD panel
- 4) eCos toolkit available from any of the following resources:
 - a) http://sources.redhat.com
 - b) For a demo Cirrus Logic CD ROM copy containing complete set of tools, contact:
 - Elizabeth Castiglioni Phone: 512-912-3070 Email: ecastig@crystal.cirrus.com

3. LOADING THE GDB ROM IMAGE INTO ON-BOARD FLASH

- 1) Connect the supplied NULL modem cable between the SERIAL PORT 0 connector on the evaluation board and the COM port 1 on the host system.
- 2) Place a jumper on jumper JP2.
- 3) Apply power to the evaluation board.
- 4) From an MS DOS prompt, run download.exe found on the Cirrus Logic EP72XX Development Kit CD:



Example:

download `filename'

where 'filename' is either

\Your eCos Directory\loaders\arm-edb72XX\edb72XX_gdb_module.bin

or

\Your eCos Directory\loaders\arm-edb72XX\edb72XX_cygmon.bin

- Notes: The file edb72XX_gdb_module.bin is a FLASH ROM image that provides a remote GDB stub only. The file edb72XX_cygmon.bin is a FLASH ROM image which provides a port of the CygMon ROM monitor, which includes a command-line interface and a GDB remote stub monitor with basic program handling and debugging commands.
- 5) Press the **Reset** button on the evaluation board.
- 6) Press the **Wakeup** button on the evaluation board.
- 7) Download.exe will display its progress as it programs the NOR FLASH. Wait until it says it is done.
- 8) Remove power from the evaluation board.
- 9) Remove the jumper from jumper JP2.

At this point the new boot code is programmed into the NOR FLASH. It can then be used in the normal operation of the board



4. BUILD ECOS CONFIGURATION FOR YOUR APPLICATION PROGRAM USING THE CONFIGURATION TOOL.

- 1) Prepare the build for the Cirrus Logic Development Board.
 - a) Start the configuration tool by selecting the menu sequence, $Start \rightarrow Programs \rightarrow Red$ Hat $eCos \rightarrow Configuration$ Tool.
 - b) Predefined templates are provided to run eCos on Cirrus Logic development boards. Choose the Cirrus Logic predefined templates by selecting **Build**→**Templates**. (See Figure 1)
 - c) Set the Hardware option to Cirrus Logic development board.
 - d) Set the **Packages** pull down menu to the appropriate value depending on your needs.
- 2) Configure options for building eCos configuration files. (See Figure 2 on page 6)
 - a) Click on the + **Serial device drivers**.
 - b) Check the box labeled **TTY mode serial device drivers**.
 - c) Check the + **ARM EDB7XXX serial device drivers** box.
 - d) Check the first box labeled **Cirrus_Logic EDB7XXX serial port 1_driver**.
 - e) Click on the buttons that were opened to get back to the original window.
 - f) Next click on the + eCos HAL button.
 - g) Next click on the + **ARM architecture** button.
 - h) Next click on the + Cirrus Logic development board button.
 - i) For the **Startup type**, change to "RAM" to allow user programs to be loaded into DRAM. See Figure
 - j) Create a new directory and save the configuration setup. *File -> Save As*.

Directories will be created on the same level as your new configuration file. These include: <filename>_build, <filename>_install, and <filename>_mlt. These directories contain all the files built to support your new program on the platform you have chosen.



Figure 1. Selecting the Cirrus Pre-Built Template





Figure 2. Selecting Startup Type to Enable User Programs to Download

	default.ecc - eCos Configuration Tool				
	<u>File Edit View Build Tools H</u> elp				
	🖃 🧰 Configuration	Property			
	🕀 🧰 Global build options	URL			
	🕀 🧰 CygMon HAL options	Macro			
	🕀 🚼 Serial device drivers	v1_3			
	🖃 🚼 eCos HAL	v1_3			
	🕀 📄 Platform-independent HAL options				
	🕀 🧰 HAL interrupt handling	Doc			
	HAL context switch support	Compile			
	E Source-level debugging support	Require:			
	ROM monitor support				
	E 🚰 ARM architecture	v1_3			
	Enable Thumb instruction set				
	Use big-endian mode				
	ARM CPU family	ARM7			
	Provide diagnostic dump for exceptions				
	Process all exceptions with the eCos app				
	Support GDB thread operations via ICE/N				
	Linker script	src/arm.ld			
Development	Cirrus Logic development board	v1_3 This optic			
Development	Cirrus Logic processor variant	EP7211			
Board 🚍	Startup type	RAM			
Options	Diagnostic serial port	0			
optionio	Diagnostic serial port baud rate	38400			
	GDB serial port	0			
	GDB serial port baud rate	38400			
	Processor clock rate	73728			
	■ Installed DRAM on board	16			
	Perform DRAM refresh in software				
	LCD installed				
	🖃 🧰 Real-time clock constants				
	🗉 🧰 Cirrus Logic build options				
		arm_edb7211_ram			
	🕀 🚼 I/O sub-system	v1_3			
	🕀 🚼 Infrastructure	v1_3			
	↓	v1_3			

Figure 3. Setting ARM Architecture Options



3) Link the required tools and build the application.

Normally the installation process will supply the information required for the eCos Configuration Tool to locate the build tools (compiler, linker, etc...) necessary to perform a build. However if this information is not registered, or it is necessary to specify the location manually (for example, when a new toolchain installation has been made).

Normally the installation process will supply the information required for the eCos Configuration Tool to locate the user tools (cat, ls, etc...) necessary to perform a build. However if this information is not registered, or it is necessary to specify the location manually (for example, when a new toolchain installation has been made).

b) Select: **Tools** \rightarrow **Paths** \rightarrow **User Tool**. The dialog box in Figure 5 will appear.

4) Next, build the library. Building the Library will cause the eCos configuration to be created. The result of a successful build will be (among other things) a library against which user code can be linked. The dialog box in Figure 6 will appear.



Figure 4. Build Tools Dialog Box



Figure 5. User Tools Dialog Box

<u>B</u> uil	d <u>T</u> ools	<u>H</u> elp
**	Library	F7
	<u>T</u> ests	Shift+F7
c .	<u>C</u> lean	
Ł	<u>S</u> top	
č A	Options	
ē	<u>R</u> epositor	у
5	Te <u>m</u> plate:	s
-	Package:	s

Figure 6. Build Library Menu Selection



To do this, select **Build**→**Library** (this can take up to 20 minutes to complete).

5) When complete, build the tests. Building the tests will cause the eCos configuration to be created. Additionally, this builds the relevant test cases linked against the eCos library. Approximately 170 selectable tests will be created to test your configuration.

To do this, select **Build** \rightarrow **Tests** (this can take up to 20 minutes to complete). See Figure 7.

5. COMPILE YOUR PROGRAM INTO AN EXECUTABLE USING GCC

- 1) Start a Cygwin bash shell under Windows by selecting **Start**→**Programs**→**Red Hat eCos**→**eCos Development Environment**
- 2) Change to the directory where your application source code resides.
- 3) Now compile by typing:

```
arm-elf-gcc -g -o `filename'.exe -I/'filename'_install/include `filename'.c
-L/'filename'_install/lib -T'filename'_install/lib/target.ld -nostdlib
```

- Note: If there are spaces and/or special characters in your path, the "Something not found" error will occur while compiling the program. In the case, you have to copy the include and lib directories elsewhere, and redefine the paths for -I and -L parameters.
- 4) If all goes well, the only indication of proper compilation is a return prompt. Otherwise, error messages will occur. Fix the error messages and recompile.

Now your new binary executable file should exist, filename.exe, and the program can be run by either executing through the eCos configuration tool, Section 6.1, through the eCos Development Environment, Section 6.2, or through using gdb and Multi-Ice, Section 6.3.

<u>B</u> uild	<u>T</u> ools	<u>H</u> elp
🔛 L	ibrary .	F7
]	ests	Shift+F7
<u>[</u>	lean	
👗 3	itop	
<u> </u>	<u>)</u> ptions	
E	<u>R</u> epositor	y
i 1	emplate:	s
E	Packages	s

Figure 7. Build Tests Menu Selection

6. RUNNING THE BINARY EXECUTABLE

Once your binary executable exists, it can be executed 3 different ways. The following three steps show how to 1) execute with the eCos Configuration tool, 2) use a bash shell and gdb, and 3) how to implement using Multi-Ice.

6.1 Executing your New Program through the eCos Configuration Tool.

- a) Execute your test under eCos configuration tool by selecting **Tools→Run Test**. See Figure 8
- b) To run ONLY the application program you created, choose the button Uncheck All
- c) Click the **Add...** button.
- d) Find your binary executable program, <filename>.exe, created in the last series of tests, select it, and click the **Open** button.
- e) Insure that the box is checked in front of your newly added test.
- f) Click the **Run** button and the dialog in Figure 10 will appear:
- g) Make sure the evaluation board is turned on and that the **Wakeup** and **Reset** buttons have been pushed. The green LED on the development board should be on.
- h) Click the **OK** button.
- i) Switch to the **Output** tab to see the application program being loaded and run.

C <u>h</u> eck All	<u>U</u> ncheck All	<u>A</u> dd	Add from Folder	<u>R</u> emove
C:\jstemp\johns\	Lygnus\ecos\defaul	t\default_install\tests	<pre>\compat\uitron\v1_3\</pre>	tests\testcxb.exe
✓ L:\jstemp\johns\	Lygnus\ecos\defaul	t\default_install\tests	<pre>\compat\uitron\v1_3\</pre>	tests\test2.exe
C:\jstemp\johns\	Cygnus\ecos\defaul	t\default_install\tests\	\language\c\libm\v1_	3\tests\vectors\-
C:\jstemp\johns\	Cygnus\ecos\defaul	t\default_install\tests\	\language\c\libc\v1_	3\tests\stdlib\qsc
C:\jstemp\johns\	Cygnus\ecos\defaul	t\default_install\tests ^v	\language\c\libc\v1_	3\tests\ctype\cty
✓C:\jstemp\johns\	Cygnus\ecos\defaul	t\default_install\tests ^v	<pre>\compat\uitron\v1_3\</pre>	tests\testintr.exe
✓C:\jstemp\johns\	.Cygnus\ecos\defaul	t\default_install\tests\	<pre>\compat\uitron\v1_3\</pre>	tests\testcx7.exe
✓C:\jstemp\johns\	.Cygnus\ecos\defaul	t\default_install\tests\	<pre>\compat\uitron\v1_3\</pre>	tests\test3.exe
■C:\jstemp\johns\	.Cygnus\ecos\defaul	t\default_install\tests ^v	\language\c\libc\v1_	3\tests\time\time
✓C:\jstemp\johns\	.Cygnus\ecos\defaul	t\default_install\tests ^v	\language\c\libc\v1_	3\tests\string\strl
✓C:\jstemp\johns\	Cygnus\ecos\defaul	t\default_install\tests ^v	\language\c\libc\v1_	3\tests\string\stro
✓C:\jstemp\johns\	Cygnus\ecos\defaul	t\default_install\tests ^v	\language\c\libc\v1_	3\tests\string\me
✓C:\jstemp\johns\	.Cygnus\ecos\defaul	t\default_install\tests ^v	language\c\libc\v1_	3\tests\stdlib\ato
✓C:\jstemp\johns\	Cygnus\ecos\defaul	t\default_install\tests ^v	language\c\libc\v1_	3\tests\stdio\ssc
C:\jstemp\johns\	.Cygnus\ecos\defaul	t\default_install\tests ^v	kernel/v1_3/tests/st	ress_threads.exe
C:\jstemp\johns\	.Cygnus\ecos\defaul	t\default_install\tests ^v	kernel\v1_3\tests\m	emvar1.exe
C:\jstemp\johns\	Cygnus\ecos\defaul	t\default_install\tests ^v	kernel\v1_3\tests\ml	box1.exe
C:\jstemp\johns\	Cygnus\ecos\defaul	t\default_install\tests ^v	kernel\v1_3\tests\ki	ntr0.exe
C:\jstemp\johns\	Cygnus\ecos\defaul	t\default_install\tests ^v	compat/uitron/v1_3	tests\testcx8.exe
C:\jstemp\johns\	Cygnus\ecos\defaul	t\default_install\tests ^v	compat/uitron/v1_3	tests\test4.exe
C:\jstemp\johns\	Cygnus\ecos\defaul	t\default_install\tests ^v	language\c\libm\v1	3\tests\vectors\:
C:\jstemp\johns\	.Cygnus\ecos\defaul	t\default_install\tests ^v	language\c\libm\v1	3\tests\vectors\l
C:\istemp\iobns\	Currous\ecos\defaul			3\tests\string\stra

Figure 8. Run Tests Window

Configuration Tool	×	
Press OK when target is reset - cancel to abort run		
OK	Cancel	

Figure 9. Dialog Box to Acknowledge that the Target Has Been Reset



6.2 Executing your New Program through gdb and the eCos Development Environment

- 1) In the bash shell, go to the directory containing your binary executable.
- 2) Type the following commands to load and execute your program:

```
$ arm-elf-gdb -nw `filename'.exe
```

- (gdb) set remotebaud 38400
- (gdb) target remote coml
- (gdb) load

```
(gdb) continue
```

A sample screen shot is shown in Figure 10.

6.3 Executing your New Program through gdb using Multi-Ice[™]

- 1) In a bash shell, go to the directory containing your binary executable.
- 2) Start Multi-ICE server v1.3 or higher: by Selecting Start→Programs→Multi-ICE 1.3→Multi-ICE server.
- 3) You will need to set up a Multi-ICE Server configuration file for your hardware if you already haven't done so. Please see the following Web site.

http://sources.redhat.com/ecos/docs-latest/tutorials/arm/ecos-tutorial.d.html - pgfId=2562244

The Red Hat Web site contains configuration file examples for supported target platforms. Once there, search for the following topic: Developing eCos Programs with the ARM Multi-ICE. A sample configuration file is shown there.

4) Type the following command in the bash shell:

```
multi-ice-gdb-server.exe --byte-sex l --config-dialog &
```

Auto -rw-rr 1 500 everyone 843 Nov 6 1998 main.c -rw-rr 1 500 everyone 7496 May 16 17:01 vectors.o -rw-rr 1 500 everyone 54 May 19 15:38 *\$README.txt BASN.EXE-2.02\$ arm-elf-gdb -nw keyboard.exe GNU gdb 4.18-ecos-99r1-991015 Copyright 1998 Free Software Foundation, Inc. GDB is free software, covered by the GNU General Public License, and you are welcome to change it and/or distribute copies of it under certain conditions. Type "show copying" to see the conditions. This version of GDB is supported for customers of Eygnus Solutions. Type "show warranty" for details. This GDB was configured as "host=i686-cygwin32target=arm-elf" (gdb) set remotebaud 38400 (gdb) target remote com1 Remote debugging using com1 Oxe0D06808 in ?? () (gdb) load Loading section .rom_vectors, size 0x60 Ima 0x8000 Loading section .rodata, size 0xb8c Ima 0x13edc Loading section .data, size 0xb8c Ima 0x14e68 Start address 0x8060 , load size 53444 Transfer rate: 26722 bits/sec. (gdb) continue	📸 arm-elf-gdb
-rw-rr 1 500 everyone 843 Nov 6 1998 main.c -rw-rr 1 500 everyone 7496 May 16 17:01 vectors.o -rw-rr 1 500 everyone 54 May 19 15:38 *\$README.txt BASH.EXE-2.02\$ arm-elf-gdb -nw keyboard.exe GNU gdb 4.18-eccs-99r1-991015 Copyright 1998 Free Software Foundation, Inc. GDB is free software, covered by the GNU General Public License, and you are welcome to change it and/or distribute copies of it under certain conditions. Type "show copying" to see the conditions. This version of GDB is supported for customers of Cygnus Solutions. Type "show warranty" for details. This GDB was configured as "host=i686-cygwin32target=arm-elf" (gdb) set remotebaud 38400 (gdb) target remote com1 Remote debugging using com1 DxeDD06880 in ?? () (gdb) load Loading section .rom_vectors, size Dx60 Ima Dx8000 Loading section .rodata, size Dxb8c Ima Dx13edc Loading section .data, size Dx65c Ima Dx13edc Loading section .data, size Dx65c Ima Dx14a68 Start address Dx8060, load size 53444 Transfer rate: 26722 bits/sec. (gdb) continue	
continuing.	-rw-rr 1 500 everyone 843 Nov 6 1998 main.c -rw-rr 1 500 everyone 7496 May 16 17:01 vectors.o -rw-rr 1 500 everyone 54 May 19 15:38 *\$README.txt BASH.EXE-2.02\$ arm-elf-gdb -nw keyboard.exe GOW gdb 4.18-ecos-9971-991015 Copyright 1998 Free Software Foundation, Inc. GDB is free software, covered by the GNU General Public License, and you are welcome to change it and/or distribute copies of it under certain conditions. Type "show copying" to see the conditions. This version of GDB is supported for customers of Cygnus Solutions. Type "show warranty" for details. This GDB was configured as "host=i686-cygwin32target=arm-elf" (gdb) set remotebaud 38400 (gdb) target remote com1 Remote debugging using com1 Dxe0006880 in ?? () (gdb) load Loading section .rom_vectors, size 0x60 Ima 0x8000 Loading section .text, size 0xb67c Ima 0x13edc Loading section .data, size 0xb65c Ima 0x1468 Start address 0x8060 , load size 53444 Iransfer rate: 26722 bits/sec. (gdb) continue Continue





Notes: The command line options mentioned above are explained below.:

-byte-sex I

Tells GDB to respond with Little-Endian. Some GDB responses are byte-order (endianess) sensitive. The default here is Big Endian, but since eCos typically runs the hardware in Little Endian mode, this needs to be specified explicitly.

-config-dialog

Forces the GDB server to bring up the Multi-ICE configuration dialog when starting. This is required in order to force the Multi-ICE server to connect properly.

When the screen shown in Figure 11 appears, insure the information is correct and click the **OK** button.

The sample screen shot in Figure 12 shows Multi-Ice-gdb-server starting.

ARM MULTI-ICE Release 1.3	×
localhost	
Processor Driver Selection	
	<u>D</u> etails
TAP 0: ARM720T	Show All
Connection Name	
- Champel) (iourore	
	_
ThumbCV.dll	<u>A</u> dd
	<u>R</u> emove
	Cancel

Figure 11. Multi-ICE Dialog Box



Figure 12. Screen Shot of Multi-Ice-gdb-Server Starting



For more information on multi-ice-gdb-server, go to the following:

http://sources.redhat.com/ecos/docs-1.3.1/ref/gnupro-ref/arm/ARM_COMBO_ap02.html

Type the following commands to load and execute your program:

```
$ arm-elf-gdb -nw `filename'.exe
(gdb) target remote localhost:2331
(gdb) load
```

(at this point, the 'Busy' LED on the Multi-ICE port should turn on)

```
(gdb) continue
```

7. DOWNLOADING EXECUTABLE IMAGES INTO FLASH MEMORY (OPTIONAL)

Follow the procedure outlined in "Loading the GDB ROM Image into On-board Flash" on page 3 of this document with the following exceptions:

- a) When setting up the eCos HAL portion, configure eCos for 'ROM' startup instead of RAM.
- b) In "Compile your program into an executable using gcc" on page 8, build your application without the debugger option -g.
- c) Type the following command to produce a binary format image:
- d) \$ arm-elf-objcopy -O binary 'filename'.exe 'application'.bin
- e) Downloaded <application>.bin into flash memory by following the steps described in Section "Loading the GDB ROM Image into On-board Flash" on page 3.



• Notes •

