AN191

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).

P.O. Box 17847, Austin, Texas 78760
(512) 445 7222 FAX: (512) 445 7581
http://www.cirrus.com

TABLE OF CONTENTS

LIST OF FIGURES

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

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→Programs→Red Hat eCos→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.

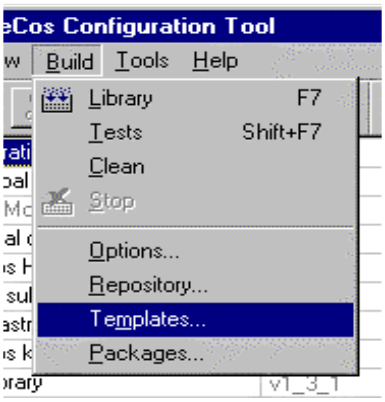

Figure 1. Selecting the Cirrus Pre-Built Template

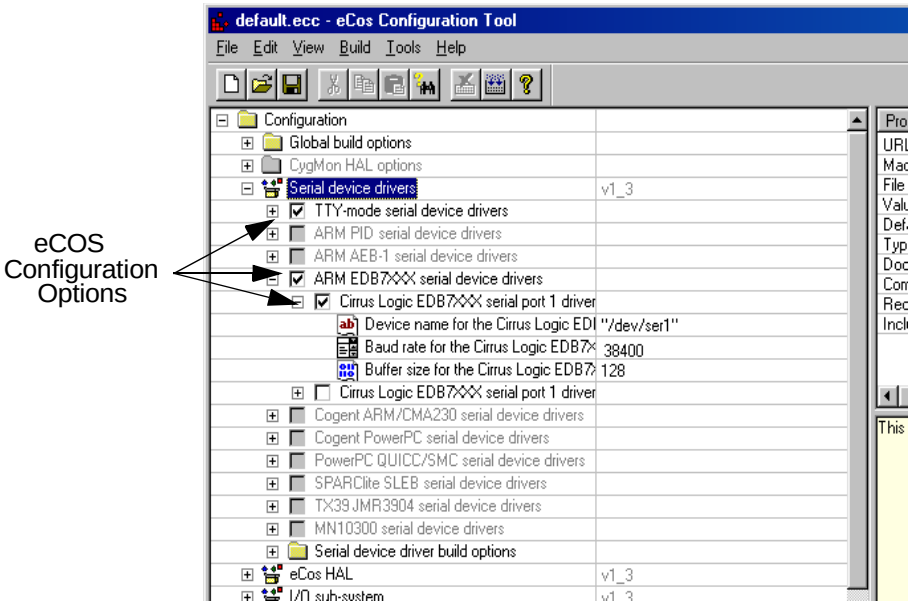

Figure 2. Selecting *Startup Type* to Enable User Programs to Download

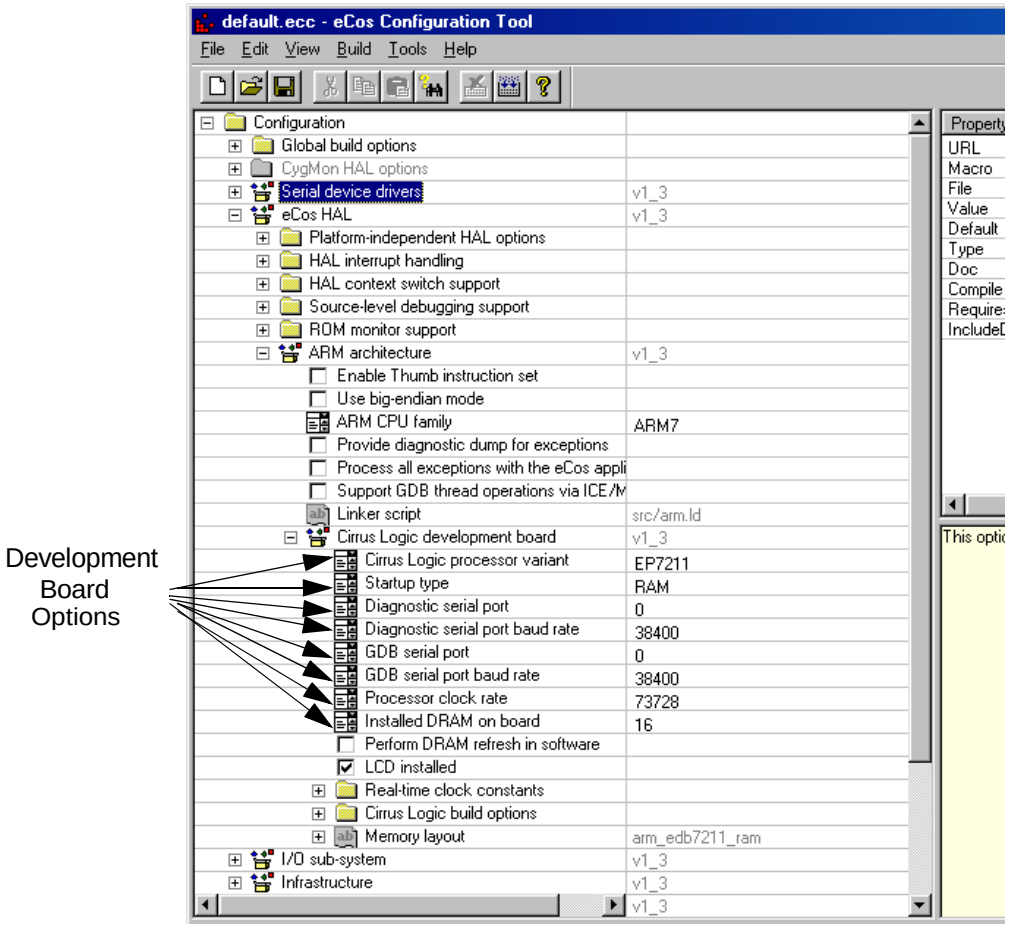

Figure 3. Setting ARM Architecture Options

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).

a) Select: **Tools→Paths→Build Tool**. The dialog box in Figure 4 will appear.

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→Paths→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.

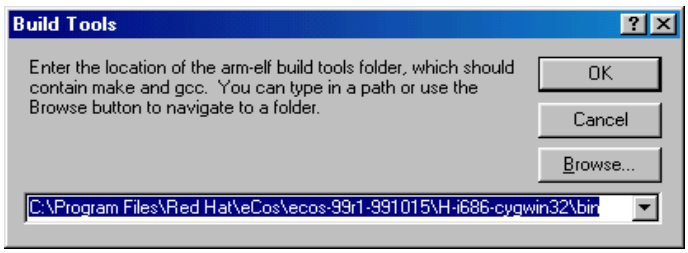

Figure 4. Build Tools Dialog Box

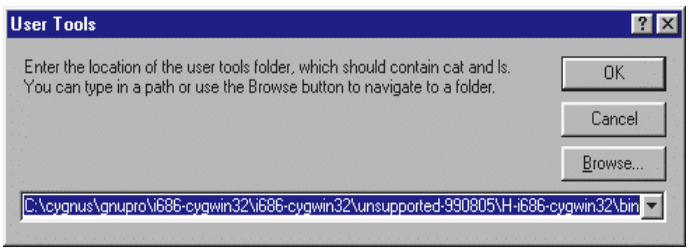

Figure 5. User Tools Dialog Box

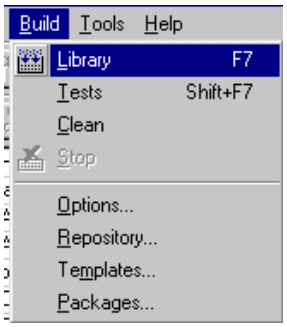

Figure 6. Build Library Menu Selection

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→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.

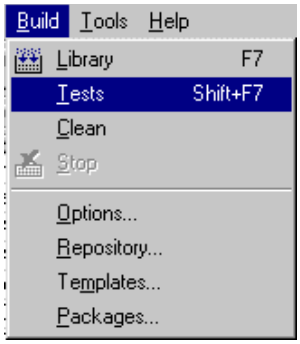

Figure 7. Build Tests Menu Selection

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.

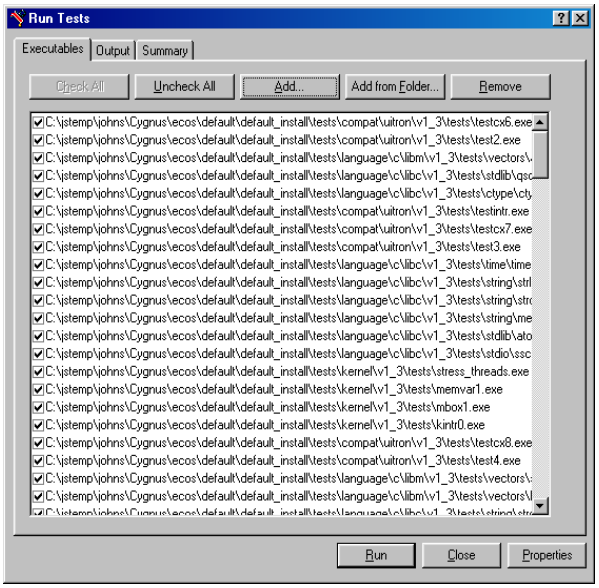

Figure 8. Run Tests Window

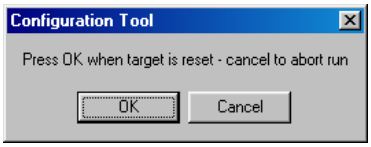

Figure 9. Dialog Box to Acknowledge that the Target Has Been Reset

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 com1
(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 &
```

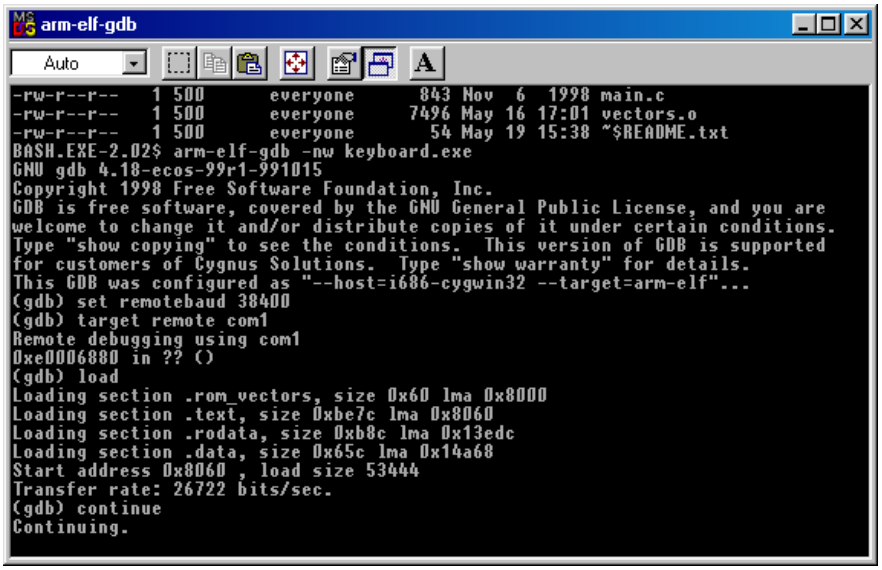

Figure 10. Screen Shot of Executed Code

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

-byte-sex l

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.

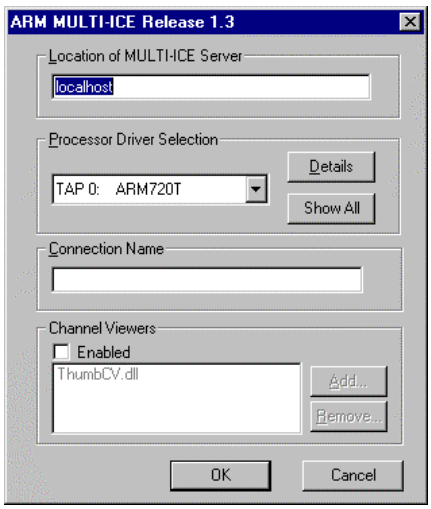

Figure 11. Multi-ICE Dialog Box

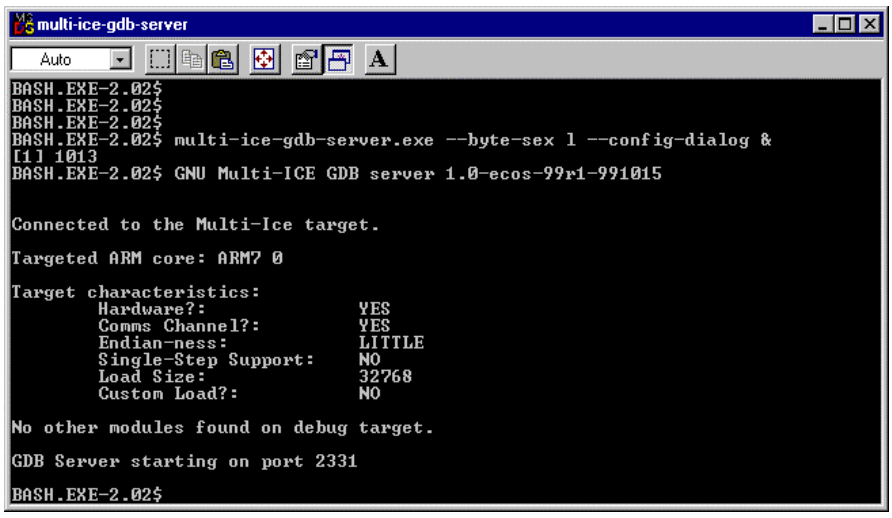

Figure 12. Screen Shot of Multi-Ice-gdb-Server **Starting**

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 •


Maverick™
from
CIRRUS LOGIC