Dharma Board Quick Start Guide

This document describes how to quickly get the Dharma Board up and running.

System Requirements

Before starting, make sure you have the minimum system requirements listed below:

PII-class processor (or better)

Windows 2000 or later

At least 600 MB of free disk space

Available serial (COM) port

The SDK installs the full Cygwin UNIX environment for Windows 2000, version 1.3.2, and uses several packages bundled into the SDK installer. **Interactive Objects strongly recommends that you install the package on the CD-ROM without modifying any of the options.** The SDK has only been tested with the bundled version of Cygwin – newer versions may not work correctly.

Documentation Overview

There are two versions of the documentation: printed and online. All of the SDK documentation is available in both forms.

The online documentation is installed on your system when you run the installer, and is accessible from your Start -> Programs menu.

The printed documentation is in the Documentation directory on the CD-ROM.

The readme.txt file contains this page only, in the root directory. The QuickStart.pdf file contains this page and the Operation Guide which describes building an image and transferring the image to the device.

Hardware Setup

Setting up the hardware is easy. Follow the steps below:

1. Carefully unpack the Dharma board and put it on a flat surface.
2. Connect a null-modem cable between the serial port on the device and an available COM port on your computer.
3. Connect the power supply to the board.

Software Setup

The software setup is similarly simple:

1. Read the ReleaseNotes.txt file on the CD-ROM for specific information about this release.
2. Double-click the **IObjects-sdk-current.exe** on the CD-ROM.
3. When prompted to install Cygwin, click **Yes**.
4. When given the choice of where to install Cygwin from, click **Install from Local Directory**.
5. You can specify different locations for the install, and where to put shortcuts, but otherwise, accept all of the other default settings.

You now have the Cygwin environment set up, and the SDK installed. The next step is to build an image, and download it to the device.

Dharma Operation Guide

This section describes the mechanics of using the Dharma board, including:

- Installing a boot loader
- Building an image
- Downloading an image to the Dharma board

- Debugging an image
- Burning an image to flash
- Debugging an image in flash

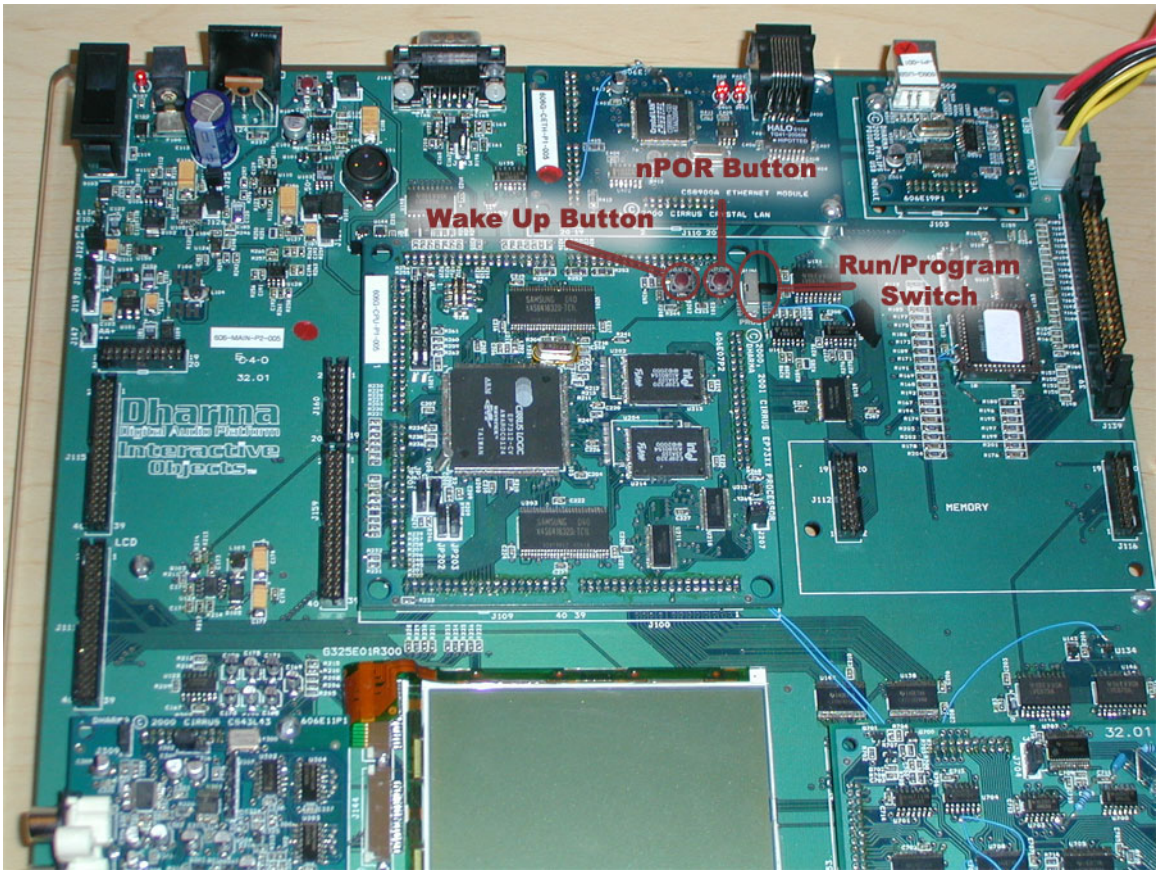
Installation of the software was discussed above, and is also available on the CD in the Readme.txt file.

To perform the steps in this section, you must connect the Dharma board to a COM port on your PC using a null modem cable.

Install the Boot Loader

Running the image requires the use of a gdb stub or a bootstrap environment on the board. Interactive Objects has created a combination gdb stub/bootstrap environment for Dharma using RedBoot™, a product from RedHat. The Dharma board you received should have a Redboot stub on it already. You can verify this by connecting the Dharma board, starting up a term program, and configuring it for your serial port and setting it to 115200 8N1. Next, verify that the run/program switch is in the "run" position, power on the dharma board, wait approximately 1 second, and hit the wakeup button (located on the processor mainboard). You should see some status information regarding the flash, a banner, and a prompt:

```
RedBoot>
```



If you do not see the RedBoot> prompt, you may need to burn a redboot stub to the device. This can be done as follows:

1. Power on the board.
2. Switch the board into "program" mode.
3. Hit the nPOR button on the board to induce a power-on-reset.
4. On the desktop side, open a bash shell, and cd into the sdk directory:
   ```
   cd /c/iobjects/dadio
   ```
5. On the desktop side, start the download application. If your null modem cable is plugged into COM1, the syntax is as follows:
   ```
   support/downloader.exe images/redboot_stub.bin
   ```
6. If you are using a COM port other than 1, you need to specify the port number (X):
   ```
   support/downloader.exe -pX images/redboot_stub.bin
   ```
7. Once this command has been issued, you should see the text:
8. Waiting for the board to wakeup...

9. At this point you can hit the WAKEUP button on the Dharma processor board, and the download sequence should start.
10. When the download has completed, switch the board back into the "run" mode, start your term program as described above, then hit nPOR, wait one second, and hit WAKEUP. you should see a RedBoot> prompt at this point.

RedBoot provides an interactive shell with flash management and image handling capabilities. Discussing the exact feature set is beyond the scope of this document, but is well covered at the following location:

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

For term programs, we highly recommend TeraTerm 2.3 (Win95, 98, NT, 2k compatable), available from the following location:

http://hp.vector.co.jp/authors/VA002416/teraterm.html

Create an Image

1. Start a bash shell and `cd` into your installation directory:

```
cd /c/iobjects/dadio
```

2. Determine which target you want to build by looking at the available targets in configs/targets.mk, then build it. This generates the build tree, and creates a player image:

```
make simple
```

3. After the build is completed, you can cd into the build directory to find the generated image:

```
cd builds/simple
ls -l simple.exe
```

Note: Since the build tree was created in step #2, you can build directly from the build directory and save a little time from this point on:

```
cd /c/iobjects/dadio/builds/simple
make
```

The build system is described more fully in the documentation.

Download an Image

Now that you have created an image, you can transfer this image to the Dharma board and start debugging it, or simply transfer and execute it.

There are two methods for downloading an image to the device:

1. Using `gdb` (the debugger) to download the image into memory.
2. Using RedBoot to download the image to flash memory.

These options are described in the following sections.

Download and Debug the Image

Once a gdb stub has been installed on the board, it is possible to use `gdb` to load and debug an image. Since our image has been cross compiled, we need to use a cross debugger. Assuming you have built the target 'simple', you can perform the following steps to debug the image:

1. Open a bash shell and `cd` into the build directory:

   ```
   cd /c/iobjects/dadio/builds/simple
   ```

2. Prepare the Dharma(tm) board by powering it on, waiting one second, then hitting the WAKEUP button.
3. At the shell, load `gdb`:

   ```
   arm-elf-gdb simple.exe
   ```

   This should bring you to the `(gdb)` prompt.

4. Configure `gdb` for the remote target:

   ```
   (gdb) set height 0
   (gdb) set remotebaud 115200
   (gdb) target remote com1
   ```

The **height** variable controls the pager – setting it to 0 disables the pager. The **remotebaud** variable controls the baud rate of the serial target. **target remote com1** indicates that `gdb` should attach to a remote target on `com1` – if you are using a different serial port, you should specify `com`*X* where *X* is your serial port number.

At this point `gdb` will try to connect to the Dharma board. If the Dharma board is not properly connected or does not have a debug stub on it, `gdb` on the desktop side will hang. You can interrupt `gdb` by pressing CTRL-C, and then type "quit" to exit the console. After the board has been properly connected and set up, repeat steps 3-4 to attach the debugger.

Note: `gdb` will look for and execute a startup script if one is available. This allows highly repetitive commands, such as the three listed above, to be executed automatically. To use a startup script, create a file named `gdb.ini` in the same directory as your image, and put the commands in that file exactly as you would type them at the `gdb` prompt.

---

5. Transfer the image from the host to the device:

```
(gdb) load
```

This may take a while, depending on the size of the image. `gdb` will provide feedback describing which section is currently transfering. When the transfer is complete, it will return to the `(gdb)` prompt.

6. Set any breakpoints you may have:

```
(gdb) break cyg_user_start
(gdb) break main/demos/simple/src/main.cpp:84
```

`gdb` will come back with the line number and file associated with the breakpoint and a breakpoint number. You can disable or delete this breakpoint using the **disable** and **delete** commands.

7. Execute the image:

```
(gdb) continue
```

Most of our demo applications provide some level of feedback over the serial port. Most images will attempt to draw to the lcd, or access peripherals on the board. This should give you feedback about the execution of the application. If you have breakpoints set, `gdb` will return you to the `(gdb)` prompt and indicate the breakpoint that has been hit.

Complete documentation regarding `gdb` syntax and features is available at:

http://www.gnu.org/manual/gdb-4.17/gdb.html

Download an Image to Flash

For demos and repetitive testing, you may want to burn the image to the flash memory rather than transfer it over a serial connection every time. RedBoot provides flash management and can be used to do this as follows:

1. Start a bash shell, and `cd` to the build directory:

   ```
   cd /c/iobjects/dadio/builds/simple
   ```

2. Start your terminal program and wake up the board. Your terminal program must support xmodem, ymodem, or zmodem for the transfer.
3. At the RedBoot prompt, initiate a transfer on the device side:

   ```
   RedBoot> load -m xmodem -b 0x20000 simple
   ```

   This tells redboot to start an xmodem transfer, and to place the transfered image at 0x20000 in RAM. The image is given a name, "simple", but this is not used at this point.

4. Initate a transfer on the host side. If you are using TeraTerm, this can be done by going to File-> Transfer-> XMODEM-> Send.... Send the previously created simple.srec image to the device
5. When the transfer completes, RedBoot will print some gathered information about the image. For example:

   ```
   Entry Point: 0x00020040, address range:
   0x00020000-0x000ae1c4, length: 0x0008e1c4
   ```

You will need this information to burn this image to the flash.

6. Tell RedBoot to burn the image to the flash using the following command:

```
RedBoot> fis create -b 0x20000 -l 0x8e1c4 -e
0x20040 -r 0x20000 simple
```

"fis create" tells redboot we want to create a new image in the Flash Image System (FIS).

"-b 0x20000" indicates the base address in RAM of the image to burn, and corresponds directly to the "-b 0x20000" argument we used in the "load" operation.

"-l 0x8e1c4" tells RedBoot the length of the image – this is taken from the info line in step 6.

"-e 0x20040" tells RedBoot the entry point to the image – again, taken from the info line in step 6.

"-r 0x20000" indicates the address to load the image to. This is important since all of our images will need to be loaded to this address.

"simple" assigns a name to this image.

At this point, RedBoot will begin burning the flash with the image. when it is done, it will return to the RedBoot prompt.

7. Verify the image was burned correctly:

```
RedBoot> fis list
```

The listing of the flash contents should have an entry for "simple".

8. Load and execute the image. It is possible to execute the image already in RAM (from step 4) without reloading it from the FIS. However, to properly verify the image, we should reset the board and reload the image:

```
RedBoot> reset
```

The board should reset and return to the RedBoot> prompt.

```
RedBoot> fis load simple
RedBoot> go
```

This should start executing the image.