Exercise 1 -- Flash an LED

Host Computer Setup

This exercise requires you to first install the Ada toolchain for your computer and check out some **`git`** source code repositories, per the instructions in:

http://git.munts.com/ada-remoteio-tutorial/Setup.pdf

After completing the standard setup procedure, the Ada compiler toolchain should be accessible at **`C:\PROGRA~1\GNAT`** for Windows or **`/usr/local/gnat/`** for Linux and MacOS X.

Last Minute Setup

If you have **not** already set up your computer per **`Setup.pdf`**, you can now perform a quick installation procedure, using an archive file provided on a USB memory stick that will be passed around at this time.

Just unpack the archive file on the USB memory stick appropriate for your computer (**`windows.zip`**, **`linux.tgz`**, **`macos.tgz`**) into your home directory.

Note: For Debian and Ubuntu Linux, you must also install these packages:

```
sudo apt install build-essential libhidapi-dev
```

Note: For MacOS, you must also build and install the HID API library:

```
cd $HOME/hidapi
./bootstrap
./configure
make
sudo make install
```

After completing the last minute setup procedure, the Ada compiler toolchain should be accessible at **`%HOMEDRIVE%%HOMEPATH%\gnat`** for Windows or **`$HOME/gnat`** for Linux and MacOS X.

Hardware Setup

Plug the development board assembly (Raspberry Pi Zero or BeagleBone Green) from the hardware kit into a USB port on your computer. Your computer should automatically configure a USB raw HID device, possibly requiring you to deal with one or more pop-ups.

Exercise Instructions

Following are instructions for Windows, and Linux/MacOS X for this exercise. For each operating system there are two sets of instructions: One using the command line and another using the **`gps`** IDE for GNAT. It is advisable to use **`gps`**, if possible, because it allows you to easily examine all of the component source files.

Microsoft Windows

*Note: The **`set GNAT`** commands below are unnecessary if the Ada toolchain **`bin`** directory is already in the program path.*

Using the DOS Command Line

Open a DOS command window by running **`cmd.exe`** and run the following commands:

```
set GNAT=C:\PROGRA~1\gnat     or     set GNAT=%HOMEDRIVE%%HOMEPATH%\gnat
cd ada-remoteio-tutorial
compile.cmd test_led
test_led
```

The LED on the LPC1114 I/O processor board should begin blinking.

When you are done observing the LED, stop the program with **`CONTROL-C`** and then run the following command to remove all of the working files and return **`ada-remoteio-tutorial/`** to the pristine state:

```
clean.cmd
```

Starting the GPS IDE (Optional)

Open a DOS command window by running **`cmd.exe`** and run the following commands:

```
set GNAT=C:\PROGRA~1\gnat     or     set GNAT=%HOMEDRIVE%%HOMEPATH%\gnat
cd ada-remoteio-tutorial
copydll.cmd
%GNAT%\bin\gps -P tutorial.gpr
```

Now skip to page 4 to build and run the LED test program.

Linux and MacOS X

*Note: The **`export`** commands below are unnecessary if the **`bin`** directory for the Ada toolchain you will be using is already in the program path (e.g. you are using the Debian native Ada toolchain).*

Using the Command Line Shell

Open a terminal window to get a command shell and run the following commands:

```
export GNAT=/usr/local/gnat     or     export GNAT=$HOME/gnat
cd ada-remoteio-tutorial
make test_led
./test_led
```

The LED on the LPC1114 I/O processor board should begin blinking.

When you are done observing the LED, stop the program with **`CONTROL-C`** and then run the following command to remove all of the working files and return **`ada-remoteio-tutorial/`** to the pristine state:

```
make clean
```

Starting the GPS IDE (Optional)

Open a terminal window to get a command shell and run the following commands:

```
export GNAT=/usr/local/gnat     or     export GNAT=$HOME/gnat
cd ada-remoteio-tutorial
$GNAT/bin/gps -P tutorial.gpr
```

Now continue on page 4 to build and run the LED test program.

Building and Running from the GPS IDE

Build test_led

1. Expand the **.** directory in the **Project** tab and then double click on **test_led.adb**.

2. From the **gps** menu bar, click **Build** -> **Project** -> **Build <current file>**.

3. A dialog box with the title **Build <current file>** will appear. Click on the **Execute** button to start the build process. Output messages from the build process will appear in the **Messages** tab.

Run test_led

1. From the **gps** menu bar, click **Debug** -> **Initialize** -> **no main file** followed by **Debug** -> **Debug** -> **Load File...** and then double click **test_led**. The first line of **test_led.adb** should now be highlighted in green.

2. From the **gps** menu bar, click **Debug** -> **Run...**

3. A dialog box with the title **Run/Start** will appear. Uncheck **Stop at beginning of main subprogram** and then click on the **OK** button to start running test_led. Output messages from **test_led** will appear in the **Debugger Execution** tab.

4. When you are done observing the LED, stop the program from the **gps** menu bar by clicking **Debug** -> **Terminate**.

5. To remove all build artifacts and return **ada-remoteio-tutorial/** to the pristine state, from the **gps** menu bar click **Build** -> **Clean** -> **Clean All** and then click on the **Execute** button.

Going Further

Try temporarily replacing the Raspberry Pi or BeagleBone Green assembly with one of the other Remote I/O server devices that are being passed around. With the notable exception of the red **FEZ**, most of the other servers also have an LED on Remote I/O GPIO channel 0. You should be able to run **test_led** unmodified with them.

If you want to try the same thing on the **FEZ**, you will need to change the value for the constant **Channel_LED** to **20** or **21** and then rebuild **test_led**. (The **FEZ** assigns Remote I/O GPIO channels 0 to 13 to the **D0** to **D13** signals of the Arduino expansion headers. See the package **RemoteIO.FEZ** for more information.)