GPIO Control using the BeagleBone Green PRU with Remoteproc

By Eric Yang, Karen Li, Joanne Yoon, Hye Lim Moon

V 1.0 Last updated: December 2016

This document guides the user through:

1. Loading firmware and starting the PRU
2. Controlling GPIO pins via the PRU
3. Writing a DTO for configuring pin modes

Table of Contents

1 Programmable Real-Time Units 2
2 Building a Sample Project 2
2.1 Setting Up the Development Environment 2
2.2 Troubleshooting: 3
3 Starting the PRU 4
3.1 Copy Compiled Firmware to /lib/firmware/ 4
3.2 Load firmware 4
3.3 Troubleshooting 5
4 GPIO Toggle with the PRU 6
4.1 Troubleshooting 7
5 Device Tree 7
5.1 How to find values for the .dts file 7
5.2 Build and Load the DTO 9
6 Helpful Links / References: 9

1 Programmable Real-Time Units

The BeagleBone Green contains 2 Programmable Real-time Units (PRU) that are accessible for you to use. Each of these 32-bit 200MHz processors provide single-cycle I/O access to several pins and full access to the internal memory on the AM335x processor, allowing you to run code independently from your main BeagleBone Green CPU. They are designed for perform tasks that have extremely strict timing requirements.

In older kernels, Linux provides an uio_pruss driver for loading firmware written onto the PRU. Newer linux kernels (v4.1.xx) no longer have this, and instead use the Remote Processor (Remoteproc) framework to interact with peripheral devices. TI's pru_rproc driver uses this new framework to allow programs to be loaded onto the PRU. For communication between the PRU and Linux kernel, TI provides the rpmsg module. The following guide will walk through running a C program controlling GPIO pins on the PRU.

2 Building a Sample Project

TI has provided example projects to show you how to use the PRU which are located on your BeagleBone at:

```
/opt/source/pru-software-support-package/examples/am335x
```

Alternatively, the sample code can also be found here: https://github.com/dinuxbg/pru-software-support-package/tree/master/examples/am335x

2.1 Setting Up the Development Environment

Before we can build a PRU project we need to setup our development environment on our BeagleBone Green.The Makefiles for the examples require an environment variable called `PRU_CGT` to be set and pointing at TI's PRU code generation tools. This environment variable **must** be set or your build will fail.

1. Check to see if you have the PRU compiler installed on your Beaglebone:

   ```
   # whereis clpru
   ```

   This should show you where the PRU compiler is. The output should look like the following:

   ```
   clpru: /usr/bin/clpru /usr/share/man/man1/clpru.1
   ```

2. Create a symbolic link to the compiler

   ```
   # cd /usr/share/ti/cgt-pru
   # mkdir bin
   # cd bin
   # ln -s /usr/bin/clpru clpru
   ```

3. We now have successfully linked our PRU compiler. To test it out you can use the following command:

   ```
   # /usr/share/ti/cgt-pru/bin/clpru
   ```

   Alternatively you could link your entire `/usr/bin` directory to `/usr/share/ti/cgt-pru/bin` which can be done by the following command:

   ```
   # ln -s /usr/bin/ /usr/share/ti/cgt-pru/bin
   ```

4. Next, we will create the environment variable needed using export
   `# export PRU_CGT=/usr/share/ti/cgt-pru`

5. You only need to create the link once but the environment variable must be set each time you want to build the project. For convenience, you should set the environment variable in `~/.profile`.

6. Select an example project to build and run `make`. After compilation, the output files will be located inside the `gen/` folder.

2.2 Troubleshooting:

1. **I could not find `/opt/source/pru-software-support-package/examples/am335x`**

   If you are unable to find this on your system then you can simply download it directly from Ti's git repo (https://git.ti.com/pru-software-support-package/).

2. **`whereis clpru` did not find anything**

   If you are unable to find the compiler already installed on your Beaglebone then you can download it from Ti directly (http://software-dl.ti.com/codegen/non-esd/downloads/download.htm#PRU). Once you downloaded it, install it on your machine and link the compiler to where you installed it.

3. **I get the following error when building the project**

```
**********************************************************************
PRU_CGT environment variable is not set. Examples given:
(Desktop Linux) export PRU_CGT=/path/to/pru/code/gen/tools/ti-cgt-pru_2.1.2
(Windows) set PRU_CGT=C:/path/to/pru/code/gen/tools/ti-cgt-pru_2.1.2
(ARM Linux*) export PRU_CGT=/usr/share/ti/cgt-pru
*ARM Linux also needs to create a symbolic link to the /usr/bin/ directory
in order to use the same Makefile
(ARM Linux) ln -s /usr/bin/ /usr/share/ti/cgt-pru/bin
**********************************************************************
.  Stop.
```

   As it says, you have not set your PRU_CGT environment variable. Set the PRU_CGT environment variable using the following command and try to build again.
   `# export PRU_CGT=/usr/share/ti/cgt-pru`

4. **I get an error when making the project that looks like the following:**

```
Building file: main.c
Invoking: PRU Compiler
/usr/share/ti/cgt-pru/bin/clpru --include_path=/usr/share/ti/cgt-pru/include
--include_path=../../../include --include_path=../../../include/am335x -v3 -
O2 --display_error_number --endian=little --hardware_mac=on --
obj_directory=gen --pp_directory=gen -ppd -ppa -fe gen/main.object main.c
make: /usr/share/ti/cgt-pru/bin/clpru: Command not found
Makefile:88: recipe for target 'gen/main.object' failed
make: *** [gen/main.object] Error 127
```

The error is caused by the makefile not being able to find the PRU compiler. Verify that you linked the PRU compiler to be in `/usr/share/ti/cgt-pru/bin/clpru` and that the `PRU_CGT` environment variable is set correctly to point to `/usr/share/ti/cgt-pru`.

3 Starting the PRU

To start the PRU, firmware compiled in the previous step needs to be loaded onto the PRU using the pru_rproc module. This module looks for the firmware for the corresponding PRU core inside the `/lib/firmware/` folder. To load firmware into PRU0, `/lib/firmware/` must contain a file named `am335x-pru0-fw`, and for PRU1, a file named `am335x-pru1-fw`. These steps will use TI's `PRU_RPMsg_Echo_Interrupt1` example to demonstrate how to work with both the pru_rproc and rpmsg modules. Follow the steps in the previous section to build the firmware first.

3.1 Copy Compiled Firmware to /lib/firmware/

1. Inside the `gen/` folder compiled from the last section, locate the `.out` file. This is the firmware built by the compiler to run on the PRU. Copy this file to the `/lib/firmware/` folder as the firmware file for the desired PRU:
   ```
   # cp PRU_RPMsg_Echo_Interrupt1.out /lib/firmware/am335x-pru1-fw
   ```

3.2 Load firmware

Once the binary file has been copied correctly to `/lib/firmware/`, restart the PRU drivers to load the firmware.

1. Stop the rproc driver:
   ```
   # rmmod -f pru_rproc
   ```
2. If using the rpmsg module (such as in the `PRU_RPMsg_Echo_Interrupt1` example), all of these modules must be stopped:
   ```
   # rmmod -f rpmsg_pru
   # rmmod -f virtio_rpmsg_bus
   # rmmod -f pru_rproc
   ```
3. Restart the rproc driver:
   ```
   # modprobe pru_rproc
   ```
4. Check `dmesg | tail -n 30` to see if the firmware was loaded correctly. The output should be similar to this:

```
root@cla233-beagle:~# dmesg | tail -n 30
[18432.652499] Disabling lock debugging due to kernel taint
[18440.103452] pruss-rproc 4a300000.pruss: unconfigured system_events =
0x0800000000000000 host_intr = 0x00000002
[18440.103479]  remoteproc1: stopped remote processor 4a338000.pru1
[18445.445481] pru-rproc 4a338000.pru1: pru_rproc_remove: removing rproc
4a338000.pru1
[18445.449590]  remoteproc1: releasing 4a338000.pru1
[18453.402376]  remoteproc1: 4a334000.pru0 is available
[18453.402401]  remoteproc1: Note: remoteproc is still under development and
```

```
considered experimental.
[18453.402410]  remoteproc1: THE BINARY FORMAT IS NOT YET FINALIZED, and
backward compatibility isn't yet guaranteed.
[18453.402599]  remoteproc1: Direct firmware load for am335x-pru0-fw failed
with error -2
[18453.402616]  remoteproc1: failed to load am335x-pru0-fw
[18453.407926] pru-rproc 4a334000.pru0: booting the PRU core manually
[18453.407938]  remoteproc1: powering up 4a334000.pru0
[18453.407976]  remoteproc1: Direct firmware load for am335x-pru0-fw failed
with error -2
[18453.407987]  remoteproc1: request_firmware failed: -2
[18453.413189] pru-rproc 4a334000.pru0: rproc_boot failed
[18453.421701]  remoteproc1: releasing 4a334000.pru0
[18453.421848] pru-rproc: probe of 4a334000.pru0 failed with error -2
[18453.422202]  remoteproc1: 4a338000.pru1 is available
[18453.422215]  remoteproc1: Note: remoteproc is still under development and
considered experimental.
[18453.422224]  remoteproc1: THE BINARY FORMAT IS NOT YET FINALIZED, and
backward compatibility isn't yet guaranteed.
[18453.423468]  remoteproc1: registered virtio0 (type 7)
[18453.423631] pru-rproc 4a338000.pru1: PRU rproc node
/ocp/pruss@4a300000/pru1@4a338000 probed successfully
[18453.463601]  remoteproc1: powering up 4a338000.pru1
[18453.464479]  remoteproc1: Booting fw image am335x-pru1-fw, size 186964
[18453.464680] pruss-rproc 4a300000.pruss: configured system_events =
0x0800000000000000 intr_channels = 0x00000002 host_intr = 0x00000002
[18453.464693]  remoteproc1: remote processor 4a338000.pru1 is now up
[18453.465274] virtio_rpmsg_bus virtio0: rpmsg host is online
[18453.465329] virtio_rpmsg_bus virtio0: creating channel rpmsg-pru addr
0x1f
[18453.484321] rpmsg_pru rpmsg0: new rpmsg_pru device: /dev/rpmsg_pru31
```

Note: pru_rproc attempts to load both PRU0 and PRU1 at the same time. If one or both of `am335x-pru0-fw` and `am335x-pru1-fw` does not exist, pru_rproc will fail when loading to the corresponding PRU.

5. You now have firmware running on the PRU! To send a message to the PRU, run:
   ```
   # echo "test" > /dev/rpmsg_pru31
   ```

3.3 Troubleshooting

1. **I get the following error when removing the `virtio_rpmsg_bus` module**

```
root@cla233-beagle:~# rmmod virtio_rpmsg_bus
rmmod: ERROR: Module virtio_rpmsg_bus is in use by: rpmsg_pru
root@cla233-beagle:~# rmmod -f virtio_rpmsg_bus
rmmod: ERROR: ../libkmod/libkmod-module.c:777 kmod_module_remove_module()
could not remove 'virtio_rpmsg_bus': Resource temporarily unavailable
rmmod: ERROR: could not remove module virtio_rpmsg_bus: Resource temporarily
unavailable
```

This module is used by the `rpmsg_pru` module. To remove it, make sure to remove `rpmsg_pru` first.

2. I get loading failure in dmesg after starting pru_rproc

```
[18453.402599]  remoteproc1: Direct firmware load for am335x-pru0-fw failed
with error -2
[18453.402616]  remoteproc1: failed to load am335x-pru0-fw
[18453.407926] pru-rproc 4a334000.pru0: booting the PRU core manually
[18453.407938]  remoteproc1: powering up 4a334000.pru0
[18453.407976]  remoteproc1: Direct firmware load for am335x-pru0-fw failed
with error -2
[18453.407987]  remoteproc1: request_firmware failed: -2
[18453.413189] pru-rproc 4a334000.pru0: rproc_boot failed
[18453.421701]  remoteproc1: releasing 4a334000.pru0
[18453.421848] pru-rproc: probe of 4a334000.pru0 failed with error -2
```

This error is most likely due to pru_rproc not being able to locate the firmware files. Check that you have the correctly named files in `/lib/firmware`:

```
# ls /lib/firmware/am335x-pru*
```

```
/lib/firmware/am335x-pru0-fw
/lib/firmware/am335x-pru1-fw
```

4 GPIO Toggle with the PRU

Refer to TI's `PRU_gpioToggle` example for a simple program on toggling a GPIO pin. Follow these steps to change TI's toggle GPIO example to toggle a specific pin on the BeagleBone.

1. Notice that the `PRU_gpioToggle.c` contains two register variables, `__R30` and `__R31`. `__R30` is for PRU output, and `__R31` is for input. We will use `__R30` for driving the GPIO pin high or low.

2. Chose a pin on the BeagleBone that can be accessed by the PRU
   - Refer to the P8 and P9 Headers documents (These documents are for BeagleBone Black, but the pins on the BeagleBone Green is the same). Chose a pin that has `pru_r30` in one of its pinmux. This guide will use P8_27.
   - Mode 5 of P8_27 says `pr1_pru1_pru_r30_8`. This means that P8_27 can be used as GPIO output (`r30`) by PRU1 (`pru1`). Note that PRU0 will not be able to toggle P8_27.
   - The 8 at the end represents the bit (offset) in the register `__R30` that corresponds to this pin.

3. Inside PRU_gpioToggle.c, change this line of code:
   ```
   gpio = 0x000F;
   ```
   to:
   ```
   gpio = 0x0100;
   ```
   This sets the 8th bit of the register (as found in the previous step) which controls P8_27. Setting this bit to 1 will drive the pin high and 0 will drive the pin low

4. Save the file and run `make`

5. Follow the steps in the previous section to load this firmware

6. Configure P8_27 to be in the pruout mode

- Enable the universal cape in order to have access to the pinmux (**Make a copy of `/boot/uEnv.txt` first!**)
  ```
  # nano /boot/uEnv.txt
  ```
- Change this line:
  ```
  cmdline=coherent_pool=1M quiet cape_universal=disabled
  ```
  to:
  ```
  cmdline=coherent_pool=1M quiet cape_universal=enabled
  ```
- Reboot
- Change the pin mode
  ```
  # config-pin P8_27 pruout
  ```

7. Use a voltmeter/oscilloscope to see the GPIO pin (P8_27) being driven high (1) and low (0) about every 0.05 seconds.
   - `__delay_cycles(100000000)` on the BBG's PRU processor is about 0.05 seconds

8. To allow the PRU to receive commands from Linux userspace, try incorporating the `PRU_gpioToggle` example into the `PRU_RPMsg_Echo_Interrupt1` example.

4.1 Troubleshooting

1. **I cannot set the pin mode**

```
root@beagle:~# config-pin P8_27 pruout
P8_27 pinmux file not found!
cape-universala overlay not found
run "config-pin overlay cape-universala" to load the cape
```

   Make sure that the universal cape is enabled in `/boot/uEnv.txt`

2. **I have a corrupt /boot/uEnv.txt file!**
   Follow the steps in section 5 of the Audio Guide to recover

5 Device Tree

We can create a device tree overlay to configure the GPIO pin for mode 5 (pruout) with the universal cape disabled. This allows us to easily load or unload the cape using C, and for other custom capes which may conflict with the universal cape to be loaded.

5.1 How to find values for the .dts file

1. From P8 Headers, we see that the address (ADDR column) of P8_27 is `0x0E0`

2. With universal cape **enabled**, run the following to find out the hex value for the pruout mode (mode 5 from P8 Headers)
   ```
   # config-pin P8_27 pruout
   # cat /sys/kernel/debug/pinctrl/44e10800.pinmux/pins | grep 56
   ```
   This should give this:
   ```
   pin 56 (44e108e0.0) 00000005 pinctrl-single
   ```

Note: 56 is in the $PINS column for P8_27 from P8 Headers, meaning P8_27 is pin 56 in the `/sys/kernel/debug/pinctrl/44e10800.pinmux/pins` file

3. From the result of the cat, we see that the hex value of the pin is `0x05` (`00000005`) when in pruout mode

4. Thus, with the `0x0E0` from step 2, and `0x05` from step 3, we write the following in the pin configuration section of the `.dts` file:

```
pinctrl-single,pins = <
   0x0E0 0x05 // P8_27: Mode 5
>;
```

Here is a full example (`test.dts`) configuring P8_27 for pruout mode:

```
/* Adapted from http://stackoverflow.com/questions/16872763/configuring-
pins-mode-beaglebone/17064969 */

/dts-v1/;
/plugin/;

/ {
	compatible = "ti,beaglebone", "ti,beaglebone-black";
	part-number = "test";
	version = "00A0";

	fragment@0 {
		target = <&am33xx_pinmux>;
		__overlay__ {
			test_pins: pinmux_test_pins {
				pinctrl-single,pins = <
					0x0E0 0x05 // P8_27: Mode 5
				>;
			};
		};
	};
	fragment@1 {
		target = <&ocp>;
		__overlay__ {
			test_pinmux {
				compatible = "bone-pinmux-helper";
				status = "okay";
				pinctrl-names = "default";
				pinctrl-0 = <&test_pins>;
			};
		};
	};
};
```

5.2 Build and Load the DTO

1. **Disable** the universal cape and reboot

2. Copy `test.dts` to the `~/` directory on the Beaglebone

3. Build the `.dtbo` file
```
# cd ~/
# dtc -O dtb -o TEST-00A0.dtbo -b 0 -@ test.dts
```

4. Copy the `TEST-00A0.dtbo` file to `/lib/firmware`
```
# cp TEST-00A0.dtbo /lib/firmware
```

5. Load the device tree overlay
```
# echo TEST > $SLOTS
```

6. To load automatically on startup, add to the cape manager:
```
# nano /etc/default/capemgr
```
Add `CAPE=TEST` to the end of the file

6 Helpful Links / References:

- Beaglebone: remoteproc “Hello, world!” http://theduchy.ualr.edu/?p=996
- Ti Documenation on PRUs http://processors.wiki.ti.com/index.php/PRU-ICSS
- Remoteproc and RPMsg Documentation http://processors.wiki.ti.com/index.php/PRU-ICSS_Remoteproc_and_RPMsg