PRU Cookbook

Mark A. Yoder

Table of Contents

1. Case Studies

The **P**rogrammable **R**eal-Time **U**nit (PRU) has two 32-bit cores which run independently of the ARM processor that is running Linux. Therefore they can be programmed to respond quickly to inputs and produce very precisely timed outputs. A good way to learn how to use the PRUs is to study how others have used them. Here we present some case studies that do just that.

In these study you'll see a high-level view of using the PRUs. In later chapters you will see the details.

Here we present

- Robotics Control Library http://strawsondesign.com/docs/roboticscape/
- BeagleLogic https://github.com/abhishek-kakkar/BeagleLogic/wiki
- LEDscape https://github.com/Yona-Appletree/LEDscape
- MachineKit http://www.machinekit.io/
- ArduPilot http://ardupilot.org/

1.1. Robotics Control Library

The Robotics Control Library is a package, that is already installed, that contains a C library and example/testing programs for the BeagleBone Blue and the BeagleBone Black with Robotics Cape. It uses the PRU to extend the real-time hardware of the Bone.

1.1.1. Problem

How do I configure the pins so the PRUs are accessable

1.1.2. Solution

It depends on which Beagle you are running on. If you are on the Blue, everything is already configured for you. If you are on the Black or Pocket you'll need to run the following script.

servos_setup.sh

```
#!/bin/bash
# Configure the PRU pins based on which Beagle is running
machine=$(awk '{print $NF}' /proc/device-tree/model)
echo -n $machine
if [ $machine = "Black" ]; then
    echo " Found"
    pins="P8_27 P8_28 P8_29 P8_30 P8_39 P8_40 P8_41 P8_42"
elif [ $machine = "Blue" ]; then
    echo " Found"
    pins=""
elif [ $machine = "PocketBeagle" ]; then
    echo " Found"
    pins="P2_35 P1_35 P1_02 P1_04"
else
    echo " Not Found"
    pins=""
fi

for pin in $pins
do
    echo $pin
    config-pin $pin pruout
    config-pin -q $pin
done
```

1.1.3. Discussion

The first part of the code looks in `/proc/device-tree/model` to see which Beagle is running. Based on that it assigns `pins` a list of pins to configure. Then the last part of the script loops through each of the pins and configures it.

1.1.4. Problem

I need to control eight servos, but the Bone doesn't have enough PWMs.

1.1.5. Solution

The Robotics Control Library provides eight additional PWM channels via the PRU that can be used out of the box. Just run:

```
bone$ sudo rc_test_servos -f 10 -p 1.5
```

The `-f 10` says to use a frequency of 10 Hz and the `-p 1.5` says to set the position to `1.5`. The range of positions is `-1.5` to `1.5`. Run `rc_test_servos -h` to see all the options.

```
bone$ rc_test_servos -h

 Options
 -c {channel}   Specify one channel from 1-8.
                Otherwise all channels will be driven equally
 -f {hz}        Specify pulse frequency, otherwise 50hz is used
 -p {position}  Drive servo to a position between -1.5 & 1.5
 -w {width_us}  Send pulse width in microseconds (us)
 -s {limit}     Sweep servo back/forth between +- limit
                Limit can be between 0 & 1.5
 -r {ch}        Use DSM radio channel {ch} to control servo
 -h             Print this help messege

sample use to center servo channel 1:
   rc_test_servo -c 1 -p 0.0
```

The BeagleBone Blue sends these eight outputs to it’s servo channels. The Black and the Pocket use the pins shown in this table.

Table 1. PRU register to pin table

Pru pin	Blue pin	Black pin	Pocket pin
pru1_r30_8	1	P8_27	P2.35
pru1_r30_10	2	P8_28	P1.35
pru1_r30_9	3	P8_29	P1.02
pru1_r30_11	4	P8_30	P1.04
pru1_r30_6	5	P8_39	
pru1_r30_7	6	P8_40	
pru1_r30_4	7	P8_41	
pru1_r30_5	8	P8_42	

1.1.6. Discussion

This comes from:

- https://github.com/beagleboard/pocketbeagle/wiki/System-Reference-Manual#673_PRUICSS_Pin_Access
- [/opt/source/Robotics_Cape_Installer/pru_firmware/src/pru1-servo.asm]
- https://github.com/derekmolloy/exploringBB/blob/master/chp06/docs/BeagleboneBlackP8HeaderTable.pdf
- https://github.com/derekmolloy/exploringBB/blob/master/chp06/docs/BeagleboneBlackP9HeaderTable.pdf

1.1.7. Problem

rc_test_servos is nice, but I need to control the servos individually.

1.1.8. Solution

You can modify rc_test_servos.c. You'll find it on the bone at /opt/source/Robotics_Cape_Installer/examples/src/rc_test_servos.c, or online at https://github.com/StrawsonDesign/Robotics_Cape_Installer/blob/master/examples/src/rc_test_servos.c.

Just past line 250 you'll find a while loop that has calls to rc_servo_send_pulse_normalized(ch,servo_pos) and rc_servo_send_pulse_us(ch, width_us). The first call sets the pulse width relative to the pulse period; the other sets the width to an absolute time. Use whichever works for you.

1.1.9. Problem

I need more than eight PWM channels, or I need less jitter on the off time.

1.1.10. Solution

This is a more advanced problem and required reprograming the PRUs. See Building Blocks - Applications for an example.

1.1.11. Problem

I want to use four encoders to measure four motors, but I only see hardware for three.

1.1.12. Solution

The forth encoder can be implemented on the PRU. If you run rc_test_encoders_eqep on the Blue, you will see the output of encoders E1-E3 which are connected to the eEQP hardware.

```
bone$ rc_test_encoders_eqep

Raw encoder positions
      E1   |      E2   |      E3   |
         0 |         0 |         0 |^C
```

You can also access these hardware encoders on the Black and Pocket using the pins shown below.

Table 2. eQEP to pin mapping

eQEP	Blue pin	Black pin A	Black pin B	Pocket pin A	Pocket pin B
0	E1	P9_42B	P9_27	P1.31	P2.24
1	E2	P8_35	P8_33	P2.10	
2	E3	P8_12	P8_11	P2.24	P2.33

eQEP	Blue pin	Black pin A	Black pin B	Pocket pin A	Pocket pin B
2		P8_41	P8_42		
	E4	P8_16	P8_15	P2.09	P2.18

You will need to first configure the pins using .encoder.sh

```
#!/bin/bash
# Configure the pins based on which Beagle is running
machine=$(awk '{print $NF}' /proc/device-tree/model)
echo -n $machine

# Configure eQEP pins
if [ $machine = "Black" ]; then
    echo " Found"
    pins="P9_92 P9_27 P8_35 P8_33 P8_12 P8_11 P8_41 P8_42"
elif [ $machine = "Blue" ]; then
    echo " Found"
    pins=""
elif [ $machine = "PocketBeagle" ]; then
    echo " Found"
    pins="P1_31 P2_34 P2_10 P2_24 P2_33"
else
    echo " Not Found"
    pins=""
fi

for pin in $pins
do
    echo $pin
    config-pin $pin qep
    config-pin -q $pin
done

###########################################
# Configure PRU pins
if [ $machine = "Black" ]; then
    echo " Found"
    pins="P8_16 P8_15"
elif [ $machine = "Blue" ]; then
    echo " Found"
    pins=""
elif [ $machine = "PocketBeagle" ]; then
    echo " Found"
    pins="P2_09 P2_18"
else
    echo " Not Found"
    pins=""
fi

for pin in $pins
do
    echo $pin
    config-pin $pin pruin
    config-pin -q $pin
done
```

The eQEP pins are configured with the top half of the code.

1.1.13. Problem

I want to access the PRU encoder.

1.1.14. Solution

The forth encoder is implemented on the PRU and accessed with `sudo rc_test_encoders_pru`

NOTE This command needs root permissions, so the `sudo` is needed.

Here's what you will see

```
bone$ sudo rc_test_encoders_pru
[sudo] password for debian:

Raw encoder position
      E4   |
         0 |^C
```

If you aren't running the Blue you will have to configure the pins as shown above. The bottom half of the code does the PRU configuring.

1.2. BeagleLogic

1.2.1. Problem

I need a 100Msps, 14-channel logic analyzer

1.2.2. Solution

BeagleLogic is a 100Msps, 14-channel logic analyzer that runs on the Beagle. The quickest solution is to get the no-setup-required image. It runs on an older image (15-Apr-2016) but should still work.

If you want to be running a newer image, there are instructions on the site for building BeagleLogic from scratch.

1.2.3. Discussion

BeagleLogic uses the two PRUs to sample at 100Msps. Getting a PRU running at 200Hz to sample at 100Msps is a slick trick. The Embedded Kitchen has a nice article explaining how the PRUs get this type of performance. In section Building Blocks we'll give an overview of the technique.

1.3. MachineKit

MachineKit is a platform for machine control applications. It can control machine tools, robots, or other automated devices. It can control servo motors, stepper motors, relays, and other devices related to machine tools.

1.4. LEDScape

1.5. ArduPilot