e-ale-elce2019

Embedded Essentials ELCE 2019

Version 2.2

e-ale

The E-ALE (Embedded Apprentice Linux Engineer) is a series of seminars held at existing conferences covering topics which are fundamental to a Linux professional in the field of Embedded Linux.

This seminar will spend equal time on lecture and hands on labs at the end of each seminar which allow you to practice the material you've learned.

> This material makes the assumption that you have minimal experience with using Linux in general, and a basic understanding of general industry terms. The assumption is also made that you have access to your own computers upon which to practice this material.

More information can be found at **https://e-ale.org/**

Contents

Chapter 1

GPIOs and libgpiod

General Purpose Input/Outputs

e-ale

1.1 GPIO

Introduction to GPIOs and libgpio

- Speaker: Behan Webster <behanw@converseincode.com>
- ELC (2019.08.22)

Brought to you by

- Linux Foundation Training has provided room funding

GPIO

- GPIO stands for **General Purpose Input/Output**
- GPIO pins can be programmed to be used as inputs or outputs
- As an input you can read back values of 1 or 0
- As an output you can write values of 1 or 0 to the GPIO pin

Active high or low

- Further they can be configured **Active high** or **Active low**
- These settings designate whether a high or low voltage is considered a 1 or a 0

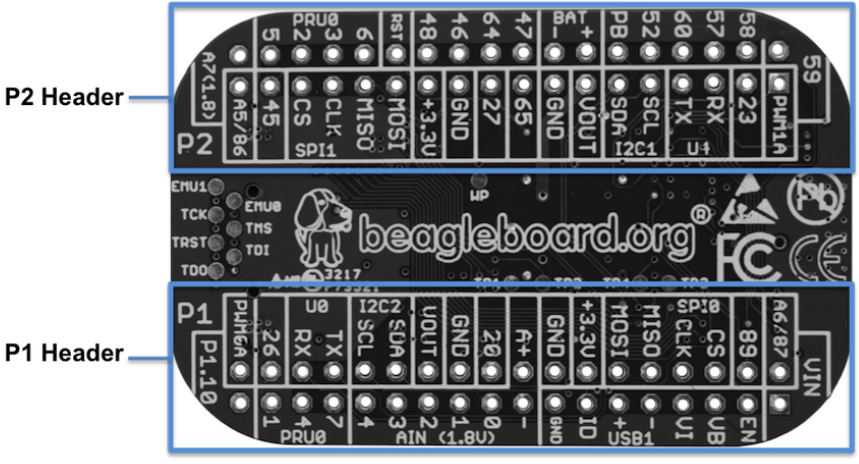
Open Drain or Open Source

- Further more some SoCs have internal pull up and/or pull down resistors which can be used to force the value of the pin up or down when the pin isn't being driven
- **Open Drain** refers to the situation where a signal usually floats high unless driven low by the value of the GPIO pin
- **Open Source** refers to the situation where a signal usually floats low unless driven high by the value of the GPIO pin

Bit Banging Protocols

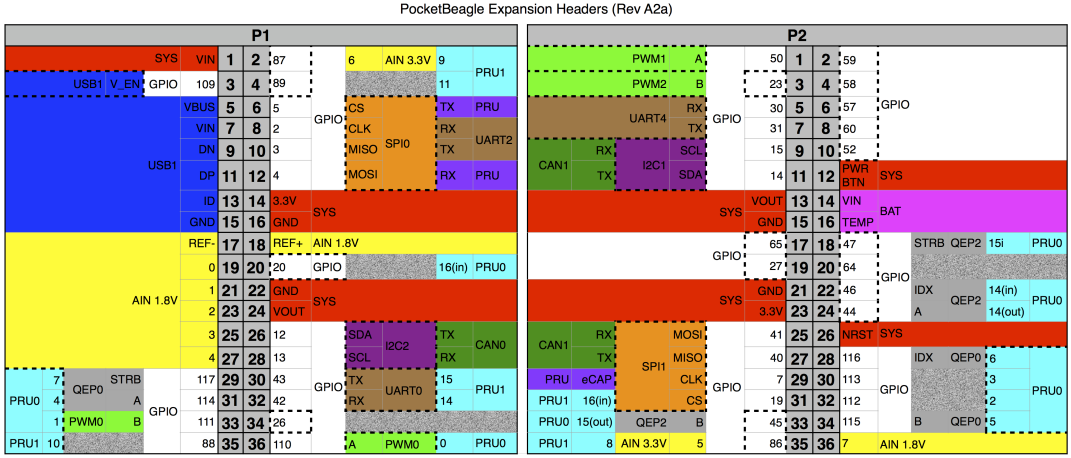
- The best situation is one where dedicated HW can be used to implement HW protocols in the most efficient manner possible
- However in situations where you don't have a HW bus controller, or you don't have enough buses, one can elect to synthesize a protocol using SW to drive GPIOS appropriately to emulate the bus protocol
- This SW implementation of a HW protocol is affectionately known as **bit banging** in the industry

PocketBeagle Pins



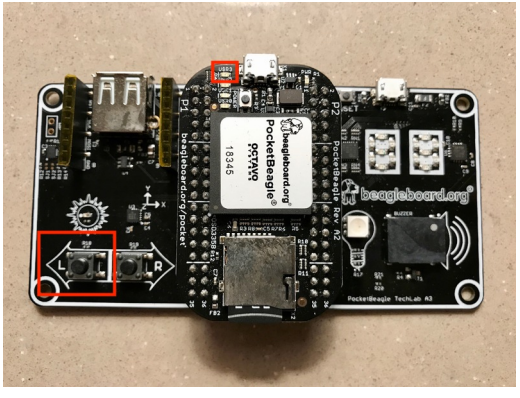
- Pins are shared amongst multiple peripherals
- A pin multiplexer is used to choose the configuration of the pins in use.

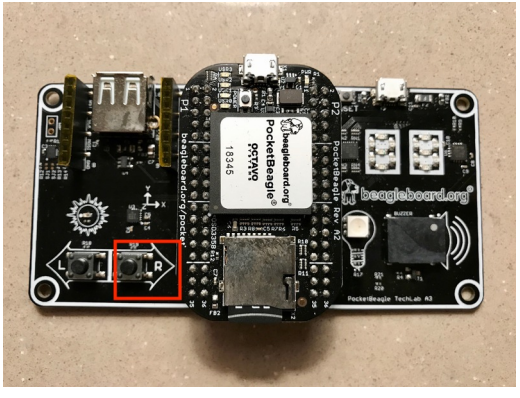
PocketBeagle GPIO pins



- We have 44 digital GPIOs accessible with 18 enabled by default.
- 4 of the GPIOs can alternately be used as PWMs with 2 of these enabled b default.

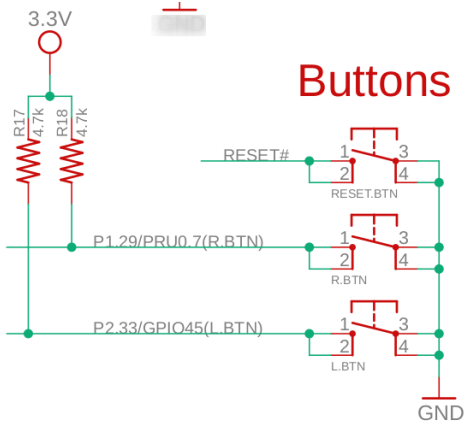
PocketBeagle GPIO pins





- We will use the GPIOS tied to the Left button on the techlab and USR3 LED on the Pocket Beagle for now.
- We will tackle the Right button on the techlab later.

TechLab Buttons



Signals:

- **P1.29/PRU0.7(R.BTN)**
- **P2.33/GPIO45(L.BTN)**

Using the sysfs gpio interface for Left button

Our Left button is tied to the 13th pin on the second out of 4 gpio banks or gpio 45.

1 x 32 + 13 = 45 (banks 0-3)

```
# Since gpio 45 is already exported we don't need to do it
# echo "45" > /sys/class/gpio/export
cat /sys/class/gpio/gpio45/direction
in
# Read from button input
cat /sys/class/gpio/gpio45/value
1
```

Device Tree

- The Right button is connected to P1.29
- One of the pinmux settings connects P1.29 to PRU0.7, however this is not the default pinmux configuration
- There are several ways to update the pinmux
- The cross platform way to do this properly is by updating the DeviceTree
- However, for expediency we're going to use the beaglebone **config-pin** to setup the pin properly.

pocketbeagle.dts P1.29 pinctrl node

```
/* P1_29 (ZCZ ball A14) pruin */
P1_29_pinmux {
        compatible = "bone-pinmux-helper";
        status = "okay";
        pinctrl-names = "default", "gpio", "gpio_pu", "gpio_pd",
                        "gpio_input", "qep", "pruout", "pruin";
        pinctrl-0 = <&P1_29_default_pin>;
        pinctrl-1 = <&P1_29_gpio_pin>;
        pinctrl-2 = <&P1_29_gpio_pu_pin>;
        pinctrl-3 = <&P1_29_gpio_pd_pin>;
        pinctrl-4 = <&P1_29_gpio_input_pin>;
        pinctrl-5 = <&P1_29_qep_pin>;
        pinctrl-6 = <&P1_29_pruout_pin>;
        pinctrl-7 = <&P1_29_pruin_pin>;
};
```

Using the sysfs gpio interface for Right button

Our button is tied to the 21st pin on the fourth out of 4 gpio banks or gpio 117.

3 x 32 + 21 = 117 (banks 0-3)

```
# First set up the pinmux appropriately
config-pin p1.29 gpio
echo "117" > /sys/class/gpio/export
cat /sys/class/gpio/gpio117/direction
in
# Read from button input
cat /sys/class/gpio/gpio117/value
1
```

Using the sysfs gpio interface for output

```
# Set up GPIO 56 for USR3 and set to output
echo "56" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio56/direction
# Write output
echo "1" > /sys/class/gpio/gpio56/value
echo "0" > /sys/class/gpio/gpio56/value
# Clean up
echo "56" > /sys/class/gpio/unexport
```

sysfs led interface

- The sysfs mechanism also provides a generic led interface

```
root@beaglebone:~# ls /sys/class/leds/
beaglebone:green:usr0  beaglebone:green:usr2
beaglebone:green:usr1  beaglebone:green:usr3
root@beaglebone:~# ls /sys/class/leds/beaglebone\:green\:usr3
brightness  device  max_brightness  power  subsystem  trigger  uevent
root@beaglebone:~# cat /sys/class/leds/beaglebone\:green\:usr3/trigger
[none] rc-feedback rfkill-any kbd-scrolllock kbd-numlock kbd-capslock
kbd-kanalock kbd-shiftlock kbd-altgrlock kbd-ctrllock kbd-altlock
kbd-shiftllock kbd-shiftrlock kbd-ctrlllock kbd-ctrlrlock usb-gadget
usb-host mmc0 timer oneshot disk-activity ide-disk mtd nand-disk
heartbeat backlight gpio cpu cpu0 default-on panic
```

sysfs gpio interface is deprecated

- The sysfs gpio mechanism is actually deprecated in favour of libgpiod
- Although the libgpiod set of tools are now the preferred way of handling gpios, platforms like the PocketBeagle have their own utilities like `show-pins.pl`

```
perl /opt/scripts/device/bone/show-pins.pl -v
```

The show-pins.pl utility

```
debian@beaglebone:~$ perl /opt/scripts/device/bone/show-pins.pl -v
P8.25 / eMMC d0                    0  U7 fast rx down 7 gpio 1.00
P8.24 / eMMC d1                    1  V7 fast rx down 7 gpio 1.01
P8.05 / eMMC d2                    2  R8 fast rx down 7 gpio 1.02
P8.06 / eMMC d3                    3  T8 fast rx down 7 gpio 1.03
...
pmic irq                         112 B18 fast rx  up  0 mpu irq
jtag emu0                        121 C14 fast rx  up  0 emu 0
jtag emu1                        122 B14 fast rx  up  0 emu 1
usb A vbus en                    141 F15 fast rx down 0 usb 1 vbus out en
```

libgpiod - gpiodetect

- The libgpiod library and set of tools is now the official way of dealing with gpios.

```
debian@beaglebone:~$ gpiodetect
gpiochip0 [gpio] (32 lines)
gpiochip1 [gpio] (32 lines)
gpiochip2 [gpio] (32 lines)
gpiochip3 [gpio] (32 lines)
gpiochip4 [mcp23s18] (16 lines)
```

libgpiod - gpioinfo

- The libgpiod library and set of tools is now the official way of dealing with gpios.

```
debian@beaglebone:~$ gpioinfo
gpiochip0 - 32 lines:
        line   0:  "MDIO_DATA"       unused   input  active-high
        line   1:   "MDIO_CLK"       unused   input  active-high
...
gpiochip1 - 32 lines:
        line   0:   "GPMC_AD0"       unused   input  active-high
        line   1:   "GPMC_AD1"       unused   input  active-high
...
gpiochip2 - 32 lines:
        line   0:  "GPMC_CSN3"      "P2_20"   input  active-high [used]
        line   1:   "GPMC_CLK"      "P2_17"   input  active-high [used]
...
gpiochip3 - 32 lines:
        line   0:  "GMII1_COL"       unused   input  active-high
        line   1:  "GMII1_CRS"       unused   input  active-high
...
```

gpioset and gpioget

- **gpioset** and **gpioget** allow for a shell programatic interface to gpios without directly manipulating sysfs

```
root@beaglebone:~# gpioset -m wait gpiochip1 24=1
root@beaglebone:~# gpioget gpiochip1 13
1
```

1.2 GPIO Subsystem

Overview

- The Linux GPIO subsystem is a framework to support control of General Purpose Input/Output pins
- https://www.kernel.org/doc/Documentation/gpio/gpio.txt
- No longer using the legacy GPIO APIs.
  - https://www.kernel.org/doc/Documentation/gpio/gpio-legacy.txt
- Prefer the descriptor-based consumer GPIO APIs.
  - https://www.kernel.org/doc/Documentation/gpio/consumer.txt

Consumer

- Get a GPIO descriptor

```
struct gpio_desc *devm_gpiod_get_index(struct device *dev,
                                       const char *con_id,
                                       enum gpiod_flags flags)
```

  - **con_id** is typically the prefix of a **Device Tree gpio(s)** property. e.g. a **power-gpio** property would require **power** for **con_id**
    - https://www.kernel.org/doc/Documentation/gpio/board.txt
  - **flags** are optional and can include direction and/or initial value for a GPIO. e.g. **GPIOD_IN** for an input
- Get a GPIO value (0 for low, nonzero for high)

```
int gpiod_get_value(const struct gpio_desc *desc);
```

1.3 Labs

Exercise 1.1: List all the PocketBeagle pins and their configuration

We'll use `show-pins.pl` to do so.

Solution 1.1

```
perl /opt/scripts/device/bone/show-pins.pl -v
```

Exercise 1.2: Playing with the button

First we'll use the button to read the value of a GPIO input.

The button on the Bacon Bits is on the 13th gpio of the second gpio chip (gpiochip1), although sysfs lists this as gpiochip32, since there are 32 gpios per chip.

So the gpio number is 32 + 13, or 45. gpio45 is on by default, but otherwise we'd have to first export it to make it available in syfs.

Read both the settings and the value of gpio45 with the USR button both pressed and not pressed.

Solution 1.2

```
root@beaglebone:~# ls /sys/class/gpio/gpio45
active_low  device  direction  edge  label  power  subsystem  uevent  value
root@beaglebone:~# cat /sys/class/gpio/gpio45/active_low
0
root@beaglebone:~# cat /sys/class/gpio/gpio45/direction
in
root@beaglebone:~# cat /sys/class/gpio/gpio45/label
P2_33
root@beaglebone:~# cat /sys/class/gpio/gpio45/value
1
watch -n0 cat /sys/class/gpio/gpio45/value
```


Exercise 1.3: Read the button with libgpiod Now use gpioget from libgpiod to read the button.

Solution 1.3

```
root@beaglebone:~# gpioget gpiochip1 13
1
```

Exercise 1.4: Light up the USR3 LED with the LED class Now use /sys/class/led/* to light up the LED.

Solution 1.4

```
root@beaglebone:~# ls /sys/class/leds/beaglebone\:green\:usr3
brightness  device  max_brightness  power  subsystem  trigger  uevent
root@beaglebone:~# cat /sys/class/leds/beaglebone\:green\:usr3/brightness
0
root@beaglebone:~# echo 255 > /sys/class/leds/beaglebone\:green\:usr3/brightness
root@beaglebone:~# cat /sys/class/leds/beaglebone\:green\:usr3/brightness
255
root@beaglebone:~# echo 0 > /sys/class/leds/beaglebone\:green\:usr3/brightness
```

Exercise 1.5: Light up the USR3 LED with a LED trigger

Solution 1.5

```
root@beaglebone:~# cat /sys/class/leds/beaglebone\:green\:usr3/trigger
[none] rc-feedback rfkill-any kbd-scrolllock kbd-numlock kbd-capslock
kbd-kanalock kbd-shiftlock kbd-altgrlock kbd-ctrllock kbd-altlock
kbd-shiftllock kbd-shiftrlock kbd-ctrlllock kbd-ctrlrlock usb-gadget
usb-host mmc0 timer oneshot disk-activity ide-disk mtd nand-disk
heartbeat backlight gpio cpu cpu0 default-on panic
root@beaglebone:~# echo heartbeat > /sys/class/leds/beaglebone\:green\:usr3/trigger
root@beaglebone:~# cat /sys/class/leds/beaglebone\:green\:usr3/trigger
none rc-feedback rfkill-any kbd-scrolllock kbd-numlock kbd-capslock
```



```
kbd-kanalock kbd-shiftlock kbd-altgrlock kbd-ctrllock kbd-altlock
kbd-shiftllock kbd-shiftrlock kbd-ctrlllock kbd-ctrlrlock usb-gadget
usb-host mmc0 timer oneshot disk-activity ide-disk mtd nand-disk
[heartbeat] backlight gpio cpu cpu0 default-on panic
root@beaglebone:~# echo none > /sys/class/leds/beaglebone\:green\:usr3/trigger
```

Exercise 1.6: Light up the USR3 LED with the button as a LED trigger

Solution 1.6

```
root@beaglebone:~# cat /sys/class/leds/beaglebone\:green\:usr3/trigger
[none] rc-feedback rfkill-any kbd-scrolllock kbd-numlock kbd-capslock
kbd-kanalock kbd-shiftlock kbd-altgrlock kbd-ctrllock kbd-altlock
kbd-shiftllock kbd-shiftrlock kbd-ctrlllock kbd-ctrlrlock usb-gadget
usb-host mmc0 timer oneshot disk-activity ide-disk mtd nand-disk
heartbeat backlight gpio cpu cpu0 default-on panic
root@beaglebone:~# echo gpio > /sys/class/leds/beaglebone\:green\:usr3/trigger
root@beaglebone:~# cat /sys/class/leds/beaglebone\:green\:usr3/trigger
none rc-feedback rfkill-any kbd-scrolllock kbd-numlock kbd-capslock
kbd-kanalock kbd-shiftlock kbd-altgrlock kbd-ctrllock kbd-altlock
kbd-shiftllock kbd-shiftrlock kbd-ctrlllock kbd-ctrlrlock usb-gadget
usb-host mmc0 timer oneshot disk-activity ide-disk mtd nand-disk
heartbeat backlight [gpio] cpu cpu0 default-on panic
root@beaglebone:~# echo 45 > /sys/class/leds/beaglebone\:green\:usr3/gpio
```

Now press the button a few times. The LED should be on when the button isn't pressed and off when the buutton i pressed.

How do we reverse this configuration and have it come on when the burron is pressed.

```
root@beaglebone:~# cat /sys/class/leds/beaglebone\:green\:usr3/inverted
0
root@beaglebone:~# echo 1 > /sys/class/leds/beaglebone\:green\:usr3/inverted
```

Exercise 1.7: Turn on USR3 LED with gpio sysfs interface

Now we'll turn on USR3 which is connected to the 24th gpio gpiochip1.

So the gpio number is 32 + 24 or gpio56.

This gpio isn't yet setup so we have a little more work to do.

Solution 1.7

```
root@beaglebone:~# echo 56 > /sys/class/gpio/export
root@beaglebone:~# ls /sys/class/gpio/gpio56
active_low  device  direction  edge  label  power  subsystem  uevent  value
root@beaglebone:~# cat /sys/class/gpio/gpio56/active_low
0
root@beaglebone:~# cat /sys/class/gpio/gpio56/direction
out
root@beaglebone:~# cat /sys/class/gpio/gpio56/value
0
root@beaglebone:~# echo 1 > /sys/class/gpio/gpio56/value
root@beaglebone:~# cat /sys/class/gpio/gpio56/value
1
root@beaglebone:~# echo 0 > /sys/class/gpio/gpio56/value
```

Exercise 1.8: Light up the USR3 LED with libgpiod Now use gpioset from libgpiod to light up the LED.

Solution 1.8

```
root@beaglebone:~# gpioset -m wait gpiochip1 24=1
# Press enter to terminate
```

