GPIO and LEDs

Summary

- gpio
- /sys/class/gpio
- input and output
- interrupts

GPIO

- General Purpose Input/Output
  - very simple software-controlled digital signals
  - each bit represents a pin on the chip
  - usually each pin can be configured as an input or an output
- Typical uses
  - Control LEDs, motors, other on-board peripherals
  - Read push buttons, digital sensors of may types

Interrupts

- Some chips can generate interrupts when the GPIO pin changes state
- Interrupts may be
  - edge triggered: interrupt when the level rises or falls or both
  - level triggered: interrupt when the level is high or low
    - level triggered GPIO is rare

GPIO chips

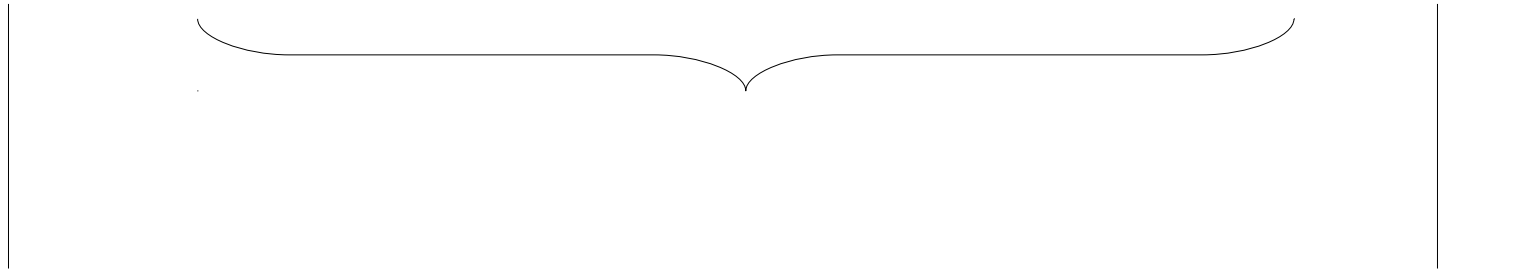
- Most SoC have dozens of GPIO pins
  - usually in banks of 16 or 32
- Power management chips (pmic) usually have some GPIO pins as well
- FPGAs are often configured to provide GPIO
- Plus, there are dedicated GPIO chips that are connected by i2c or SPI

“gpiolib”

- All this diversity is handled by the GPIO sub-system in the kernel called “gpiolib”
  - (it's not actually a library)
- It provides
  - kernel interfaces for registering GPIO chips
  - kernel interfaces for other kernel code to configure and use GPIO pins
  - a set of files in /sys/class/gpio that user-space applications can use

/sys/class/gpio

```
# ls /sys/class/gpio
export     gpiochip0    gpiochip32   gpiochip64   gpiochip96  unexport
```



Write to this file to export a GPIO to user space

This system has 4 gpio chips (each is a 32-bit register)

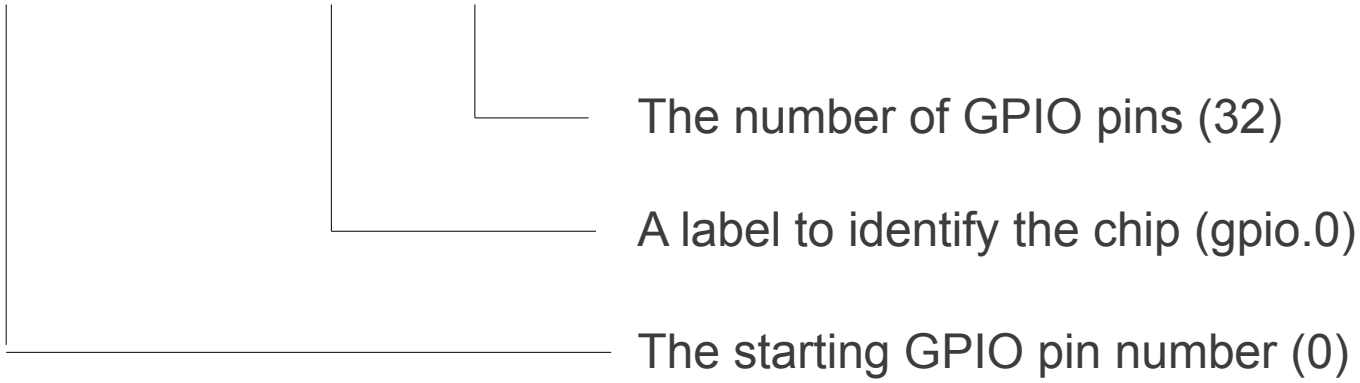
Write to this file to unexport a GPIO to user space

gpiochip

Each GPIO chip is represented by a directory gpiochipnnn where nnn is the GPIO number of the fist pin

For example this chip has 32 GPIO pins numbered from 0 to 31

```
# ls /sys/class/gpio/gpiochip0
base  device  label  ngpio  power  subsystem  uevent
```



Exporting a GPIO pin

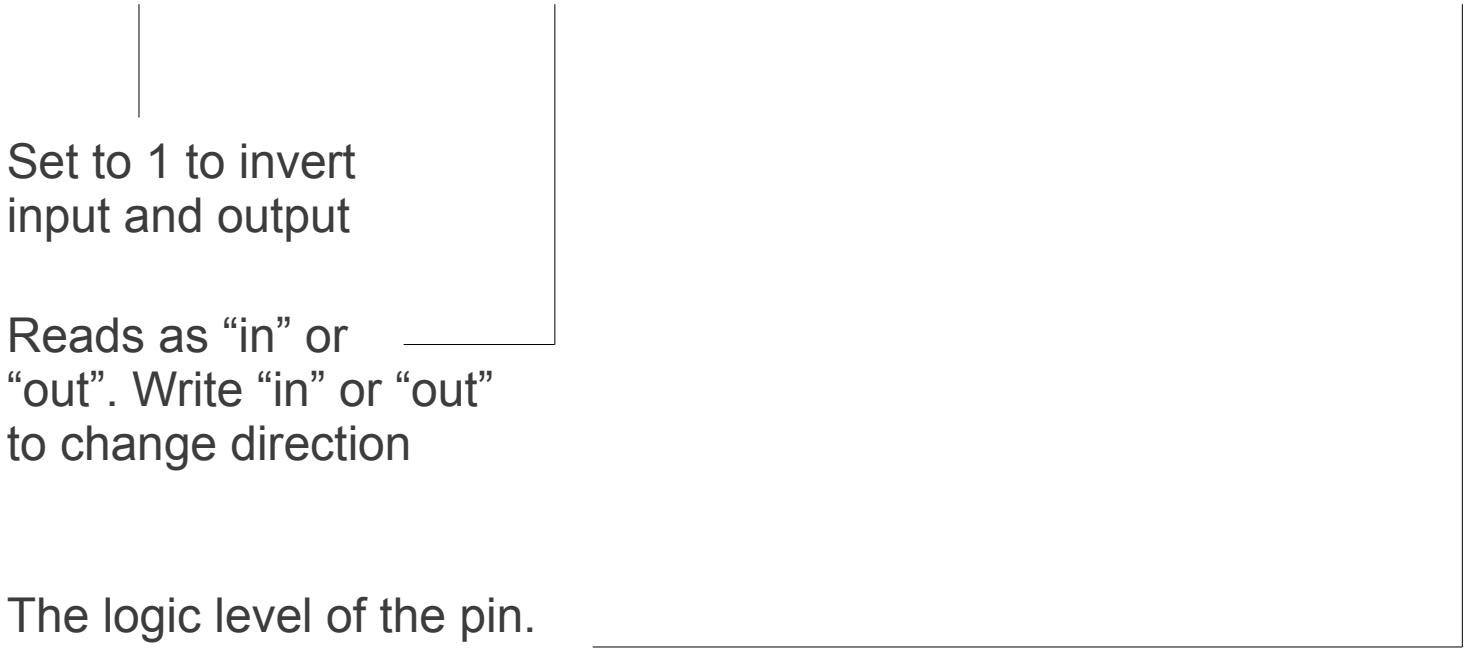
- To export a pin from the kernel to user space write the pin number to /sys/class/gpio/export

```
# echo 142 > /sys/class/gpio/export
# ls /sys/class/gpio
export     gpio142  gpiochip0    gpiochip32   gpiochip64   gpiochip96  unexport
```

If the export is successful a new directory is created through which you can interact with the pin

Inputs and outputs

```
# ls /sys/class/gpio/gpio142/
active_low  device  direction  edge  power  subsystem  uevent  value
```



Set to 1 to invert input and output

Reads as “in” or “out”. Write “in” or “out” to change direction

The logic level of the pin. For an output write 0 or 1 to set the logic level

Interrupts

```
# ls /sys/class/gpio/gpio142/
active_low  device  direction  edge  power  subsystem  uevent  value
```


If a GPIO pin can generate an interrupt the “edge” file will be present
It may take on these values

“none”	No interrupts generated (usually the default)
“rising”	Interrupt on rising edge
“falling”	Interrupt on falling edge
“both”	Interrupt on both edges

Waiting for an interrupt

To wait for an interrupt to arrive, poll the value file:

```
char *gpio_value_path = "/sys/class/gpio/gpio142/value";

int wait_for_int (void) {
    int f;
    struct pollfd poll_fds [1];

    f = open (gpio_value_path, O_RDONLY);

    poll_fds[0].fd = f;
    poll_fds[0].events = POLLPRI | POLLERR;

    res = poll (poll_fds, 1, -1);
    if (res > 0){
        printf ("Interrupt!\n");
    }
}
```

LEDs

- Many devices have LED indicators
- The LED driver system offers a consistent way to control them

/sys/class/leds

```
# ls /sys/class/leds
led0

# ls /sys/class/leds/led0
brightness  device  max_brightness  power  subsystem  trigger  uevent
```

Set the brightness. Values are from 0 to max_brightness. If the LED is either on or off, 0 = off; any other value = on

Set a source that will trigger the LED, for example

```
# cat /sys/class/leds/led0/trigger
none nand-disk [heartbeat]
```

“None” means that there is no trigger

/sys/class/backlight

- Systems with LCD screens usually have a backlight
- That also can be controlled by from user space

```
# ls /sys/class/backlight/acpi_video0
actual_brightness  brightness  max_brightness  subsystem
bl_power           device      power           uevent
```

Summary

- Most devices use GPIO pins for simple control and monitoring
- “gpiolib” is a consistent way of presenting GPIO
  - part of most BSPs these days
- Access from user space via /sys/class/gpio
- For LEDs, there is /sys/class/leds
- Also /sys/class/backlight