Wiring Pi

GPIO Interface library for the Raspberry Pi

The GPIO utility

WiringPi comes with a separate program to help manage the on-board GPIO interface as well as additional modules such as the PiFace and other devices like the Gertboard as well as generic GPIO expander type devices.

This program, called **gpio**, can also be used in scripts to manipulate the GPIO pins – set outputs and read inputs. It's even possible to write entire programs just using the **gpio** command in a shell-script, although it's not terribly efficient doing it that way… Another way to call it is using the system() function in C/C++ or it's equivalent in other programming languages.

- The **gpio** command is designed to be installed as a setuid program and called by a normal user without using the sudo command or logging in as root.

In addition to using the **gpio** utility to control, read and write the GPIO pins, you can:

- Export/Unexport pins via the */sys/class/gpio* interface, where they will then be available to user programs (that then do not need to be run as root or with sudo)
- Export pins to enable edge-triggered interrupts via the */sys/class/gpio* interface.
- Load SPI and I2C modules and set /dev/ permissions to enable read/write by the user running the **gpio** program.
- Run the i2cdetect program with appropriate flags for your Raspberry Pi board revision.
- Set the SPI buffer size and I2C baud rate (when loading the modules)
- Determine your Raspberry Pi board hardware revision.

See the man page for the **gpio** program to see what all the features are by typing

```
man gpio
```

at the command prompt.

Usage

From the Linux command line:

- **gpio -v**

  This prints the version.

- **gpio -g …**

  The optional **-g** flag causes pin numbers to be interpreted as BCM_GPIO pin numbers rather than standard ***wiringPi*** pin numbers.

- **gpio -1 …**

  The optional **-1** flag causes pin numbers to be interpreted as hardware pin numbers – this works for the **P1** connector only.

- **gpio -p …**

  The optional **-p** flag causes the **gpio** program to assume there is a PiFace board fitted to the Rasberry Pi and subsequent commands are interpreted as pins on the PiFace. Note: Pins on the PiFace are 200 through 207 for both reading and writing, with pins 208 through 215 reading the state of the output latch register (ie. you can read the state of the output pins)

- **gpio -x …**

  The optional **-x** flag causes the **gpio** program to initialise an additional expansion module. Expansion modules are defined with their name (e.g. mcp23s17), and a set of parameters separated by colons. The first parameter is always the base pin number for this expansion module. See the documentation for each module type to determine the additional parameter values and functions.

Standard input and output commands

- **gpio mode <pin> in/out/pwm/clock/up/down/tri**

  This sets the mode of a pin to be input, output, pwm or clock mode, and additionally can set the internal pull-up/down resistors to pull-up, pull-down or none.

- **gpio write <pin> 0/1**

  This sets an output pin to high (1) or low (0)

- **gpio pwm <pin> <value>**

  Set the pin to a PWM value (0-1023 is supported)

- **gpio read <pin>**

  Reads and prints the logic value of the given pin. It will print 0 (low) or 1 (high).

- **gpio awrite <pin> <value>**

  This performs an analog read from the given pin. The Pi has no on-board analog hardware so you need to specify an external module using the **-x** flag.

- **gpio aread <pin>**

  This read the analog value on the given pin. The has no on-board analog hardware so you need to specify an external module using the **-x** flag.

- **gpio readall**

  This reads all the normally accessible pins and prints a table of their numbers (wiringPi, BCM_GPIO and physical pin numbers), so makes for a handy cross-reference chart), along with their modes and current values. This command will detect the version/model of your Pi and printout the pin diagram appropriate to your Pi.

- **gpio allreadall**

  This reads all the pins on your Raspberry Pi. It's essentially the same format as used on the Compute Module.

- **gpio wfi <pin> rising/falling/both**

  This causes GPIO to perform a non-busy wait on a single GPIO pin until it changes state to that indicated.

Kernel module Load Commands

Note that these command will not work if you have enabled the device-tree interface. A warning will be printed if-so.

- **gpio load spi [buffer size in KB]**

  This loads the SPI kernel modules and optionally sets the internal buffer to the given size in KB (multiples of 1024). The default is 4KB and is usually more than enough for most application which only exchange a byte or 2 at a time over the SPI bus.

  The /dev/spi* entries are set to be owned by the person using the **gpio** program, so there is no need to run subsequent programs as root (unless they use other wiringPi functions)

- **gpio load i2c [baud rate in Kb/sec]**

  This loads the I2C kernel modules and optionally sets the baud rate to the given speed in Kb/sec (multiples of 1000). The default is 100Kb/sec.

  The /dev/I2c* entries are set to be owned by the person using the **gpio** program, so there is no need to run subsequent programs as root (unless they use other ***wiringPi*** functions)

I2C Detection

- **gpio i2cdetect**

  This runs the **i2cdetect** command, but passes in the correct parameters for the I2C bus for the Pi's hardware revision. ie. it runs **i2cdetect -y 0** on a Rev. 1 Pi, and **i2cdetect -y 1** on a Rev. 2 Pi. (You can shorten i2cdetect to i2cd)

/sys/class/gpio mode commands

- **gpio export <pin> in/out**

  This exports the given pin (BCM-GPIO pin number) as an input or output and makes it available for a user program running as the same user to use.

- **gpio unexport <pin>**

  Removes the export of the given pin.

- **gpio unexportall**

  Removes all */sys/class/gpio* exports.

- **gpio exports**

  This prints a list of all gpio pins which have been exported via the */sys/class/gpio* interface and their modes.

- **gpio edge <pin> rising/falling/both/none**

  This enables the given pin for edge interrupt triggering on the rising, falling or both edges. (Or none which disables it)

Note: The pin numbers in the sys mode are *always* BCM-GPIO pin numbers.

Examples

```
gpio mode 0 out
gpio write 0 1
```

The above uses the ***wiringPi*** pin numbers to set pin 0 as an output and then sets the pin to a logic 1.

```
gpio -g mode 17 out
gpio -g write 17 1
```

This uses the BCM_GPIO pin numbering scheme and performs the same operation as above.

```
gpio -1 mode 11 out
gpio -1 write 11 1
```

This uses the physical P1 pin numbering scheme and performs the same operation as above.

Internal pull up/down resistors

The GPIO lines have internal pull up or pull-down resistors which can be controlled via software when a pin is in input mode. There is no-way to read the status of these resistors.

```
gpio mode 0 up
gpio mode 0 down
gpio mode 0 tri
```

These set the resistors to pull-up, pull-down and none respectively on ***wiringPi*** pin 0.