**NAME**

RPi::Pin - Access and manipulate Raspberry Pi GPIO pins

**SYNOPSIS**

use [RPi::Pin](https://metacpan.org/pod/RPi::Pin);

use [RPi::Const](https://metacpan.org/pod/RPi::Const) qw(:all);

my $pin = RPi::Pin->new(5);

$pin->mode(INPUT);

$pin->write(LOW);

$pin->set\_interrupt(EDGE\_RISING, 'main::pin5\_interrupt\_handler');

my $num = $pin->num;

my $mode = $pin->mode;

my $state = $pin->read;

print "pin number $num is in mode $mode with state $state\n";

sub pin5\_interrupt\_handler {

print "in interrupt handler\n";

}

**DESCRIPTION**

An object that represents a physical GPIO pin.

Using the pin object's methods, the GPIO pins can be controlled and monitored.

This distribution can be accessed through [RPi::WiringPi](https://metacpan.org/pod/RPi::WiringPi). Using that distribution provides safety and cleanup procedures. Using this module directly requires you to reset your pins manually.

We use the BCM (GPIO) pin numbering scheme.

**METHODS**

**new($pin\_num)**

Takes the number representing the Pi's GPIO pin you want to use, and returns an object for that pin.

Parameters:

$pin\_num

Mandatory, Integer: The pin number to attach to.

**mode($mode)**

Puts the pin into either INPUT, OUTPUT, PWM\_OUT or GPIO\_CLOCK mode. If $mode is not sent in, we'll return the pin's current mode.

Parameters:

$mode

Optional: If not sent in, we'll simply return the current mode of the pin. Otherwise, send in:0 for INPUT, 1 for OUTPUT, 2 for PWM\_OUT and 3 for GPIO\_CLOCK mode.

**mode\_alt($alt)**

Allows you to set any pin to any mode.

Parameters:

$alt

Optional: If not sent in, we'll simply return the current mode of the pin. The possible values of this method are as follows:

Value Mode

------------

0 INPUT

1 OUTPUT

4 ALT0

5 ALT1

6 ALT2

7 ALT3

3 ALT4

2 ALT5

**read()**

Returns 1 if the pin is HIGH (on) and 0 if the pin is LOW (off).

write($state)

For pins in OUTPUT mode, will turn HIGH (on) the pin, or LOW (off).

Parameters:

$state

Send in 1 to turn the pin on, and 0 to turn it off.

**pull($direction)**

Used to set the internal pull-up or pull-down resistor for a pin. Calling this method on a pin will automatically set the pin to INPUT mode.

Parameter:

$direction

Mandatory: 2 for PUD\_UP, 1 for PUD\_DOWN and 0 for PUD\_OFF (disabled the resistor).

**set\_interrupt($edge, $callback)**

Listen for an interrupt on a pin, and do something if it is triggered.

Parameters:

$edge

Mandatory: 1 for EDGE\_FALLING, 2 for EDGE\_RISING, or 3 for EDGE\_BOTH.

$callback

The string name of a Perl subroutine that you've already written within your code. This is the interrupt handler. When an interrupt is triggered, the code in this subroutine will run. If you get errors when the handler is called, specify the full package name to the handler (eg: 'main::callback').

**interrupt\_set**

DEPRECATED; See set\_interrupt().

**pwm($value)**

Sets the level of the Pulse Width Modulation (PWM) of the pin. Dies if the pin's mode() is not set to PWM (2). Note that only physical pin 12 (wiringPi pin 1, GPIO pin 18) is PWM hardware capable.

Parameter:

$value

Mandatory: values range from 0-1023. 0 for 0% (off) and 1023 for 100% (fully on).

See ["pwm\_range-range" in Rpi](https://metacpan.org/pod/RPi" \l "pwm_range-range) for details on how to modify the range to something other than 0-1023.

**num()**

Returns the pin number associated with the pin object.