Module `GPIO`

Raspberry Pi GPIO binding for Lua.

A Lua binding to the (Python) library by Ben Croston to use the GPIO from Lua.

Info:

- **Copyright**: (c) 2012-2014; Ben Croston for the original Python library, Andre Simon for the initial Lua binding, Thijs Schreijer for the extensions of the Lua binding
- **Author**: Ben Croston,Andre Simon,Thijs Schreijer

Functions

cleanup ()	Cleans up the modules' running operations.
gpio_function (channel)	Gets the configuration of a pin.
input (channel)	Reads the pin value.
output (channel, value)	Sets the output of a pin.
setmode (mode)	Sets the pin numbering scheme to be used.
setup_channel (channel, direction, pull_up_down, initial)	Sets a channel up on the GPIO interface.
setwarnings (mode)	Turns warnings on or off.

Tables

constants	Constants in the module table

PWM object

ChangeDutyCycle (self, dutycycle)	Sets the dutycycle for a PWM object.
ChangeFrequency (self, freq)	Sets the frequency for a PWM object.
newPWM (channel, freq)	Creates a software PWM object.
start (self, dutycycle)	Starts the PWM mode.
stop (self)	Stops the PWM mode.

Event detection

add_event_callback (channel, callback, bouncetime)	Adds an event callback function.
add_event_detect (channel, edge, callback, bouncetime)	Adds event detection for a pin.
event_detected (channel)	Reads events detected (non-blocking).
remove_event_detect (channel)	Removes event detection for a pin.
wait_for_edge (channel, edge)	Wait for an event (blocking).

Functions

cleanup ()

Cleans up the modules' running operations. It will set all pins configured before to input.

gpio_function (channel)

Gets the configuration of a pin.

Parameters:

- `channel`: channel/pin to be reported (see **setmode**)

Returns:

Pin configuration, being `IN`, `OUT`, `I2C`, `PWM`, `SERIAL`, `SPI` or `UNKNOWN`.

input (channel)

Reads the pin value. For pins configured as output, it returns the current output value.

Parameters:

- `channel`: channel/pin to be read (see **setmode**)

Returns:

Boolean `true` for a `HIGH` value, or `false` for a `LOW` value

output (channel, value)

Sets the output of a pin.

Parameters:

- `channel`: channel/pin to be changed (see **setmode**)
- `value`: (boolean) Use a truthy value to set the pin out to `HIGH`, or falsy to set to `LOW`. NOTE: a numeric '0' is also considered falsy! for compatibility with the original Python code.

setmode (mode)

Sets the pin numbering scheme to be used.

Parameters:

- `mode`: (optional) either `BCM` (chip numbering) or `BOARD` (Rpi connector numbering)

Returns:

currently set mode, being `BCM`, `BOARD`, or `UNKNOWN`.

setup_channel (channel, direction, pull_up_down, initial)

Sets a channel up on the GPIO interface.

Parameters:

- `channel`: channel/pin to be setup (see **setmode**)
- `direction`: Sets the direction of the pin, either `IN` or `OUT`
- `pull_up_down`: (optional, only for inputs) Should the builtin pullup/down resistor be used. Either `PUD_OFF`, `PUD_DOWN`, or `PUD_UP`
- `initial`: (boolean, optional, only for outputs) Should an initial value be set? set to truthy value to set the pin out to `HIGH`, or falsy to set to `LOW`. NOTE: a numeric '0' is also considered falsy! for compatibility with the original Python code.

setwarnings (mode)

Turns warnings on or off.

Parameters:

- `mode`: if `nil` or `false` turns warnings off, or on otherwise

Tables

constants

Constants in the module table

Fields:

- `RPI_REVISION`: Revision of the Raspberry Pi board as detected (either 1 or 2)
- `VERSION`: Version of the Lua module
- `HIGH`: for setting outputs and reading inputs (see **output** and **input**)
- `LOW`: for setting outputs and reading inputs (see **output** and **input**)
- `OUT`: Pin configuration, see **setup_channel** and **gpio_function**
- `IN`: Pin configuration, see **setup_channel** and **gpio_function**
- `PWM`: Pin configuration, see **gpio_function**
- `SERIAL`: Pin configuration, see **gpio_function**
- `I2C`: Pin configuration, see **gpio_function**
- `SPI`: Pin configuration, see **gpio_function**
- `UNKNOWN`: Pin and pinmode configuration, see **gpio_function** and **setmode**
- `BOARD`: Pinmode configuration, see **setmode**
- `BCM`: Pinmode configuration, see **setmode**
- `RISING`: Event edge-type detection, see event functions
- `FALLING`: Event edge-type detection, see event functions
- `BOTH`: Event edge-type detection, see event functions

PWM object

PWM has been implemented as software PWM. Hardware PWM is not available.

ChangeDutyCycle (self, dutycycle)

Sets the dutycycle for a PWM object.

Parameters:

- `self`: PWM object to operate on
- `dutycycle`: Dutycycle to use for the object, from 0 to 100 %

Returns:

PWM object

ChangeFrequency (self, freq)

Sets the frequency for a PWM object.

Parameters:

- `self`: PWM object to operate on
- `freq`: Frequency to use for the object, in Hz.

Returns:

PWM object

newPWM (channel, freq)

Creates a software PWM object.

Parameters:

- `channel`: channel/pin to use for WPM (see **setmode**)
- `freq`: Frequency for the PWM object (in Hz)

Returns:

PWM object.

Usage:

```
local gpio = require("rpi-gpio")
local gpio.setmode(gpio.BOARD)

local Pin, Hz, Duty = 11, 100, 50  -- Pin 11, 100Hz, 50% dutycycle

gpio.setup_channel(Pin, gpio.OUT, gpio.HIGH)
local pwm = gpio.newPWM(Pin, Hz):start(Duty)
```

start (self, dutycycle)

Starts the PWM mode.

Parameters:

- `self`: PWM object to operate on
- `dutycycle`: Dutycycle to use for the object, from 0 to 100 %

Returns:

PWM object

stop (self)

Stops the PWM mode.

Parameters:

- `self`: PWM object to operate on

Returns:

PWM object

Event detection

Using event detection, the rising or falling edges of the GPIO pins can be detected. Either blocking, non-blocking or asynchroneous.

add_event_callback (channel, callback, bouncetime)

Adds an event callback function. Using this function requires the helper library `darksidesync` (async callback support).

Parameters:

- `channel`: channel/pin for which to call the callback (see **setmode**)
- `callback`: Callback function to call (a single parameter, the channel number, will be passed to the callback)
- `bouncetime`: (optional) minimum time between two callbacks in milliseconds (intermediate events will be ignored)

add_event_detect (channel, edge, callback, bouncetime)

Adds event detection for a pin. Using this function with a callback (which is optional) requires the helper library `darksidesync` (async callback support).

Parameters:

- `channel`: channel/pin to detect events for (see **setmode**)
- `edge`: What type of edge to catch events for. Either `RISING`, `FALLING` or `BOTH`.
- `callback`: (optional) Callback function to call on the event (a single parameter, the channel number, will be passed to the callback). More can be added using **add_event_callback** .
- `bouncetime`: (optional) minimum time between two callbacks in milliseconds (intermediate events will be ignored)

event_detected (channel)

Reads events detected (non-blocking). Pins must first be configured using **add_event_detect** , events will be queued, so **event_detected** will not miss events.

Parameters:

- `channel`: channel/pin to check for events (see **setmode**)

Returns:

boolean, `true` if an event was detected, `false` otherwise

remove_event_detect (channel)

Removes event detection for a pin.

Parameters:

- `channel`: channel/pin to stop detecting events for (see **setmode**)

wait_for_edge (channel, edge)

Wait for an event (blocking).

Parameters:

- `channel`: channel/pin to check for events (see **setmode**)
- `edge`: What type of edge to wait for. Either `RISING`, `FALLING` or `BOTH`.