General Purpose Input / Output

|  |  |
| --- | --- |
| Status : working, ~~validated~~, ~~documented~~ | 04/08/2016 |

# Introduction

This library provides an API for General Purpose Input Output (GPIO) pins.

## State of the library

This library is currently usable but not fully validated.

## Origin of the library

The GPIO library originates with the official Espressif SDK. Several source files were merged to provide a single library for controlling the ESP8266 I/O pins. This includes:

* Espressif’s “gpio.h” and “gpio.c” files
* Espressif’s “gpio16.h” and “gpio16.c” files

GPIO 16 is very different form GPIO 0 through 15 in that it is actually part of a different internal peripheral and is controlled through different registers. It also has reduced functionality, and notably lacks interrupt support. I consider

Relatively few modifications to the actual code were necessary. The most important modification is FreeRTOS support.

# Using the library

## Building

By default, the library’s source code is part of the ESP8266 template project and will compile and link with your code. All that may be required for building is to **#include “gpio.h”** in your own source files.

## Using GPIO

Typical GPIO programming requires an initialization step to set the pin(s) direction and other characteristics (open drain, interrupt source, etc…), then pin states can be set or read.

On the ESP8266 GPIO is essentially provided by two different peripherals:

* A dedicated GPIO peripheral provides up to 16 GPIO (GPIO 0 through 15)
* The RTC provides a seventeenth GPIO, GPIO16.

GPIO16 has limited functionality, most notably it cannot be used as an interrupt source. It is best used for controlling discrete inputs on external peripheral devices.

## GPIO Interrupts

For a complete example of how to write GPIO interrupt code, read this article:

<http://www.nefastor.com/esp8266-gpio-programming/>

# API

## gpio\_config

## gpio\_output\_conf

## gpio\_input\_get

Returns the current state of pins GPIO 0 to 15 as an integer.

You will need to use bit masks to get the value of individual pins or sets of pins.

## gpio\_intr\_handler\_register

|  |  |
| --- | --- |
| Sets the GPIO interrupt service routine | |
| Status | Validated |
| Returns | - |
| Argument | void\* (pointer to the function to set as GPIO ISR) |
| Argument | void\* (pointer to an argument to pass to the GPIO ISR upon a GPIO interrupt) |

This function must be called prior to enabling GPIO interrupts.

Sets the function that will be called upon a GPIO interrupt. The same function will be called no matter which pin triggered an interrupt: it is up to the ISR to determine which pin(s) triggered.

The function to be used as GPIO ISR must have this prototype:

void *name\_of\_your\_ISR* (uint32 **mask**, void\* **argument**);

The **mask** contains the states of GPIO 0..15 at the time the interrupt was triggered. The **argument** pointer has the same value provided when the ISR was registered.

In other words:

* Use **mask** to determine which pin(s) triggered an interrupt
* Use **argument** to exchange data between your normal code and your ISR

For an example of how to use this function and how to write a GPIO ISR, read this article:

<http://www.nefastor.com/esp8266-gpio-programming/>

## gpio\_pin\_wakeup\_enable

Status: untested.

## gpio\_pin\_wakeup\_disable

## gpio\_pin\_intr\_state\_set

## gpio16\_output\_conf

|  |  |
| --- | --- |
| Sets pin GPIO16 as an output | |
| Status | Validated |
| Returns | - |
| Argument | - |

GPIO16 (NodeMCU D0) is part of the RTC peripheral and is controlled through different registers, thus requiring dedicated API calls.

## gpio16\_output\_set

|  |  |
| --- | --- |
| Sets the state of pin GPIO16 | |
| Status | Validated |
| Returns | - |
| Argument | uint8 **value** |

Sets the state of the GPIO16 pin according to **value** (0 or 1).

Due to the way this function is coded, a non-zero but even **value** will not set the pin.

## gpio16\_input\_conf

|  |  |
| --- | --- |
| Sets pin GPIO16 as an input | |
| Status | Untested |
| Returns | - |
| Argument | - |

## gpio16\_input\_get

|  |  |
| --- | --- |
| Reads the state of the GPIO16 pin | |
| Status | Untested |
| Returns | uint8 |
| Argument | - |

Returns the state of the GPIO16 pin (0 or 1).

Behavior when GPIO16 is set as an output: untested.

## ETS\_GPIO\_INTR\_ENABLE

|  |  |
| --- | --- |
| Macro to enable GPIO interrupts | |
| Status | Validated |
| Returns | - |
| Argument | - |

Must be called only after registering an interrupt service routine with **gpio\_intr\_handler\_register**.

## GPIO\_OUTPUT\_SET

|  |  |
| --- | --- |
| Macro to set or clear a pin | |
| Status | Validated |
| Returns | - |
| Argument | ESP8266 GPIO pin number (0 to 15) |
| Argument | Pin state (0 or 1) |

## GPIO\_OUTPUT

Macro.

## GPIO\_DIS\_OUTPUT

Macro

## GPIO\_AS\_INPUT

Macro

## GPIO\_AS\_OUTPUT

Macro

## GPIO\_INPUT\_GET

Macro