**class Pin – control I/O pins**

A pin is the basic object to control I/O pins (also known as GPIO - general-purpose input/output). It has methods to set the mode of the pin (input, output, etc) and methods to get and set the digital logic level. For analog control of a pin, see the ADC class.

**Quick usage example**

**from** machine **import** Pin

*# initialize ``P9`` in gpio mode and make it an output*

p\_out **=** Pin('P9', mode**=**Pin**.**OUT)

p\_out**.**value(1)

p\_out**.**value(0)

p\_out**.**toggle()

p\_out(**True**)

*# make ``P10`` an input with the pull-up enabled*

p\_in **=** Pin('P10', mode**=**Pin**.**IN, pull**=**Pin**.**PULL\_UP)

p\_in() *# get value, 0 or 1*

**Constructors**

***class*machine.Pin(*id*, *...*)**

Create a new Pin object associated with the string id. If additional arguments are given, they are used to initialize the pin. See **Pin.init()**.

**from** machine **import** Pin

p **=** Pin('P10', mode**=**Pin**.**OUT, pull**=None**, alt**=-**1)

**Methods**

**pin.init(*mode*, *pull*, *\**, *alt*)**

Initialize the pin:

* **mode** can be one of:
  + **Pin.IN** - input pin.
  + **Pin.OUT** - output pin in push-pull mode.
  + **Pin.OPEN\_DRAIN** - input or output pin in open-drain mode.
* **pull** can be one of:
  + **None** - no pull up or down resistor.
  + **Pin.PULL\_UP** - pull up resistor enabled.
  + **Pin.PULL\_DOWN** - pull down resistor enabled.
* **alt** is the id of the alternate function.

Returns: **None**.

**pin.id()**

Get the pin id.

**pin.value([*value*])**

Get or set the digital logic level of the pin:

* With no argument, return 0 or 1 depending on the logic level of the pin.
* With **value** given, set the logic level of the pin. **value** can be anything that converts to a boolean. If it converts to **True**, the pin is set high, otherwise it is set low.

**pin([*value*])**

Pin objects are callable. The call method provides a (fast) shortcut to set and get the value of the pin.

Example:

**from** machine **import** Pin

pin **=** Pin('P12', mode**=**Pin**.**IN, pull**=**Pin**.**PULL\_UP)

pin() *# fast method to get the value*

See **[pin.value()](https://docs.pycom.io/pycom_esp32/library/machine.Pin.html" \l "machine.pin.value" \o "machine.pin.value)** for more details.

**pin.toggle()**

Toggle the value of the pin.

**pin.mode([*mode*])**

Get or set the pin mode.

**pin.pull([*pull*])**

Get or set the pin pull.

**pin.hold([*hold*])**

Get or set the pin hold. Can be used to retain the pin state through a core reset and system reset triggered by watchdog time-out or Deep-sleep events.

**pin.callback(*trigger*, *handler=None*, *arg=None*)**

Set a callback to be triggered when the input level at the pin changes.

* **trigger** is the type of event that triggers the callback. Possible values are:
  + **Pin.IRQ\_FALLING** interrupt on falling edge.
  + **Pin.IRQ\_RISING** interrupt on rising edge.
  + **Pin.IRQ\_LOW\_LEVEL** interrupt on low level.
  + **Pin.IRQ\_HIGH\_LEVEL** interrupt on high level.

The values can be *ORed* together, for instance trigger=Pin.IRQ\_FALLING | Pin.IRQ\_RISING

* **handler** is the function to be called when the event happens. This function will receive one argument. Set **handler** to **None** to disable it.
* **arg** is an optional argument to pass to the callback. If left empty or set to **None**, the function will receive the **Pin** object that triggered it.

Example:

**from** machine **import** Pin

**def** **pin\_handler**(arg):

print("got an interrupt in pin %s" **%** (arg**.**id()))

p\_in **=** Pin('P10', mode**=**Pin**.**IN, pull**=**Pin**.**PULL\_UP)

p\_in**.**callback(Pin**.**IRQ\_FALLING **|** Pin**.**IRQ\_RISING, pin\_handler)

**Note**

For more information on how Pycom’s products handle interrupts, see [here](https://docs.pycom.io/pycom_esp32/pycom_esp32/toolsandfeatures.html#pycom-interrupt-handling).

**Attributes**

***class*pin.exp\_board**

Contains all **Pin** objects supported by the expansion board. Examples:

Pin**.**exp\_board**.**G16

led **=** Pin(Pin**.**exp\_board**.**G16, mode**=**Pin**.**OUT)

Pin**.**exp\_board**.**G16**.**id()

***class*pin.module**

Contains all **Pin** objects supported by the module. Examples:

Pin**.**module**.**P9

led **=** Pin(Pin**.**module**.**P9, mode**=**Pin**.**OUT)

Pin**.**module**.**P9**.**id()

**Constants**

The following constants are used to configure the pin objects. Note that not all constants are available on all ports.

**Pin.IN**

**Pin.OUT**

**Pin.OPEN\_DRAIN**

Selects the pin mode.

**Pin.PULL\_UP**

**Pin.PULL\_DOWN**

Enables the pull up or pull down resistor.