class Timer – control hardware timers

Hardware timers deal with timing of periods and events. Timers are perhaps the most flexible and heterogeneous kind of hardware in MCUs and SoCs, differently greatly from a model to a model. MicroPython's Timer class defines a baseline operation of executing a callback with a given period (or once after some delay), and allow specific boards to define more non-standard behavior (which thus won't be portable to other boards).

See discussion of important constraints on Timer callbacks.

Note

Memory can't be allocated inside irq handlers (an interrupt) and so exceptions raised within a handler don't give much information. See `micropython.alloc_emergency_exception_buf()` for how to get around this limitation.

If you are using a WiPy board please refer to machine.TimerWiPy instead of this class.

Constructors

class `machine.Timer`(*id, ...*)

Construct a new timer object of the given id. Id of -1 constructs a virtual timer (if supported by a board).

Methods

`Timer.init`(**, mode=Timer.PERIODIC, period=-1, callback=None*)

Initialise the timer. Example:

```
tim.init(period=100)                         # periodic with 100ms period
tim.init(mode=Timer.ONE_SHOT, period=1000)   # one shot firing after 1000ms
```

Keyword arguments:

- `mode` can be one of:
  - `Timer.ONE_SHOT` - The timer runs once until the configured period of the channel expires.
  - `Timer.PERIODIC` - The timer runs periodically at the configured frequency of the channel.

`Timer.deinit()`

Deinitialises the timer. Stops the timer, and disables the timer peripheral.

Constants

`Timer.ONE_SHOT`

`Timer.PERIODIC`

Timer operating mode.