uRTC Documentation

Release 1.2

Radomir Dopieralski

Sep 27, 2017

Contents

This is a MicroPython library that makes it easy to communicate with a number of different real-time clock modules.

Contents:

CHAPTER 1

Installation

In order to install this library, you need to copy the `urtc.py` file onto your board, so that it can be imported.

ESP8266 Port, Binary Releases

The easiest way to make this library importable with the binary releases of MicroPython for ESP8266 is to upload the `urtc.py` file to the board using the WebREPL utility.

ESP8266 Port, Compiled Firmware

If you are compiling your own firmware, the easiest way to include this library in it is to copy or symlink the `urtc.py` file to the `esp8266/scripts` directory before compilation.

Other Ports

For boards that are visible as USB mass storage after connecting to your computer, simply copy the `urtc.py` file onto that storage.

CHAPTER 2

urtc module

DS1307

class urtc.**DS1307**(*i2c*, *address=0x68*)

datetime(*datetime*)
Get or set the current time.

The `datetime` is an 8-tuple of the format `(year, month, day, weekday, hour, minute, second, millisecond)` describing the time to be set. If not specified, the method returns a tuple in the same format.

memory(*address*, *buffer=None*)
Read or write the non-volatile random acces memory.

stop(*value=None*)
Get or set the status of the stop clock flag. This can be used to start the clock at a precise moment in time.

DS3231

class urtc.**DS3231**(*i2c*, *address=0x68*)

datetime(*datetime*)
Get or set the current time.

The `datetime` is an 8-tuple of the format `(year, month, day, weekday, hour, minute, second, millisecond)` describing the time to be set. If not specified, the method returns a tuple in the same format.

alarm_time(*self*, *datetime=None*)
Get or set the alarm time.

The `datetime` is a tuple in the same format as for `datetime()` method.

Only `day`, `hour`, `minute` and `weekday` values are used, the rest is ignored. If a value is `None`, it will also be ignored. When the values match the current date and time, the alarm will be triggered.

lost_power()
Return `True` if the clock lost the power recently and needs to be re-set.

alarm(*value=None*)
Get or set the value of the alarm flag. This is set to `True` when an alarm is triggered, and has to be cleared manually.

stop(*value=None*)
Get or set the status of the stop clock flag. This can be used to start the clock at a precise moment in time.

PCF8523

class urtc.**PCF8523**(*i2c*, *address=0x68*)

datetime(*datetime*)
Get or set the current time.

The `datetime` is an 8-tuple of the format `(year, month, day, weekday, hour, minute, second, millisecond)` describing the time to be set. If not specified, the method returns a tuple in the same format.

alarm_time(*self*, *datetime=None*)
Get or set the alarm time.

The `datetime` is a tuple in the same format as for `datetime()` method.

Only `day`, `hour`, `minute` and `weekday` values are used, the rest is ignored. If a value is `None`, it will also be ignored. When the values match the current date and time, the alarm will be triggered.

lost_power()
Return `True` if the clock lost the power recently and needs to be re-set.

alarm(*value=None*)
Get or set the value of the alarm flag. This is set to `True` when an alarm is triggered, and has to be cleared manually.

stop(*value=None*)
Get or set the status of the stop clock flag. This can be used to start the clock at a precise moment in time.

reset()
Perform a software reset of the clock module.

battery_low()
Return `True` if the battery is discharged and needs to be replaced.

Utilities

class urtc.**DateTimeTuple**
A `NamedTuple` of the format required by the `datetime` methods.

urtc.**datetime_tuple**(*year*, *month*, *day*, *weekday*, *hour*, *minute*, *second*, *millisecond*)
A factory function for *DateTimeTuple*.

urtc.**tuple2seconds**(*datetime*)
Convert datetime tuple into seconds since Jan 1, 2000.

urtc.**seconds2tuple**()
Convert seconds since Jan 1, 2000 into a *DateTimeTuple*.

CHAPTER 3

Usage Examples

To use this library with an Adalogger FeatherWing on Adafruit Feather HUZZAH with ESP8266, you can use the following code:

```
import urtc
from machine import I2C, Pin
i2c = I2C(Pin(5), Pin(4)) # The SCL is on GPIO5, the SDA is on GPIO4
rtc = urtc.PCF8523(i2c)
```

now you can set the current time:

```
datetime = urtc.datetime_tuple(year=2016, month=8, day=14)
rtc.datetime(datetime)
```

or read it:

```
datetime = rtc.datetime()
print(datetime.year)
print(datetime.month)
print(datetime.day)
print(datetime.weekday)
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

u

Index