

---

# **uRTC Documentation**

***Release 1.2***

**Radomir Dopieralski**

**Sep 27, 2017**



---

# Contents

---

<b>1</b>	<b>Installation</b>	<b>3</b>
1.1	ESP8266 Port, Binary Releases . . . . .	3
1.2	ESP8266 Port, Compiled Firmware . . . . .	3
1.3	Other Ports . . . . .	3
<b>2</b>	<b>urtc module</b>	<b>5</b>
2.1	DS1307 . . . . .	5
2.2	DS3231 . . . . .	5
2.3	PCF8523 . . . . .	6
2.4	Utilities . . . . .	6
<b>3</b>	<b>Usage Examples</b>	<b>9</b>
<b>4</b>	<b>Indices and tables</b>	<b>11</b>
	<b>Python Module Index</b>	<b>13</b>



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

Contents:



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.





### 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 access 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*, *alarm=0*)

Get or set the alarm time.

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

The `alarm` is the id of the alarm, it can be either 0 or 1.

For alarm 1, only `day`, `hour`, `minute` and `weekday` values are used, the rest is ignored. Alarm 0 additionally supports seconds. 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, alarm=0*)

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 HUZAZH with ESP8266, you can use the following code:

```
import urtc
from machine import I2C, Pin
i2c = I2C(scl=Pin(5), sda=Pin(4))
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`





**u**

urtc, 5



## A

alarm() (urtc.DS3231 method), 6  
alarm() (urtc.PCF8523 method), 6  
alarm\_time() (urtc.DS3231 method), 5  
alarm\_time() (urtc.PCF8523 method), 6

## B

battery\_low() (urtc.PCF8523 method), 6

## D

datetime() (urtc.DS1307 method), 5  
datetime() (urtc.DS3231 method), 5  
datetime() (urtc.PCF8523 method), 6  
datetime\_tuple() (in module urtc), 6  
DateTimeTuple (class in urtc), 6  
DS1307 (class in urtc), 5  
DS3231 (class in urtc), 5

## L

lost\_power() (urtc.DS3231 method), 6  
lost\_power() (urtc.PCF8523 method), 6

## M

memory() (urtc.DS1307 method), 5

## P

PCF8523 (class in urtc), 6

## R

reset() (urtc.PCF8523 method), 6

## S

seconds2tuple() (in module urtc), 7  
stop() (urtc.DS1307 method), 5  
stop() (urtc.DS3231 method), 6  
stop() (urtc.PCF8523 method), 6

## T

tuple2seconds() (in module urtc), 7

## U

urtc (module), 5