

---

# **Adafruit DS1307 Library Documentation**

***Release 1.0***

**Phillip Moyer**

**Sep 24, 2019**



---

## Contents

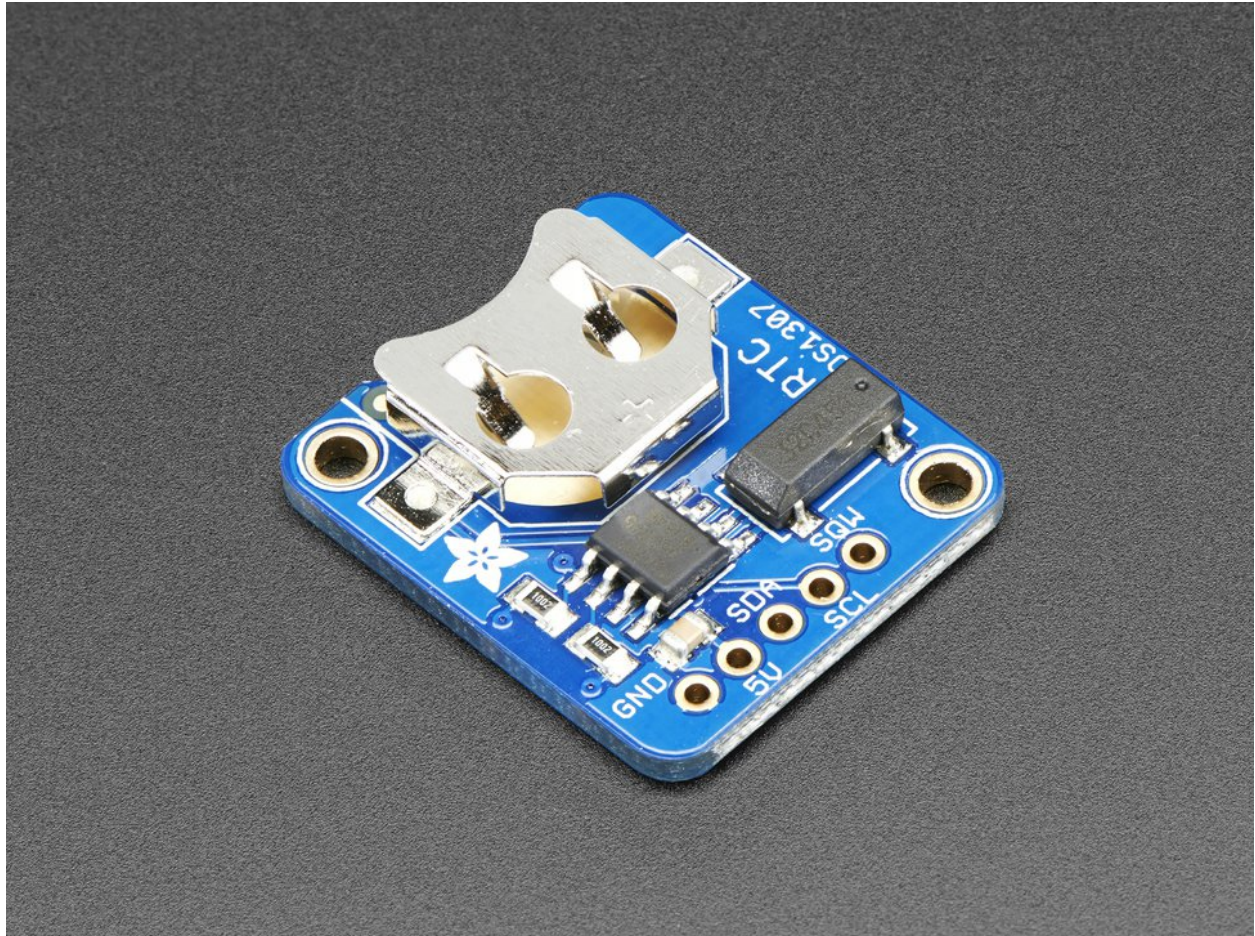
---

<b>1</b>	<b>Dependencies</b>	<b>3</b>
<b>2</b>	<b>Usage Notes</b>	<b>5</b>
<b>3</b>	<b>Contributing</b>	<b>7</b>
<b>4</b>	<b>Building locally</b>	<b>9</b>
4.1	Sphinx documentation . . . . .	9
<b>5</b>	<b>Table of Contents</b>	<b>11</b>
5.1	Simple test . . . . .	11
5.2	adafruit_ds1307 - DS1307 Real Time Clock module . . . . .	12
5.2.1	Implementation Notes . . . . .	12
<b>6</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>
	<b>Index</b>	<b>19</b>



This is a great battery-backed real time clock (RTC) that allows your microcontroller project to keep track of time even if it is reprogrammed, or if the power is lost. Perfect for datalogging, clock-building, time stamping, timers and alarms, etc. The DS1307 is the most popular RTC - but it requires 5V power to work.

The DS1307 is simple and inexpensive but not a high precision device. It may lose or gain up to two seconds a day. For a high-precision, temperature compensated alternative, please check out the [DS3231 precision RTC](#). If you do not need a DS1307, or you need a 3.3V-power/logic capable RTC please check out our affordable [PCF8523 RTC breakout](#).





# CHAPTER 1

---

## Dependencies

---

This driver depends on:

- [Adafruit CircuitPython](#)
- [Bus Device](#)
- [Register](#)

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the [Adafruit library and driver bundle](#).





## CHAPTER 2

---

### Usage Notes

---

Of course, you must import the library to use it:

```
import busio
import adafruit_ds1307
import time
```

All the Adafruit RTC libraries take an instantiated and active I2C object (from the `busio` library) as an argument to their constructor. The way to create an I2C object depends on the board you are using. For boards with labeled SCL and SDA pins, you can:

```
from board import *
```

You can also use pins defined by the onboard microcontroller through the `microcontroller.pin` module.

Now, to initialize the I2C bus:

```
myI2C = busio.I2C(SCL, SDA)
```

Once you have created the I2C interface object, you can use it to instantiate the RTC object:

```
rtc = adafruit_ds1307.DS1307(myI2C)
```

To set the time, you need to set `datetime` to a `time.struct_time` object:

```
rtc.datetime = time.struct_time((2017, 1, 9, 15, 6, 0, 0, 9, -1))
```

After the RTC is set, you retrieve the time by reading the `datetime` attribute and access the standard attributes of a `struct_time` such as `tm_year`, `tm_hour` and `tm_min`.

```
t = rtc.datetime
print(t)
print(t.tm_hour, t.tm_min)
```



## CHAPTER 3

---

### Contributing

---

Contributions are welcome! Please read our [Code of Conduct](#) before contributing to help this project stay welcoming.



## CHAPTER 4

---

### Building locally

---

To build this library locally you'll need to install the `circuitpython-build-tools` package.

```
python3 -m venv .env
source .env/bin/activate
pip install circuitpython-build-tools
```

Once installed, make sure you are in the virtual environment:

```
source .env/bin/activate
```

Then run the build:

```
circuitpython-build-bundles --filename_prefix adafruit-circuitpython-ds1307 --library_
↪location .
```

### 4.1 Sphinx documentation

Sphinx is used to build the documentation based on rST files and comments in the code. First, install dependencies (feel free to reuse the virtual environment from above):

```
python3 -m venv .env
source .env/bin/activate
pip install Sphinx sphinx-rtd-theme
```

Now, once you have the virtual environment activated:

```
cd docs
sphinx-build -E -W -b html . _build/html
```

This will output the documentation to `docs/_build/html`. Open the `index.html` in your browser to view them. It will also (due to `-W`) error out on any warning like Travis will. This is a good way to locally verify it will pass.



## 5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/ds1307\_simpletest.py

```
1  # Simple demo of reading and writing the time for the DS1307 real-time clock.
2  # Change the if False to if True below to set the time, otherwise it will just
3  # print the current date and time every second. Notice also comments to adjust
4  # for working with hardware vs. software I2C.
5
6  import time
7  import board
8  # For hardware I2C (M0 boards) use this line:
9  import busio as io
10 # Or for software I2C (ESP8266) use this line instead:
11 #import bitbangio as io
12
13 import adafruit_ds1307
14
15 # Change to the appropriate I2C clock & data pins here!
16 i2c_bus = io.I2C(board.SCL, board.SDA)
17
18 # Create the RTC instance:
19 rtc = adafruit_ds1307.DS1307(i2c_bus)
20
21 # Lookup table for names of days (nicer printing).
22 days = ("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")
23
24
25 #pylint: disable-msg=bad-whitespace
26 #pylint: disable-msg=using-constant-test
27 if False: # change to True if you want to set the time!
```

(continues on next page)

(continued from previous page)

```

28     #                               year, mon, date, hour, min, sec, wday, yday, isdst
29     t = time.struct_time((2017, 10, 29, 15, 14, 15, 0, -1, -1))
30     # you must set year, mon, date, hour, min, sec and weekday
31     # yearday is not supported, isdst can be set but we don't do anything with it at_
→ this time
32     print("Setting time to:", t)      # uncomment for debugging
33     rtc.datetime = t
34     print()
35     #pylint: enable-msg=using-constant-test
36     #pylint: enable-msg=bad-whitespace
37
38     # Main loop:
39     while True:
40         t = rtc.datetime
41         #print(t)      # uncomment for debugging
42         print("The date is {} {}/{} / {}".format(days[int(t.tm_wday)], t.tm_mday, t.tm_mon,
→ t.tm_year))
43         print("The time is {}:02:02".format(t.tm_hour, t.tm_min, t.tm_sec))
44         time.sleep(1) # wait a second

```

## 5.2 adafruit\_ds1307 - DS1307 Real Time Clock module

CircuitPython library to support DS1307 Real Time Clock (RTC).

This library supports the use of the DS1307-based RTC in CircuitPython.

Beware that most CircuitPython compatible hardware are 3.3v logic level! Make sure that the input pin is 5v tolerant.

- Author(s): Philip R. Moyer and Radomir Dopieralski for Adafruit Industries

### 5.2.1 Implementation Notes

#### Hardware:

- Adafruit [DS1307 RTC breakout](#) (Product ID: 3296)

#### Software and Dependencies:

- Adafruit CircuitPython firmware (0.8.0+) for the ESP8622 and M0-based boards: <https://github.com/adafruit/circuitpython/releases>
- Adafruit's Register library: [https://github.com/adafruit/Adafruit\\_CircuitPython\\_Register](https://github.com/adafruit/Adafruit_CircuitPython_Register)
- Adafruit's Bus Device library: [https://github.com/adafruit/Adafruit\\_CircuitPython\\_BusDevice](https://github.com/adafruit/Adafruit_CircuitPython_BusDevice)

#### Notes:

1. Milliseconds are not supported by this RTC.
2. Alarms and timers are not supported by this RTC.
3. Datasheet: <https://datasheets.maximintegrated.com/en/ds/DS1307.pdf>

**class** `adafruit_ds1307.DS1307` (*i2c\_bus*)

Interface to the DS1307 RTC.

#### **datetime**

Gets the current date and time or sets the current date and time then starts the clock.



**datetime\_register**

Current date and time.

**disable\_oscillator**

True if the oscillator is disabled.



## CHAPTER 6

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### **a**

adafruit\_ds1307, [12](#)



## A

`adafruit_ds1307` (*module*), [12](#)

## D

`datetime` (*adafruit\_ds1307.DS1307 attribute*), [12](#)

`datetime_register` (*adafruit\_ds1307.DS1307 attribute*), [12](#)

`disable_oscillator` (*adafruit\_ds1307.DS1307 attribute*), [13](#)

`DS1307` (*class in adafruit\_ds1307*), [12](#)