Adafruit DS1307 Library Documentation

Release 1.0

Phiilip Moyer

Mar 11, 2018

Contents

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.
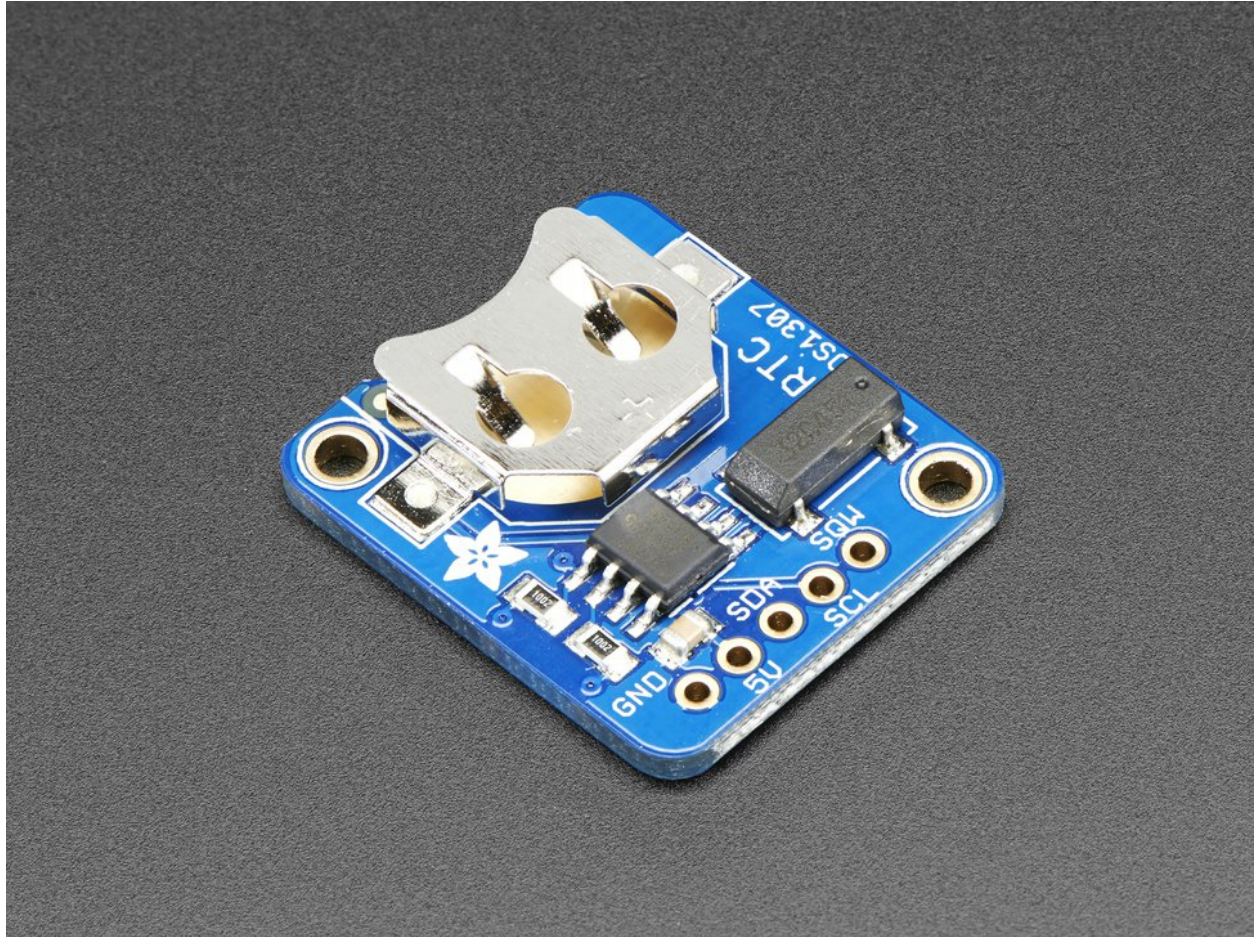
Dependencies

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.

Usage Notes

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

```python
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:

```python
from board import *
```

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

Now, to initialize the I2C bus:

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

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

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

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

```python
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`.

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

Contributing

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

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.

Table of Contents

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/ds1307_simpletest.py

```python
# Simple demo of reading and writing the time for the DS1307 real-time clock.
# Change the if False to if True below to set the time, otherwise it will just
# print the current date and time every second.  Notice also comments to adjust
# for working with hardware vs. software I2C.

import time
import board
# For hardware I2C (M0 boards) use this line:
import busio as io
# Or for software I2C (ESP8266) use this line instead:
#import bitbangio as io

import adafruit_ds1307

# Change to the appropriate I2C clock & data pins here!
i2c_bus = io.I2C(board.SCL, board.SDA)

# Create the RTC instance:
rtc = adafruit_ds1307.DS1307(i2c_bus)

# Lookup table for names of days (nicer printing).
days = ("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")


#pylint: disable-msg=bad-whitespace
#pylint: disable-msg=using-constant-test
if False:    # change to True if you want to set the time!
```

```
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
- Adafruit's Bus Device library: 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

Python Module Index

a

Index

A

D