Adafruit DS1307 Library Documentation

Release 1.0

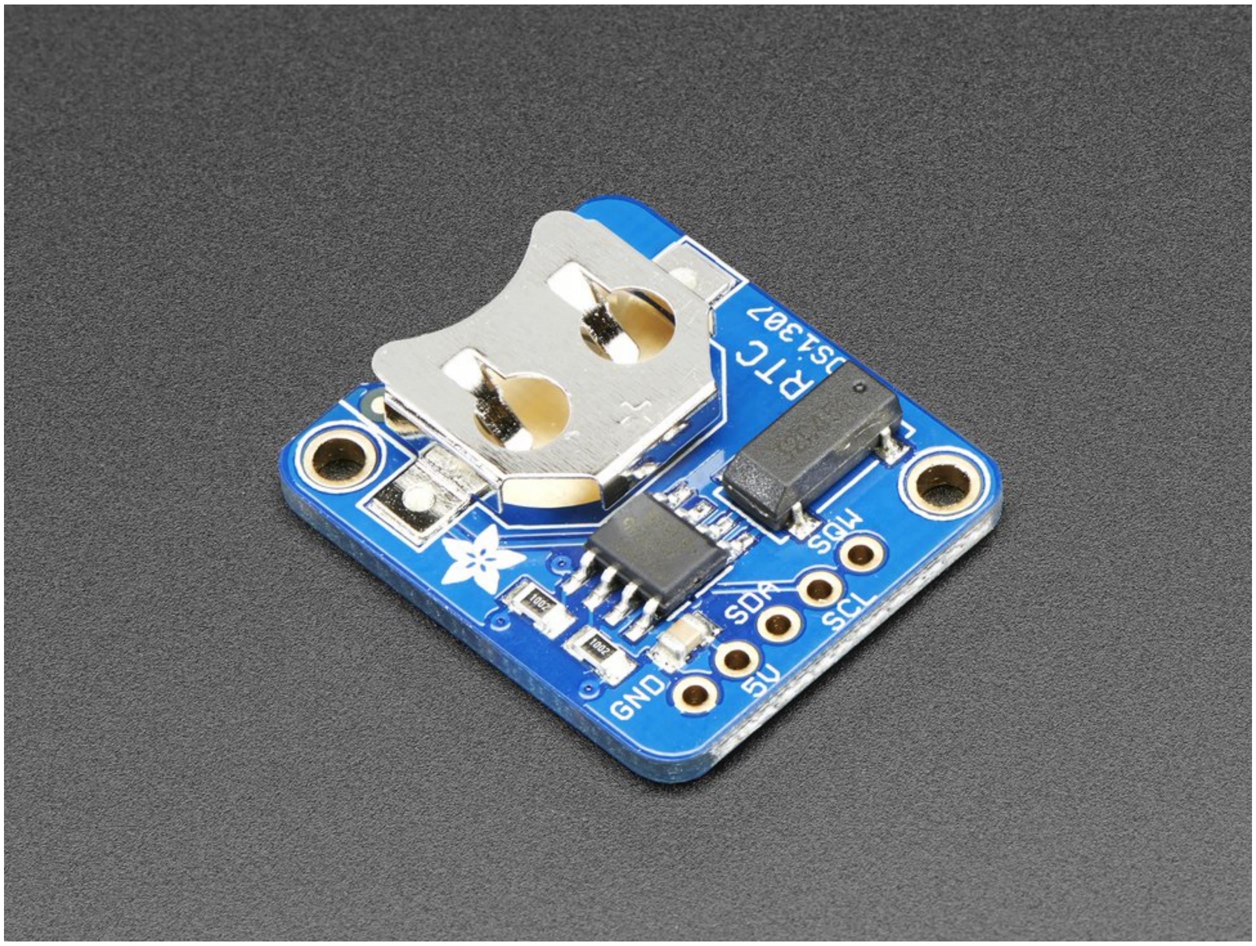
Phiilip Moyer

Oct 25, 2021

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.



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

Installing from PyPI

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally from PyPI. To install for current user:

```
pip3 install adafruit-circuitpython-ds1307
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-ds1307
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .env
source .env/bin/activate
pip3 install adafruit-circuitpython-ds1307
```

CHAPTER 3

Usage Notes

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

```
import board
import adafruit_ds1307
import time
```

All the Adafruit RTC libraries take an instantiated and active I2C object (from the `board` 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:

```
import board
```

Now, to initialize the I2C bus:

```
i2c = board.I2C()
```

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

```
rtc = adafruit_ds1307.DS1307(i2c)
```

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 4

Documentation

API documentation for this library can be found on Read the Docs.

CHAPTER 5

Contributing

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

CHAPTER 6

Documentation

For information on building library documentation, please check out this guide.

CHAPTER 7

Table of Contents

7.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/ds1307_simpletest.py

```python
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

# 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
import adafruit_ds1307

i2c = board.I2C()
rtc = adafruit_ds1307.DS1307(i2c)

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


# pylint: disable-msg=using-constant-test
if False:  # change to True if you want to set the time!
    #                     year, mon, date, hour, min, sec, wday, yday, isdst
    t = time.struct_time((2017, 10, 29, 15, 14, 15, 0, -1, -1))
    # you must set year, mon, date, hour, min, sec and weekday
    # yearday is not supported, isdst can be set but we don't do anything with it at
↪this time
    print("Setting time to:", t)  # uncomment for debugging
```

(continues on next page)

(continued from previous page)

```
    rtc.datetime = t
    print()
# pylint: enable-msg=using-constant-test

# Main loop:
while True:
    t = rtc.datetime
    # print(t)     # uncomment for debugging
    print(
        "The date is {} {}/{}/{}".format(
            days[int(t.tm_wday)], t.tm_mday, t.tm_mon, t.tm_year
        )
    )
    print("The time is {}:{:02}:{:02}".format(t.tm_hour, t.tm_min, t.tm_sec))
    time.sleep(1)  # wait a second
```

7.2 `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

7.2.1 Implementation Notes

Hardware:

- Adafruit DS1307 RTC breakout (Product ID: 3296)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://circuitpython.org/downloads
- 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.

Parameters **`i2c_bus`** (*`I2C`*) – The I2C bus the device is connected to

Quickstart: Importing and using the device

Here is an example of using the *`DS1307`* class. First you will need to import the libraries to use the sensor

```
import time
import board
import adafruit_ds1307
```

Once this is done you can define your `board.I2C` object and define your sensor object

```
i2c = board.I2C()  # uses board.SCL and board.SDA
rtc = adafruit_ds1307.DS1307(i2c)
```

Now you can give the current time to the device.

```
t = time.struct_time((2017, 10, 29, 15, 14, 15, 0, -1, -1))
rtc.datetime = t
```

You can access the current time accessing the `datetime` attribute.

```
current_time = rtc.datetime
```

`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 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index