Adafruit's PCF8523 RTC Library Documentation

Release 1.0

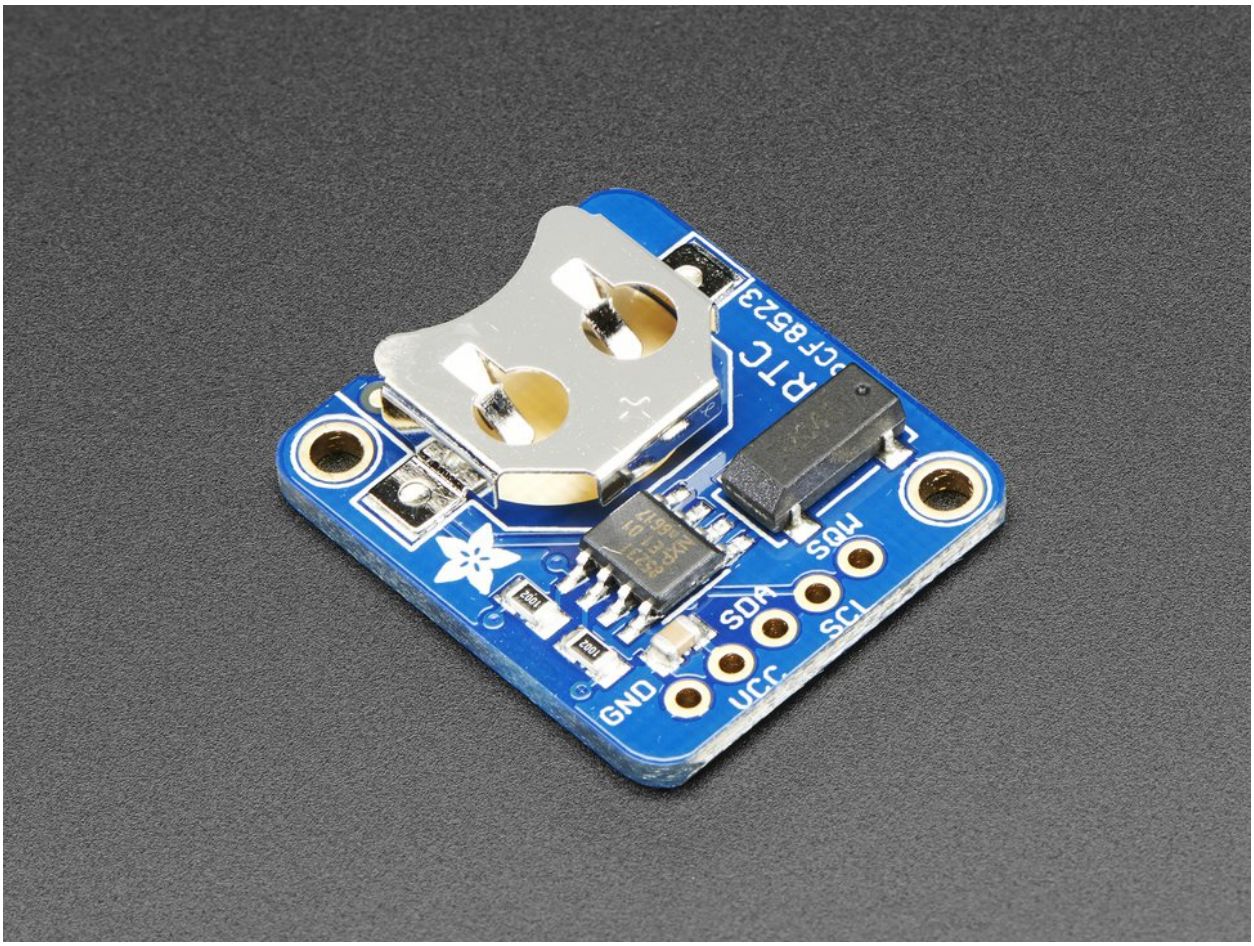
Philip Moyer

Jan 30, 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. Equipped with PCF8523 RTC - it can run from 3.3V or 5V power & logic!

The PCF8523 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 need a DS1307 for compatibility reasons, check out our DS1307 RTC breakout.



CHAPTER 1

Dependencies

This driver depends on the Register and Bus Device libraries. Please ensure they are also available on the CircuitPython filesystem. This is easily achieved by downloading a library and driver bundle.

CHAPTER 2

Usage Notes

2.1 Basics

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

```
import busio
import adafruit_pcf8523
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_pcf8523.PCF8523(myI2C)
```

2.2 Date and time

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)
```

2.3 Alarm

To set the time, you need to set `alarm` to a tuple with a `time.struct_time` object and string representing the frequency such as "hourly":

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

After the RTC is set, you retrieve the alarm status by reading the *`alarm_status`* attribute. Once True, set it back to False to reset.

```
if rtc.alarm_status:
    print("wake up!")
    rtc.alarm_status = False
```

CHAPTER 3

API Reference

3.1 adafruit_pcf8523 - PCF8523 Real Time Clock module

This library supports the use of the PCF8523-based RTC in CircuitPython. It contains a base RTC class used by all Adafruit RTC libraries. This base class is inherited by the chip-specific subclasses.

Functions are included for reading and writing registers and manipulating datetime objects.

Author(s): Philip R. Moyer and Radomir Dopieralski for Adafruit Industries. Date: November 2016 Affiliation: Adafruit Industries

3.1.1 Implementation Notes

Hardware:

- Adafruit Adalogger FeatherWing - RTC + SD Add-on (Product ID: 2922)
- Adafruit PCF8523 RTC breakout (Product ID: 3295)

Software and Dependencies:

- Adafruit CircuitPython firmware for the ESP8622 and M0-based boards: https://github.com/adafruit/micropython/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. Datasheet: http://cache.nxp.com/documents/data_sheet/PCF8523.pdf

class adafruit_pcf8523.**PCF8523**(*i2c*)
Interface to the PCF8523 RTC.

alarm
Alarm time for the first alarm.

alarm_interrupt
True if the interrupt pin will output when alarm is alarming.

alarm_status
True if alarm is alarming. Set to False to reset.

battery_low
True if the battery is low and should be replaced.

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

datetime_register
Current date and time.

lost_power
True if the device has lost power since the time was set.

power_management
Power management state that dictates battery switchover, power sources and low battery detection. Defaults to BATTERY_SWITCHOVER_OFF (0b000).

Python Module Index

a

Index