

---

# **Adafruit PCA9685 Library Documentation**

***Release 1.0***

**Radomir Dopieralski**

**Mar 04, 2019**



---

## Contents

---

<b>1</b>	<b>Dependencies</b>	<b>3</b>
<b>2</b>	<b>Usage Example</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_pca9685 . . . . .	13
5.2.1	Implementation Notes . . . . .	14
<b>6</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>



Driver for the PCA9685, a 16-channel, 12-bit PWM chip



# 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 Example

---

See `examples/pca9685_simpletest.py` for a demo of the usage.



## 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-pca9685 --
↳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/pca9685\_simpletest.py

```
1  # This simple test outputs a 50% duty cycle PWM single on the 0th channel. Connect an_
   ↳ LED and
2  # resistor in series to the pin to visualize duty cycle changes and its impact on_
   ↳ brightness.
3
4  from board import SCL, SDA
5  import busio
6
7  # Import the PCA9685 module.
8  from adafruit_pca9685 import PCA9685
9
10 # Create the I2C bus interface.
11 i2c_bus = busio.I2C(SCL, SDA)
12
13 # Create a simple PCA9685 class instance.
14 pca = PCA9685(i2c_bus)
15
16 # Set the PWM frequency to 60hz.
17 pca.frequency = 60
18
19 # Set the PWM duty cycle for channel zero to 50%. duty_cycle is 16 bits to match_
   ↳ other PWM objects
20 # but the PCA9685 will only actually give 12 bits of resolution.
21 pca.channels[0].duty_cycle = 0x7fff
```

Listing 2: examples/calibration.py

```

1  # This advanced example can be used to compute a more precise reference_clock_speed.
   ↳ Use an
2  # oscilloscope or logic analyzer to measure the signal frequency and type the results.
   ↳ into the
3  # prompts. At the end it'll give you a more precise value around 25 mhz for your
   ↳ reference clock
4  # speed.
5
6  import time
7
8  from board import SCL, SDA
9  import busio
10
11 # Import the PCA9685 module.
12 from adafruit_pca9685 import PCA9685
13
14 # Create the I2C bus interface.
15 i2c_bus = busio.I2C(SCL, SDA)
16
17 # Create a simple PCA9685 class instance.
18 pca = PCA9685(i2c_bus)
19
20 # Set the PWM frequency to 100hz.
21 pca.frequency = 100
22
23 input("Press enter when ready to measure default frequency.")
24
25 # Set the PWM duty cycle for channel zero to 50%. duty_cycle is 16 bits to match
   ↳ other PWM objects
26 # but the PCA9685 will only actually give 12 bits of resolution.
27 print("Running with default calibration")
28 pca.channels[0].duty_cycle = 0x7fff
29 time.sleep(1)
30 pca.channels[0].duty_cycle = 0
31
32 measured_frequency = float(input("Frequency measured: "))
33 print()
34
35 pca.reference_clock_speed = pca.reference_clock_speed * (measured_frequency / pca.
   ↳ frequency)
36 # Set frequency again so we can get closer. Reading it back will produce the real
   ↳ value.
37 pca.frequency = 100
38
39 input("Press enter when ready to measure coarse calibration frequency.")
40 pca.channels[0].duty_cycle = 0x7fff
41 time.sleep(1)
42 pca.channels[0].duty_cycle = 0
43 measured_after_calibration = float(input("Frequency measured: "))
44 print()
45
46 reference_clock_speed = measured_after_calibration * 4096 * pca.prescale_reg
47
48 print("Real reference clock speed: {0:.0f}".format(reference_clock_speed))

```



Listing 3: examples/servo.py

```

1  # This example moves a servo its full range (180 degrees by default) and then back.
2
3  from board import SCL, SDA
4  import busio
5
6  # Import the PCA9685 module.
7  from adafruit_pca9685 import PCA9685
8
9  # This example also relies on the Adafruit motor library available here:
10 # https://github.com/adafruit/Adafruit_CircuitPython_Motor
11 from adafruit_motor import servo
12
13 i2c = busio.I2C(SCL, SDA)
14
15 # Create a simple PCA9685 class instance.
16 pca = PCA9685(i2c)
17 pca.frequency = 50
18
19 # To get the full range of the servo you will likely need to adjust the min_pulse and
20 # ↪ max_pulse to
21 # match the stall points of the servo.
22 # This is an example for the Sub-micro servo: https://www.adafruit.com/product/2201
23 # servo7 = servo.Servo(pca.channels[7], min_pulse=580, max_pulse=2480)
24 # This is an example for the Micro Servo - High Powered, High Torque Metal Gear:
25 # https://www.adafruit.com/product/2307
26 # servo7 = servo.Servo(pca.channels[7], min_pulse=600, max_pulse=2400)
27 # This is an example for the Standard servo - TowerPro SG-5010 - 5010:
28 # https://www.adafruit.com/product/155
29 # servo7 = servo.Servo(pca.channels[7], min_pulse=600, max_pulse=2500)
30 # This is an example for the Analog Feedback Servo: https://www.adafruit.com/product/
31 # ↪ 1404
32 # servo7 = servo.Servo(pca.channels[7], min_pulse=600, max_pulse=2600)
33
34 # The pulse range is 1000 - 2000 by default.
35 servo7 = servo.Servo(pca.channels[7])
36
37 for i in range(180):
38     servo7.angle = i
39 for i in range(180):
40     servo7.angle = 180 - i
41
42 pca.deinit()

```

## 5.2 adafruit\_pca9685

Driver for the PCA9685 PWM control IC. Its commonly used to control servos, leds and motors.

### See also:

The [Adafruit CircuitPython Motor library](#) can be used to control the PWM outputs for specific uses instead of generic duty\_cycle adjustments.

- Author(s): Scott Shawcroft

### 5.2.1 Implementation Notes

#### Hardware:

- Adafruit 16-Channel 12-bit PWM/Servo Driver - I2C interface - PCA9685 (Product ID: 815)

#### Software and Dependencies:

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

**class** `adafruit_pca9685.PCA9685` (*i2c\_bus*, \*, *address*=64, *reference\_clock\_speed*=25000000)  
Initialise the PCA9685 chip at address on *i2c\_bus*.

The internal reference clock is 25mhz but may vary slightly with environmental conditions and manufacturing variances. Providing a more precise *reference\_clock\_speed* can improve the accuracy of the frequency and duty\_cycle computations. See the `calibration.py` example for how to derive this value by measuring the resulting pulse widths.

#### Parameters

- ***i2c\_bus*** (*I2C*) – The I2C bus which the PCA9685 is connected to.
- ***address*** (*int*) – The I2C address of the PCA9685.
- ***reference\_clock\_speed*** (*int*) – The frequency of the internal reference clock in Hertz.

**channels** = `None`

Sequence of 16 *PWMChannel* objects. One for each channel.

**deinit** ()

Stop using the pca9685.

**frequency**

The overall PWM frequency in Hertz.

**reference\_clock\_speed** = `None`

The reference clock speed in Hz.

**reset** ()

Reset the chip.

**class** `adafruit_pca9685.PCAChannels` (*pca*)

Lazily creates and caches channel objects as needed. Treat it like a sequence.

**class** `adafruit_pca9685.PWMChannel` (*pca*, *index*)

A single PCA9685 channel that matches the *PWMOut* API.

**duty\_cycle**

16 bit value that dictates how much of one cycle is high (1) versus low (0). 0xffff will always be high, 0 will always be low and 0x7fff will be half high and then half low.

**frequency**

The overall PWM frequency in Hertz (read-only). A PWMChannel's frequency cannot be set individually. All channels share a common frequency, set by PCA9685.frequency.

## CHAPTER 6

---

### Indices and tables

---

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



### a

adafruit\_pca9685, [13](#)



### A

`adafruit_pca9685` (module), [13](#)

### C

`channels` (`adafruit_pca9685.PCA9685` attribute), [14](#)

### D

`deinit()` (`adafruit_pca9685.PCA9685` method), [14](#)

`duty_cycle` (`adafruit_pca9685.PWMChannel` attribute),  
[14](#)

### F

`frequency` (`adafruit_pca9685.PCA9685` attribute), [14](#)

`frequency` (`adafruit_pca9685.PWMChannel` attribute), [14](#)

### P

`PCA9685` (class in `adafruit_pca9685`), [14](#)

`PCAChannels` (class in `adafruit_pca9685`), [14](#)

`PWMChannel` (class in `adafruit_pca9685`), [14](#)

### R

`reference_clock_speed` (`adafruit_pca9685.PCA9685` attribute), [14](#)

`reset()` (`adafruit_pca9685.PCA9685` method), [14](#)