

---

# **AdafruitMCP230xx Library Documentation**

***Release 1.0***

**Tony DiCola**

**Dec 21, 2018**



---

## 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	Zip release files . . . . .	9
4.2	Sphinx documentation . . . . .	9
<b>5</b>	<b>Table of Contents</b>	<b>11</b>
5.1	Simple test . . . . .	11
5.2	adafruit_mcp230xx . . . . .	12
<b>6</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>



CircuitPython module for the MCP23017 and MCP23008 I2C I/O extenders.



# CHAPTER 1

---

## Dependencies

---

This driver depends on:

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

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/mcp230xx_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.



### 4.1 Zip release files

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-mcp230xx --
↳library_location .
```

### 4.2 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/mcp230xx\_simpletest.py

```
1  # Simple demo of reading and writing the digital I/O of the MCP2300xx as if
2  # they were native CircuitPython digital inputs/outputs.
3  # Author: Tony DiCola
4  import time
5
6  import board
7  import busio
8  import digitalio
9
10 import adafruit_mcp230xx
11
12
13 # Initialize the I2C bus:
14 i2c = busio.I2C(board.SCL, board.SDA)
15
16 # Create an instance of either the MCP23008 or MCP23017 class depending on
17 # which chip you're using:
18 mcp = adafruit_mcp230xx.MCP23008(i2c) # MCP23008
19 #mcp = adafruit_mcp230xx.MCP23017(i2c) # MCP23017
20
21 # Optionally change the address of the device if you set any of the A0, A1, A2
22 # pins. Specify the new address with a keyword parameter:
23 #mcp = adafruit_mcp230xx.MCP23017(i2c, address=0x21) # MCP23017 w/ A0 set
24
25 # Now call the get_pin function to get an instance of a pin on the chip.
26 # This instance will act just like a digitalio.DigitalInOut class instance
27 # and has all the same properties and methods (except you can't set pull-down
```

(continues on next page)

(continued from previous page)

```

28 # resistors, only pull-up!). For the MCP23008 you specify a pin number from 0
29 # to 7 for the GP0...GP7 pins. For the MCP23017 you specify a pin number from
30 # 0 to 15 for the GPIOA0...GPIOA7, GPIOB0...GPIOB7 pins (i.e. pin 12 is GPIOB4).
31 pin0 = mcp.get_pin(0)
32 pin1 = mcp.get_pin(1)
33
34 # Setup pin0 as an output that's at a high logic level.
35 pin0.switch_to_output(value=True)
36
37 # Setup pin1 as an input with a pull-up resistor enabled. Notice you can also
38 # use properties to change this state.
39 pin1.direction = digitalio.Direction.INPUT
40 pin1.pull = digitalio.Pull.UP
41
42 # Now loop blinking the pin 0 output and reading the state of pin 1 input.
43 while True:
44     # Blink pin 0 on and then off.
45     pin0.value = True
46     time.sleep(0.5)
47     pin0.value = False
48     time.sleep(0.5)
49     # Read pin 1 and print its state.
50     print('Pin 1 is at a high level: {0}'.format(pin1.value))

```

## 5.2 adafruit\_mcp230xx

CircuitPython module for the MCP23017 and MCP23008 I2C I/O extenders.

- Author(s): Tony DiCola

**class** adafruit\_mcp230xx.**DigitalInOut** (pin\_number, mcp230xx)

Digital input/output of the MCP230xx. The interface is exactly the same as the digitalio.DigitalInOut class (however the MCP230xx does not support pull-down resistors and an exception will be thrown attempting to set one).

**direction**

The direction of the pin, either True for an input or False for an output.

**pull**

Enable or disable internal pull-up resistors for this pin. A value of digitalio.Pull.UP will enable a pull-up resistor, and None will disable it. Pull-down resistors are NOT supported!

**switch\_to\_input** (pull=None, \*\*kwargs)

Switch the pin state to a digital input with the provided starting pull-up resistor state (optional, no pull-up by default). Note that pull-down resistors are NOT supported!

**switch\_to\_output** (value=False, \*\*kwargs)

Switch the pin state to a digital output with the provided starting value (True/False for high or low, default is False/low).

**value**

The value of the pin, either True for high or False for low. Note you must configure as an output or input appropriately before reading and writing this value.

**class** adafruit\_mcp230xx.**MCP23008** (i2c, address=<sphinx.ext.autodoc.importer.\_MockObject object>)

Initialize MCP23008 instance on specified I2C bus and optionally at the specified I2C address.



**get\_pin** (*pin*)

Convenience function to create an instance of the DigitalInOut class pointing at the specified pin of this MCP23008 device.

**gpio**

The raw GPIO output register. Each bit represents the output value of the associated pin (0 = low, 1 = high), assuming that pin has been configured as an output previously.

**gppu**

The raw GPPU pull-up register. Each bit represents if a pull-up is enabled on the specified pin (1 = pull-up enabled, 0 = pull-up disabled). Note pull-down resistors are NOT supported!

**iodir**

The raw IODIR direction register. Each bit represents direction of a pin, either 1 for an input or 0 for an output mode.

**class** adafruit\_mcp230xx.**MCP23017** (*i2c*, *address*=<*sphinx.ext.autodoc.importer.\_MockObject object*>)

Initialize MCP23017 instance on specified I2C bus and optionally at the specified I2C address.

**get\_pin** (*pin*)

Convenience function to create an instance of the DigitalInOut class pointing at the specified pin of this MCP23017 device.

**gpio**

The raw GPIO output register. Each bit represents the output value of the associated pin (0 = low, 1 = high), assuming that pin has been configured as an output previously.

**gpioa**

The raw GPIO A output register. Each bit represents the output value of the associated pin (0 = low, 1 = high), assuming that pin has been configured as an output previously.

**gpiob**

The raw GPIO B output register. Each bit represents the output value of the associated pin (0 = low, 1 = high), assuming that pin has been configured as an output previously.

**gppu**

The raw GPPU pull-up register. Each bit represents if a pull-up is enabled on the specified pin (1 = pull-up enabled, 0 = pull-up disabled). Note pull-down resistors are NOT supported!

**gppua**

The raw GPPU A pull-up register. Each bit represents if a pull-up is enabled on the specified pin (1 = pull-up enabled, 0 = pull-up disabled). Note pull-down resistors are NOT supported!

**gppub**

The raw GPPU B pull-up register. Each bit represents if a pull-up is enabled on the specified pin (1 = pull-up enabled, 0 = pull-up disabled). Note pull-down resistors are NOT supported!

**iodir**

The raw IODIR direction register. Each bit represents direction of a pin, either 1 for an input or 0 for an output mode.

**iodira**

The raw IODIR A direction register. Each bit represents direction of a pin, either 1 for an input or 0 for an output mode.

**iodirb**

The raw IODIR B direction register. Each bit represents direction of a pin, either 1 for an input or 0 for an output mode.



## CHAPTER 6

---

### Indices and tables

---

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



### a

adafruit\_mcp230xx, [12](#)



## A

adafruit\_mcp230xx (module), [12](#)

## D

DigitalInOut (class in adafruit\_mcp230xx), [12](#)

direction (adafruit\_mcp230xx.DigitalInOut attribute), [12](#)

## G

get\_pin() (adafruit\_mcp230xx.MCP23008 method), [12](#)

get\_pin() (adafruit\_mcp230xx.MCP23017 method), [13](#)

gpio (adafruit\_mcp230xx.MCP23008 attribute), [13](#)

gpio (adafruit\_mcp230xx.MCP23017 attribute), [13](#)

gpioa (adafruit\_mcp230xx.MCP23017 attribute), [13](#)

gpiob (adafruit\_mcp230xx.MCP23017 attribute), [13](#)

gppu (adafruit\_mcp230xx.MCP23008 attribute), [13](#)

gppu (adafruit\_mcp230xx.MCP23017 attribute), [13](#)

gppua (adafruit\_mcp230xx.MCP23017 attribute), [13](#)

gppub (adafruit\_mcp230xx.MCP23017 attribute), [13](#)

## I

iodir (adafruit\_mcp230xx.MCP23008 attribute), [13](#)

iodir (adafruit\_mcp230xx.MCP23017 attribute), [13](#)

iodira (adafruit\_mcp230xx.MCP23017 attribute), [13](#)

iodirb (adafruit\_mcp230xx.MCP23017 attribute), [13](#)

## M

MCP23008 (class in adafruit\_mcp230xx), [12](#)

MCP23017 (class in adafruit\_mcp230xx), [13](#)

## P

pull (adafruit\_mcp230xx.DigitalInOut attribute), [12](#)

## S

switch\_to\_input() (adafruit\_mcp230xx.DigitalInOut method), [12](#)

switch\_to\_output() (adafruit\_mcp230xx.DigitalInOut method), [12](#)

## V

value (adafruit\_mcp230xx.DigitalInOut attribute), [12](#)