
AdafruitMCP230xx Library Documentation

Release 1.0

Tony DiCola

Feb 12, 2020

Contents

1	Dependencies	3
2	Installing from PyPI	5
3	Usage Example	7
4	Contributing	9
5	Documentation	11
6	Table of Contents	13
6.1	Simple test	13
6.2	mcp230xx	14
6.3	mcp23008	14
6.4	mcp23017	15
6.5	digital_inout	16
7	Indices and tables	19
	Python Module Index	21
	Index	23

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

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-mcp230xx
```

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

```
sudo pip3 install adafruit-circuitpython-mcp230xx
```

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-mcp230xx
```


CHAPTER 3

Usage Example

See `examples/mcp230xx_simpletest.py` for a demo of the usage.

CHAPTER 4

Contributing

Contributions are welcome! Please read our [Code of Conduct](#) before contributing to help this project stay welcoming.

CHAPTER 5

Documentation

For information on building library documentation, please check out [this guide](#).

6.1 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 from adafruit_mcp230xx.mcp23008 import MCP23008
11 #from adafruit_mcp230xx.mcp23017 import MCP23017
12
13
14 # Initialize the I2C bus:
15 i2c = busio.I2C(board.SCL, board.SDA)
16
17 # Create an instance of either the MCP23008 or MCP23017 class depending on
18 # which chip you're using:
19 mcp = MCP23008(i2c) # MCP23008
20 #mcp = MCP23017(i2c) # MCP23017
21
22 # Optionally change the address of the device if you set any of the A0, A1, A2
23 # pins. Specify the new address with a keyword parameter:
24 #mcp = MCP23017(i2c, address=0x21) # MCP23017 w/ A0 set
25
26 # Now call the get_pin function to get an instance of a pin on the chip.
27 # This instance will act just like a digitalio.DigitalInOut class instance
```

(continues on next page)

(continued from previous page)

```

28 # and has all the same properties and methods (except you can't set pull-down
29 # resistors, only pull-up!). For the MCP23008 you specify a pin number from 0
30 # to 7 for the GP0...GP7 pins. For the MCP23017 you specify a pin number from
31 # 0 to 15 for the GPIOA0...GPIOA7, GPIOB0...GPIOB7 pins (i.e. pin 12 is GPIOB4).
32 pin0 = mcp.get_pin(0)
33 pin1 = mcp.get_pin(1)
34
35 # Setup pin0 as an output that's at a high logic level.
36 pin0.switch_to_output(value=True)
37
38 # Setup pin1 as an input with a pull-up resistor enabled. Notice you can also
39 # use properties to change this state.
40 pin1.direction = digitalio.Direction.INPUT
41 pin1.pull = digitalio.Pull.UP
42
43 # Now loop blinking the pin 0 output and reading the state of pin 1 input.
44 while True:
45     # Blink pin 0 on and then off.
46     pin0.value = True
47     time.sleep(0.5)
48     pin0.value = False
49     time.sleep(0.5)
50     # Read pin 1 and print its state.
51     print('Pin 1 is at a high level: {}'.format(pin1.value))

```

6.2 mcp230xx

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

- Author(s): Tony DiCola

class adafruit_mcp230xx.mcp230xx.**MCP230XX** (*i2c, address*)
 Base class for MCP230xx devices.

6.3 mcp23008

CircuitPython module for the MCP23008 I2C I/O extenders.

- Author(s): Tony DiCola

class adafruit_mcp230xx.mcp23008.**MCP23008** (*i2c, address=<sphinx.ext.autodoc.importer._MockObject object>*)

Supports 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.

6.4 mcp23017

CircuitPython module for the MCP23017 I2C I/O extenders.

- Author(s): Tony DiCola

class adafruit_mcp230xx.mcp23017.**MCP23017** (*i2c*, *address*=<*sphinx.ext.autodoc.importer._MockObject object*>)

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

clear_inta ()

Clears port A interrupts.

clear_intb ()

Clears port B interrupts.

clear_ints ()

Clears interrupts by reading INTCAP.

default_value

The raw DEFVAL interrupt control register. The default comparison value is configured in the DEFVAL register. If enabled (via GPINTEN and INTCON) to compare against the DEFVAL register, an opposite value on the associated pin will cause an interrupt to occur.

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!

int_flag

Returns a list with the pin numbers that caused an interrupt port A → pins 0-7 port B → pins 8-15

int_flaga

Returns a list of pin numbers that caused an interrupt in port A pins: 0-7

int_flagb

Returns a list of pin numbers that caused an interrupt in port B pins: 8-15

interrupt_configuration

The raw INTCON interrupt control register. The INTCON register controls how the associated pin value is compared for the interrupt-on-change feature. If a bit is set, the corresponding I/O pin is compared against the associated bit in the DEFVAL register. If a bit value is clear, the corresponding I/O pin is compared against the previous value.

interrupt_enable

The raw GPINTEN interrupt control register. The GPINTEN register controls the interrupt-on-change feature for each pin. If a bit is set, the corresponding pin is enabled for interrupt-on-change. The DEFVAL and INTCON registers must also be configured if any pins are enabled for interrupt-on-change.

io_control

The raw IOCON configuration register. Bit 1 controls interrupt polarity (1 = active-high, 0 = active-low). Bit 2 is whether irq pin is open drain (1 = open drain, 0 = push-pull). Bit 3 is unused. Bit 4 is whether SDA slew rate is enabled (1 = yes). Bit 5 is if I2C address pointer auto-increments (1 = no). Bit 6 is whether interrupt pins are internally connected (1 = yes). Bit 7 is whether registers are all in one bank (1 = no).

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.

6.5 digital_inout

Digital input/output of the MCP230xx.

- Author(s): Tony DiCola

class adafruit_mcp230xx.digital_inout.**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.

CHAPTER 7

Indices and tables

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

a

`adafruit_mcp230xx.digital_inout`, 16
`adafruit_mcp230xx.mcp23008`, 14
`adafruit_mcp230xx.mcp23017`, 15
`adafruit_mcp230xx.mcp230xx`, 14

A

adafruit_mcp230xx.digital_inout (*module*),
16

adafruit_mcp230xx.mcp23008 (*module*), 14

adafruit_mcp230xx.mcp23017 (*module*), 15

adafruit_mcp230xx.mcp230xx (*module*), 14

C

clear_inta() (*adafruit_mcp230xx.mcp23017.MCP23017*
method), 15

clear_intb() (*adafruit_mcp230xx.mcp23017.MCP23017*
method), 15

clear_ints() (*adafruit_mcp230xx.mcp23017.MCP23017*
method), 15

D

default_value (*adafruit_mcp230xx.mcp23017.MCP23017*
attribute), 15

DigitalInOut (*class in*
adafruit_mcp230xx.digital_inout), 16

direction (*adafruit_mcp230xx.digital_inout.DigitalInOut*
attribute), 16

G

get_pin() (*adafruit_mcp230xx.mcp23008.MCP23008*
method), 14

get_pin() (*adafruit_mcp230xx.mcp23017.MCP23017*
method), 15

gpio (*adafruit_mcp230xx.mcp23008.MCP23008* *at-*
tribute), 14

gpio (*adafruit_mcp230xx.mcp23017.MCP23017* *at-*
tribute), 15

gpioa (*adafruit_mcp230xx.mcp23017.MCP23017* *at-*
tribute), 15

gpiob (*adafruit_mcp230xx.mcp23017.MCP23017* *at-*
tribute), 15

gppu (*adafruit_mcp230xx.mcp23008.MCP23008* *at-*
tribute), 14

gppu (*adafruit_mcp230xx.mcp23017.MCP23017* *at-*
tribute), 15

gppua (*adafruit_mcp230xx.mcp23017.MCP23017* *at-*
tribute), 15

gppub (*adafruit_mcp230xx.mcp23017.MCP23017* *at-*
tribute), 15

I

int_flag (*adafruit_mcp230xx.mcp23017.MCP23017*
attribute), 15

int_flaga (*adafruit_mcp230xx.mcp23017.MCP23017*
attribute), 15

int_flagb (*adafruit_mcp230xx.mcp23017.MCP23017*
attribute), 16

interrupt_configuration
(*adafruit_mcp230xx.mcp23017.MCP23017*
attribute), 16

interrupt_enable (*adafruit_mcp230xx.mcp23017.MCP23017*
attribute), 16

io_control (*adafruit_mcp230xx.mcp23017.MCP23017*
attribute), 16

iodir (*adafruit_mcp230xx.mcp23008.MCP23008* *at-*
tribute), 15

iodir (*adafruit_mcp230xx.mcp23017.MCP23017* *at-*
tribute), 16

iodira (*adafruit_mcp230xx.mcp23017.MCP23017* *at-*
tribute), 16

iodirb (*adafruit_mcp230xx.mcp23017.MCP23017* *at-*
tribute), 16

M

MCP23008 (*class in* *adafruit_mcp230xx.mcp23008*), 14

MCP23017 (*class in* *adafruit_mcp230xx.mcp23017*), 15

MCP230XX (*class in* *adafruit_mcp230xx.mcp230xx*), 14

P

pull (*adafruit_mcp230xx.digital_inout.DigitalInOut* *at-*
tribute), 16

S

`switch_to_input()`
(*adafruit_mcp230xx.digital_inout.DigitalInOut*
method), 16

`switch_to_output()`
(*adafruit_mcp230xx.digital_inout.DigitalInOut*
method), 16

V

`value` (*adafruit_mcp230xx.digital_inout.DigitalInOut*
attribute), 17