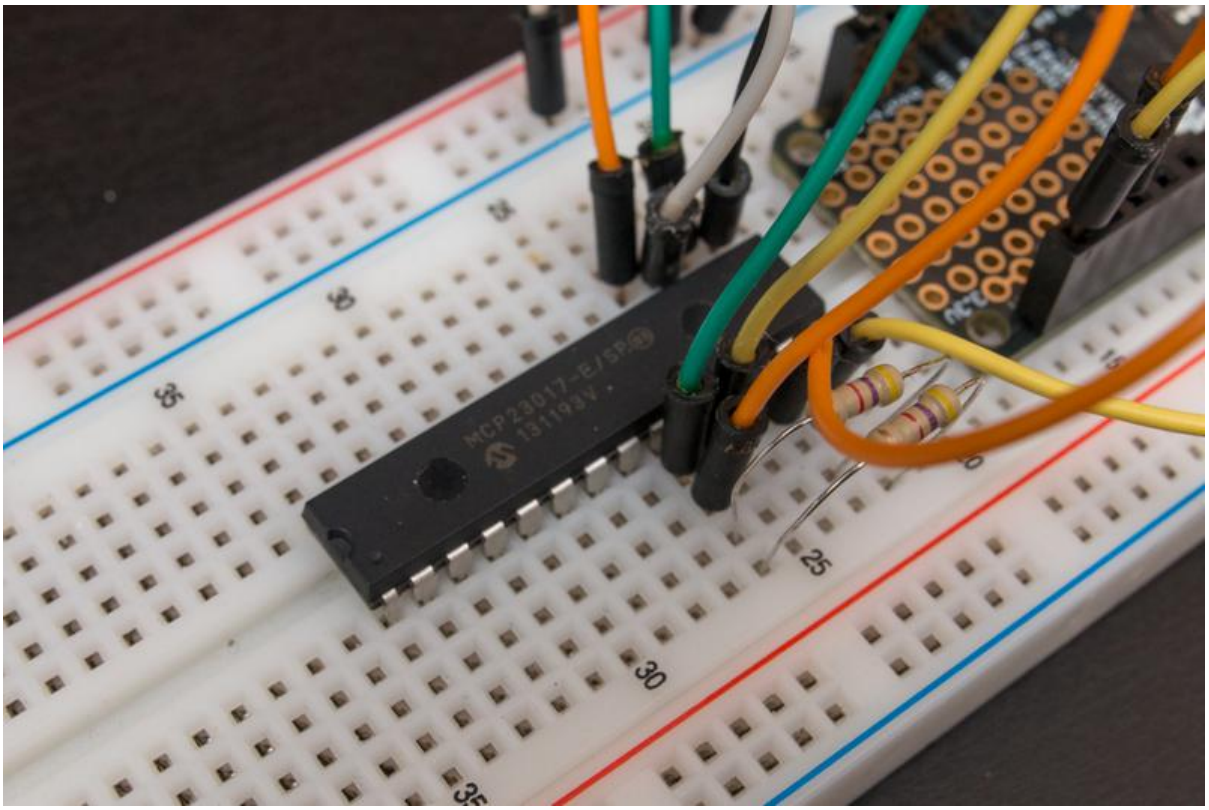
Using MCP23008 & MCP23017 with CircuitPython

Created by Tony DiCola

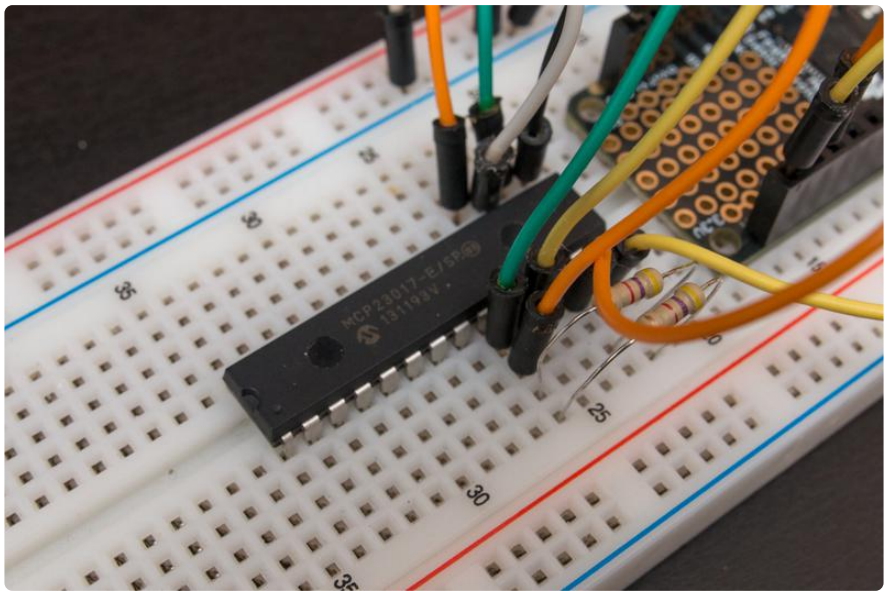


https://learn.adafruit.com/using-mcp23008-mcp23017-with-circuitpython

Last updated on 2021-11-15 07:08:58 PM EST

Table of Contents

Overview



The MCP23008 (https://adafru.it/Bog) and MCP23017 (https://adafru.it/sCR) family of chips provide an easy way to add extra digital inputs and outputs to your development board. These chips are controlled with an I2C connection and add 8 or 16 extra digital pins that can act as outputs or inputs (even with optional pull-up resistors). This is great for connecting more digital devices to your board when you run out of native digital I/O!

This guide explores how to use the MCP23008 and MCP23017 with CircuitPython. You'll learn how to connect the chip to a CircuitPython board, load an Adafruit MCP230xx module (https://adafru.it/Boh), and control the I/O pins of the chip from Python code!

Python & CircuitPython

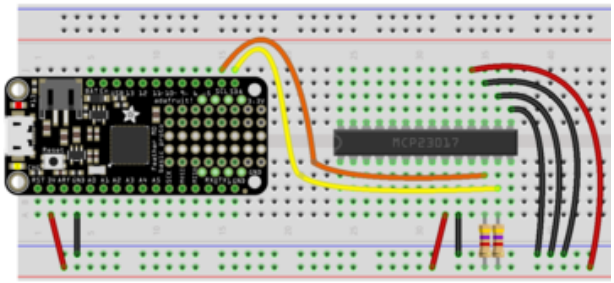
It's easy to use an MCP23xx I/O extender with Python or CircuitPython and the Adafruit CircuitPython MCP23xx (https://adafru.it/CMr) module. This module allows you to easily write Python code to add extra digital inputs and outputs.

You can use this I/O expander with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN).

CircuitPython Microcontroller Wiring

Connect your MCP230xx to your CircuitPython board using a standard I2C connection.

Here's an example of wiring a MCP23017 to a Feather M0 board:

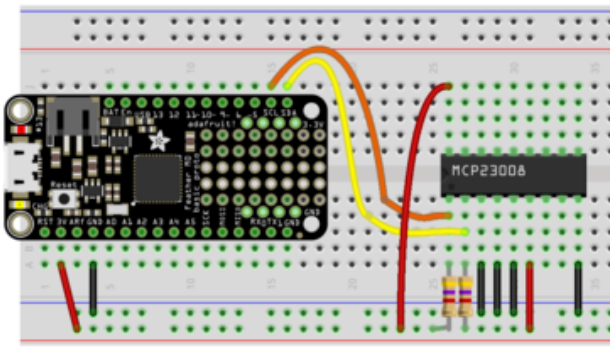


Remember you need to explicitly add pull-up resistors to the I2C SCL and SDA connections as shown above!

- Board 3.3V output to MCP23017 Vdd
- Board ground/GND to MCP23017 Vss
- Board SCL to MCP23017 SCL
- Board SDA to MCP23017 SDA
- MCP23017 SCL to 4.7 KΩ resistor connected to 3.3V
- MCP23017 SDA to 4.7 KΩ resistor connected to 3.3V
- MCP23017 A0 to ground
- MCP23017 A1 to ground
- MCP23017 A2 to ground
- MCP23017 reset to 3.3V

When in doubt consult the MCP23017 datasheet (https://adafru.it/BHq) for exact details on its pins and wiring.

Here's an example of wiring a MCP23008 to a Feather M0 board:



Remember you need to explicitly add pull-up resistors to the I2C SCL and SDA connections as shown above!

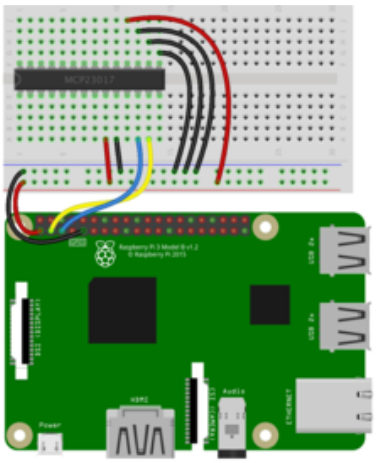
- Board 3.3V output to MCP23008 Vdd
- Board ground/GND to MCP23008 Vss
- Board SCL to MCP23008 SCL
- Board SDA to MCP23008 SDA
- MCP23008 SCL to 4.7 KΩ resistor connected to 3.3V
- MCP23008 SDA to 4.7 KΩ resistor connected to 3.3V
- MCP23008 A0 to ground
- MCP23008 A1 to ground
- MCP23008 A2 to ground
- MCP23008 reset to 3.3V

When in doubt consult the MCP23008 datasheet (https://adafru.it/aRp) for exact details on its pins and wiring.

Python Computer Wiring

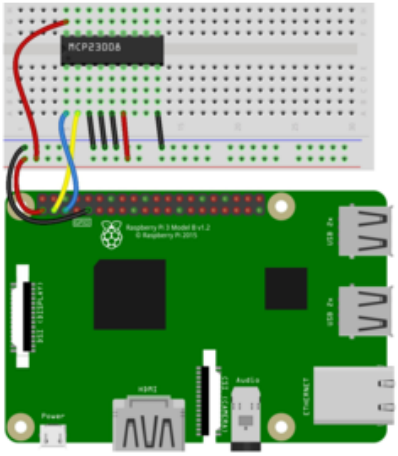
Since there's dozens of Linux computers/boards you can use we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

Here's the Raspberry Pi wired to the MCP23017 with I2C:



- Pi 3V3 output to MCP23017 Vdd
- Pi GND to MCP23017 Vss
- Pi SCL to MCP23017 SCL
- Pi SDA to MCP23017 SDA
- Pi GND to MCP23017 A0
- Pi GND to MCP23017 A1
- Pi GND to MCP23017 A2
- Pi 3V3 to MCP23017 reset

Here's the Raspberry Pi wired to the MCP23008 with I2C:



- Pi 3V3 to MCP23008 Vdd
- Pi GND to MCP23008 Vss
- Pi SCL to MCP23008 SCL
- Pi SDA to MCP23008 SDA
- Pi GND to MCP23008 A0
- Pi GND to MCP23008 A1
- Pi GND to MCP23008 A2
- Pi 3V3 to MCP23008 reset

CircuitPython Installation of MCP230xx Library

You'll need to install the Adafruit CircuitPython MCP230xx (https://adafru.it/Boh) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Amd) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/uap). Our CircuitPython starter guide has a great page on how to install the library bundle (https://adafru.it/ABU).

For non-express boards like the Trinket M0 or Gemma M0, you'll need to manually install the necessary libraries from the bundle:

- adafruit_mcp230xx.mpy
- adafruit_bus_device

Before continuing make sure your board's lib folder or root filesystem has the adafruit_mcp230xx.mpy, and adafruit_bus_device files and folders copied over.

Next connect to the board's serial REPL (https://adafru.it/Awz) so you are at the CircuitPython >>> prompt.

Python Installation of MCP230xx Library

You'll need to install the Adafruit_Blinka library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready (https://adafru.it/BSN)!

Once that's done, from your command line run the following command:

- `sudo pip3 install adafruit-circuitpython-mcp230xx`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

CircuitPython & Python Usage

To demonstrate the usage of the device we'll initialize it and control the chip's digital I/O lines from the Python REPL.

Run the following code to import the necessary modules and initialize the I2C connection. Note that the import syntax will change depending on which MCP230xx chip you have. Here we show MCP23008:

```
import board
import busio
from digitalio import Direction
from adafruit_mcp230xx.mcp23008 import MCP23008
i2c = busio.I2C(board.SCL, board.SDA)
```

For a MCP23017, the import would look like this:

```
from adafruit_mcp230xx.mcp23017 import MCP23017
```

Then create an instance of either the MCP23017 or MCP23008 class depending on which chip you're using. For example the MCP23017:

```
mcp = MCP23017(i2c)
```

Or the MCP23008:

```
mcp = MCP23008(i2c)
```

By default the chip's `0x20` I2C address will be assumed (with A0, A1, A2 all grounded). But either class initializer can also be passed an optional address keyword argument to override the I2C address, if you set any of the A0, A1, or A2 pins as described in the datasheet. For example:

```
mcp = MCP23008(i2c, address=0x24)
```

Next you're ready to control the GPIO ports of the chip. This is as easy as using a DigitalInOut instance (https://adafru.it/C4c) in CircuitPython, the MCP230xx library makes each chip pin look like CircuitPython DigitalInOut class. You just need to call the get_pin function to retrieve an instance of the chip's DigitalInOut class.

For example to create GPIO0 (or GPIOA0 on the MCP23017) as a digital output:

```
pin0 = mcp.get_pin(0)
pin0.direction = Direction.OUTPUT
```

Then toggle the pin high or low by controlling its value property, just like using the DigitalInOut class. Try connecting the cathode or anode of a LED to the output to see it turn on or off (or use a multimeter to measure the output voltage).

```
pin0.value = True  # GPIO0 / GPIOA0 to high logic level
pin0.value = False # GPIO0 / GPIOA0 to low logic level
```

You can also set a pin as an input, again using the same functions and properties as the DigitalInOut class. For example here's how to create GPIO1 / GPIOA1 as a digital input with a pull-up resistor enabled:

```
import digitalio
pin1 = mcp.get_pin(1)
pin1.direction = digitalio.Direction.INPUT
pin1.pull = digitalio.Pull.UP
```

Note that pull-down resistors are not supported by the chip. If you need them you'll have to add external resistors yourself!

Again read the state of the input with the value property like a DigitalInOut instance. Notice since the pull-up resistor is enabled if the pin is not connected it will read True / high logic level. Connect the pin to ground and you'll see it read False / low logic level.

```
pin1.value
pin1.value
```

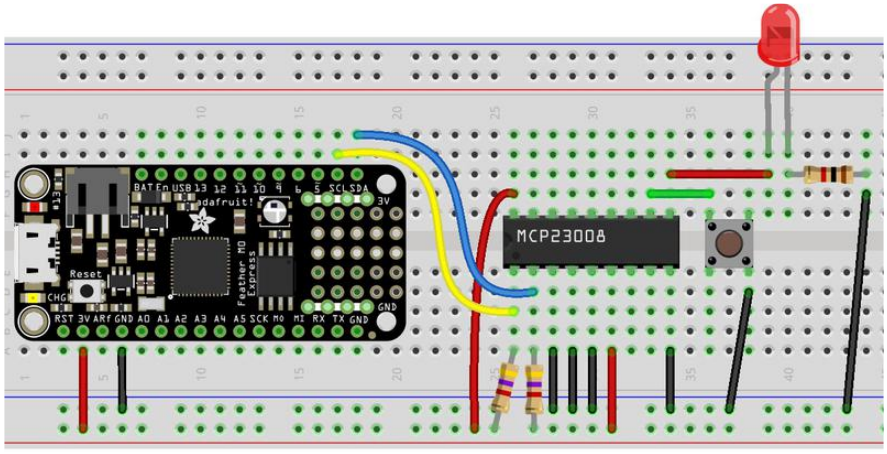
```
>>> pin12 = mcp.get_pin(12)
>>> pin12.switch_to_input()
>>> import digitalio
>>> pin12.pull = digitalio.Pull.UP
>>> pin12.value
True
>>> pin12.value
True
>>> pin12.value
True
>>> pin12.value
False
>>> pin12.value
False
```

For the MCP23008 you can get a pin instance for any pin numbered 0 to 7. These correspond to the GPIO0 to GPIO7 pins of the chip.

For the MCP23017 you can get a pin instance for any pin numbered 0 to 15. These correspond to the GPIOA0 to GPIOA7, then GPIOB0 to GPIOB7 pins. For example pin 12 corresponds to GPIOB4 on the MCP23017.

That's all there is to using the MCP230xx I2C I/O extender with CircuitPython!

Below is a complete example that will blink pin 0 and read the state of pin 1 with a pull-up resistor enabled. Add a button and an LED to your setup, like in the diagram below. Save the program as code.py on your board, then open the serial REPL to see the output.



Full Example Code

```
# SPDX-FileCopyrightText: 2017 Tony DiCola for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# Simple demo of reading and writing the digital I/O of the MCP2300xx as if
# they were native CircuitPython digital inputs/outputs.
# Author: Tony DiCola
import time

import board
import busio
import digitalio

from adafruit_mcp230xx.mcp23008 import MCP23008

# from adafruit_mcp230xx.mcp23017 import MCP23017


# Initialize the I2C bus:
i2c = busio.I2C(board.SCL, board.SDA)

# Create an instance of either the MCP23008 or MCP23017 class depending on
# which chip you're using:
mcp = MCP23008(i2c)  # MCP23008
# mcp = MCP23017(i2c)  # MCP23017

# Optionally change the address of the device if you set any of the A0, A1, A2
# pins.  Specify the new address with a keyword parameter:
# mcp = MCP23017(i2c, address=0x21)  # MCP23017 w/ A0 set

# Now call the get_pin function to get an instance of a pin on the chip.
# This instance will act just like a digitalio.DigitalInOut class instance
# and has all the same properties and methods (except you can't set pull-down
# resistors, only pull-up!).  For the MCP23008 you specify a pin number from 0
# to 7 for the GP0...GP7 pins.  For the MCP23017 you specify a pin number from
# 0 to 15 for the GPIOA0...GPIOA7, GPIOB0...GPIOB7 pins (i.e. pin 12 is GPIOB4).
pin0 = mcp.get_pin(0)
pin1 = mcp.get_pin(1)

# Setup pin0 as an output that's at a high logic level.
pin0.switch_to_output(value=True)

# Setup pin1 as an input with a pull-up resistor enabled.  Notice you can also
# use properties to change this state.
pin1.direction = digitalio.Direction.INPUT
pin1.pull = digitalio.Pull.UP

# Now loop blinking the pin 0 output and reading the state of pin 1 input.
while True:
    # Blink pin 0 on and then off.
    pin0.value = True
    time.sleep(0.5)
    pin0.value = False
    time.sleep(0.5)
    # Read pin 1 and print its state.
    print("Pin 1 is at a high level: {0}".format(pin1.value))
```

Python Docs

Python Docs (https://adafru.it/CMs)