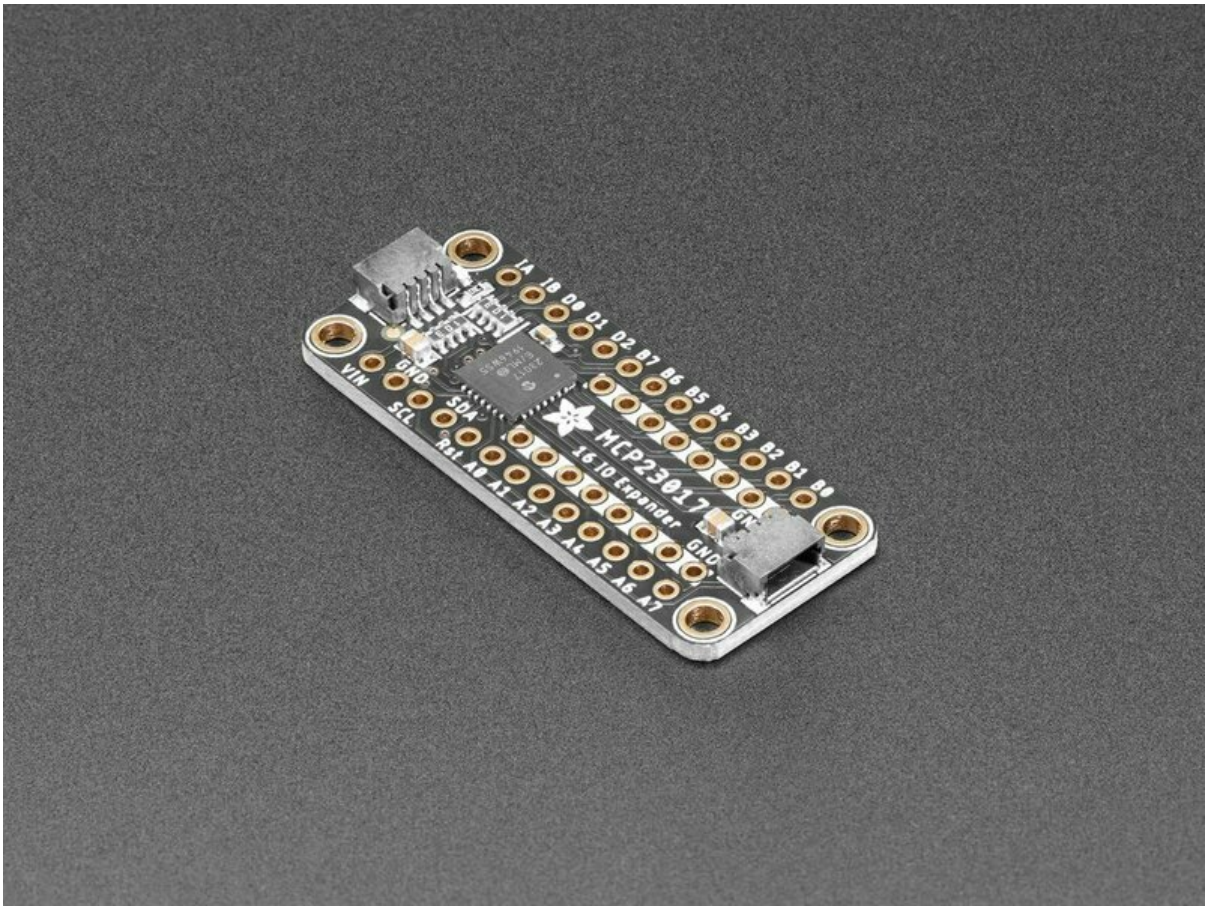
Adafruit MCP23017 I2C GPIO Expander

Created by Liz Clark

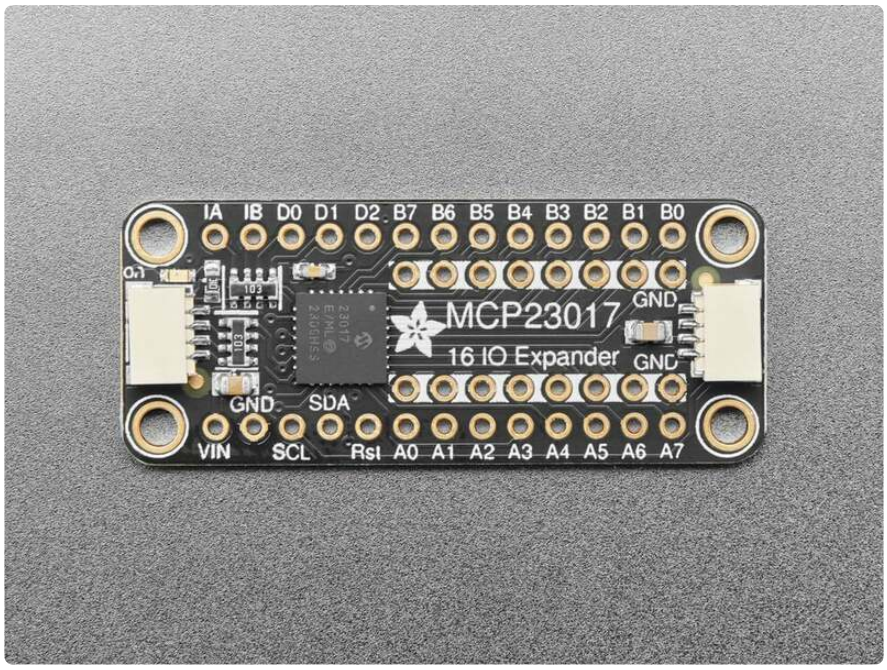


https://learn.adafruit.com/adafruit-mcp23017-i2c-gpio-expander

Last updated on 2024-06-03 03:34:52 PM EDT

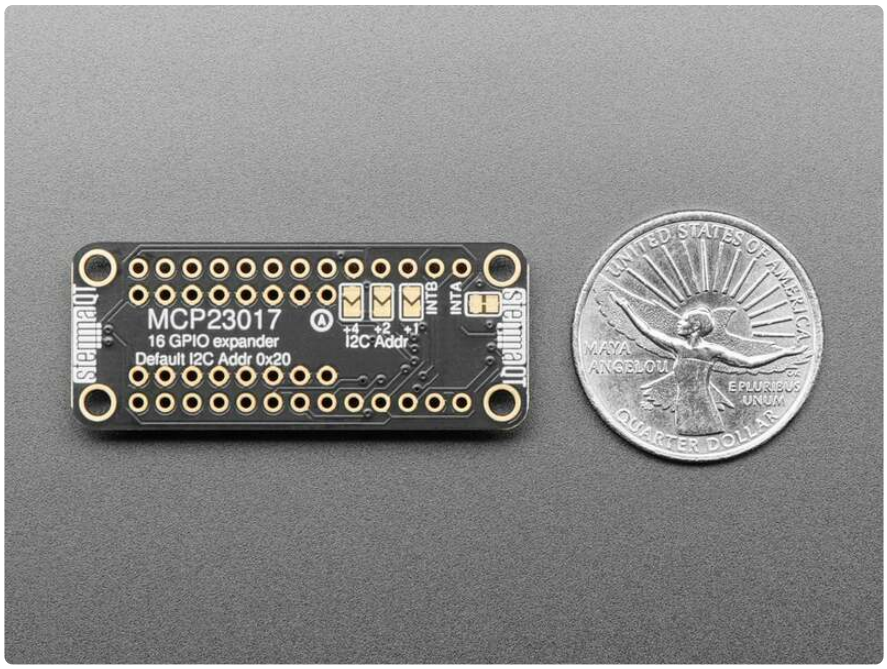
Table of Contents

Overview



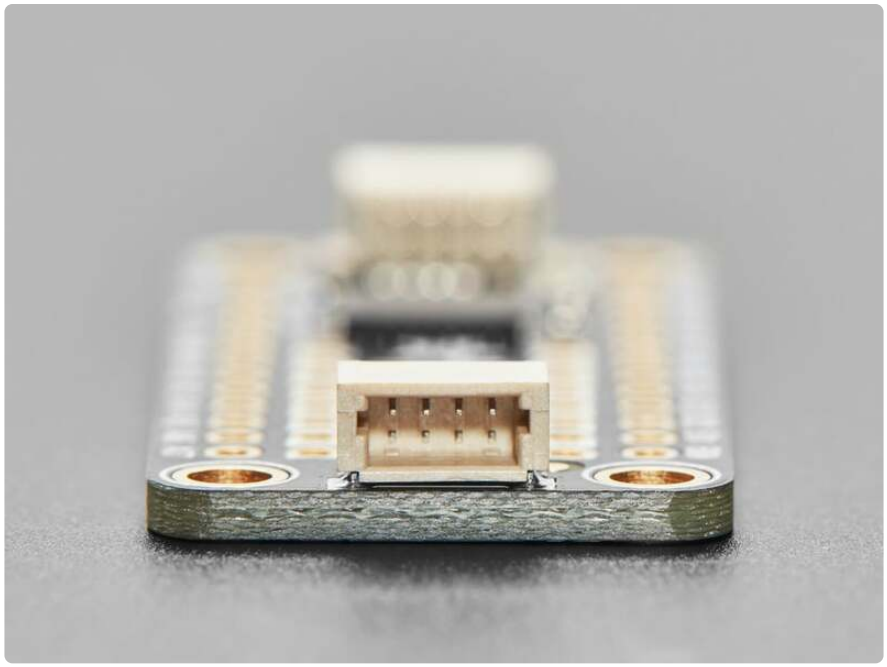
We've gotten a lot of requests for a MCP23017 breakout and we've always sorta been like "ehh why not just use the DIP chip? (http://adafru.it/732)" but with STEMMA QT we could see the use case for a plug and play version that comes with all the passives on board. This **Adafruit MCP23017 I2C GPIO Expander Breakout** has 16 GPIO with matching ground pad.

We particularly like the '17 as an expander for it's simple no-nonsense capability. It runs happily from 3V or 5V logic and power. Each GPIO can be an output driving up to 25mA, so LEDs are no problem. Or, each can be an input, with optional pullup. There are two IRQ pins that are configurable for what inputs to keep track of so no I2C bus polling is required. With 3 address pins, you can have up to 8 on a single bus for a total of 8 x 16 = 128 GPIO all on one I2C bus!



We've got solid Arduino (https://adafru.it/jFN) and CircuitPython libraries (https://adafru.it/Lkc) with examples, all ready for this chip. But even if you are using some other platform, the MCP23017 is so classic, you'll likely be able to find example code.

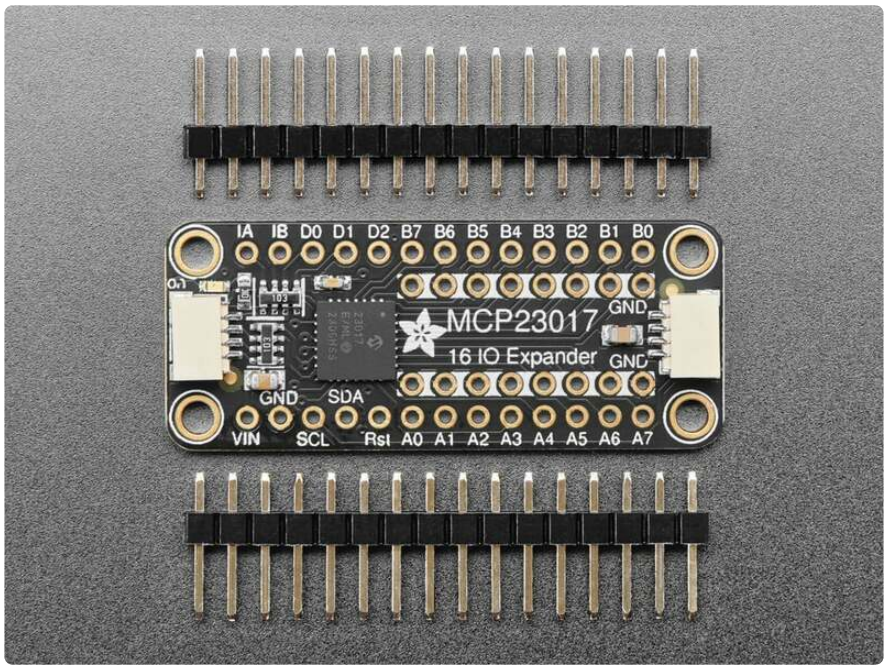
Comes with two sticks of header so you can use it in a breadboard, with some soldering. You can also free-wire buttons by connecting one side to the GPIO (set as input with pullup) and the other side to a ground pad.



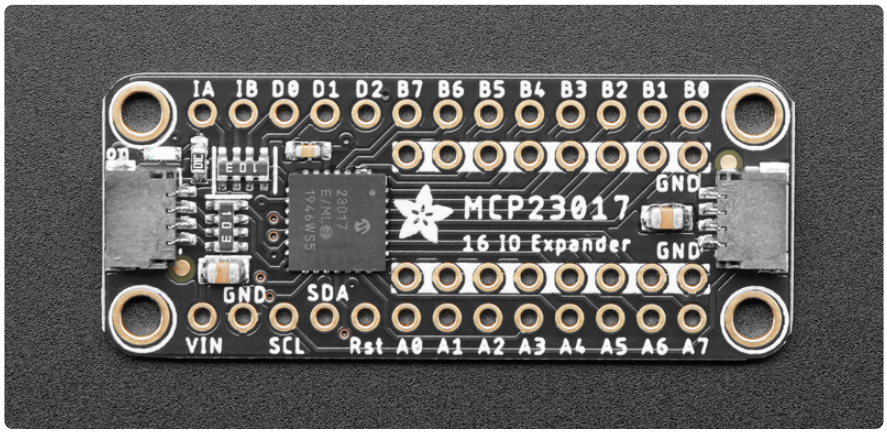
To get you going fast, we spun up a custom-made PCB in the **STEMMA QT** form factor (https://adafru.it/LBQ), making it easy to interface with. The STEMMA QT connectors (https://adafru.it/JqB) on either side are compatible with the SparkFun Qwiic (https://adafru.it/Fpw) I2C connectors. This allows you to make solderless

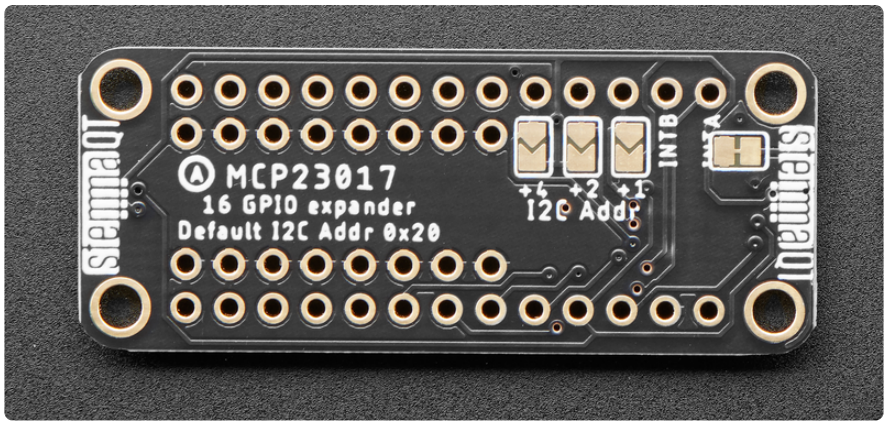
connections between your development board and the MCP23017 or to chain it with a wide range of other sensors and accessories using a **compatible cable** (https://adafru.it/JnB).

QT Cable is not included, but we have a variety in the shop (https://adafru.it/17VE).



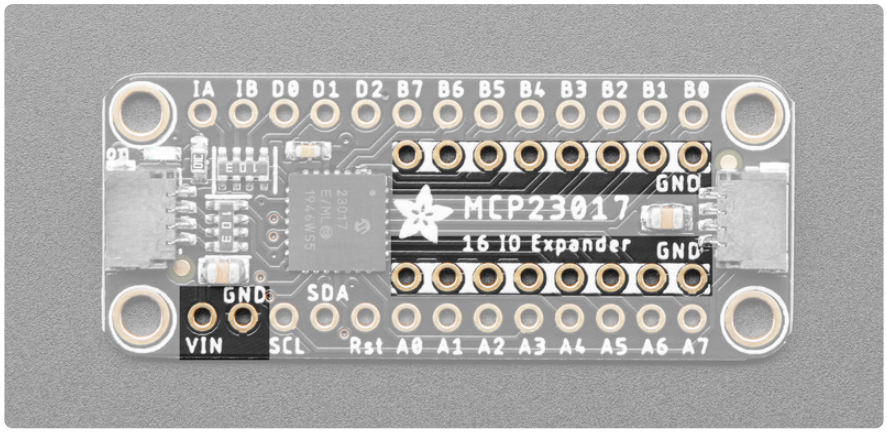
Pinouts





The default I2C address for this board is **0x20**.

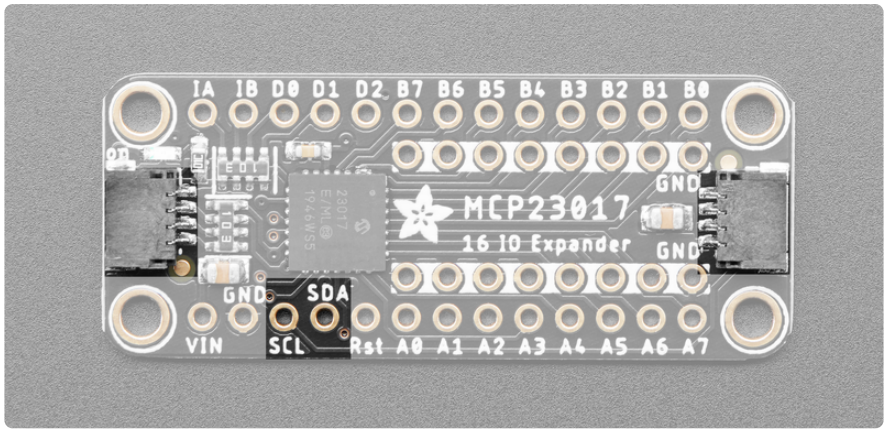
Power Pins



This breakout works with both 3V and 5V power and logic, so it can be easily used with most microcontrollers from an Arduino to a Feather or something else.

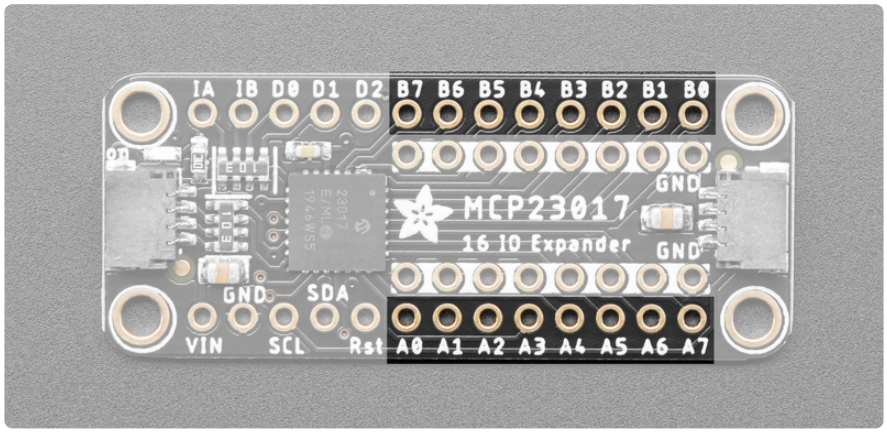
- **VIN** - This is the power pin. To power the board, give it the same power as the logic level of your microcontroller - i.e. for a 5V micro like Arduino, use 5V, or for a 3V micro like a Feather, use 3V.
- **GND** - This is common ground for power and logic.
- **GND pads** - The ground pads (highlighted in white on the board's silk) are available as discrete ground connections for the GPIO.

I2C Logic Pins



- **SCL** - I2C clock pin, connect to your microcontroller's I2C clock line. This pin is level shifted so you can use 3-5V logic, and there's a **10K pullup** on this pin.
- **SDA** - I2C data pin, connect to your microcontroller's I2C data line. This pin is level shifted so you can use 3-5V logic, and there's a **10K pullup** on this pin.
- **STEMMA QT** (https://adafru.it/Ft4) **-** These connectors allow you to connect to development boards with **STEMMA QT** connectors, or to other things, with various associated accessories (https://adafru.it/JRA).

GPIO Pins

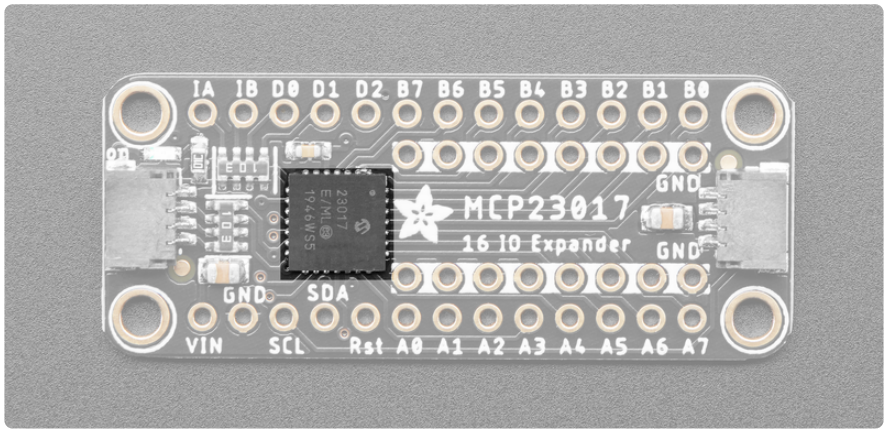


Pins **A0**-**A7** and **B0**-**B7** are bidirectional digital input and digital output pins, for a total of 16 inputs or outputs. When a pin is an output, it can drive up to 25 mA. When a pin is an input, it can have an optional pull-up. The A pins are PORTA and the B pins are PORTB.

Despite the naming convention, the A prefix **does not** stand for analog input/output.

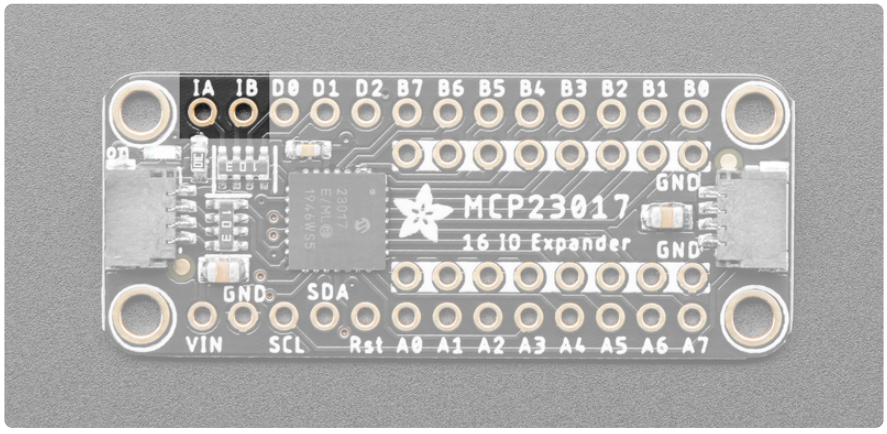
Despite the naming convention, the A prefix does not stand for analog input/ output.

MCP23017



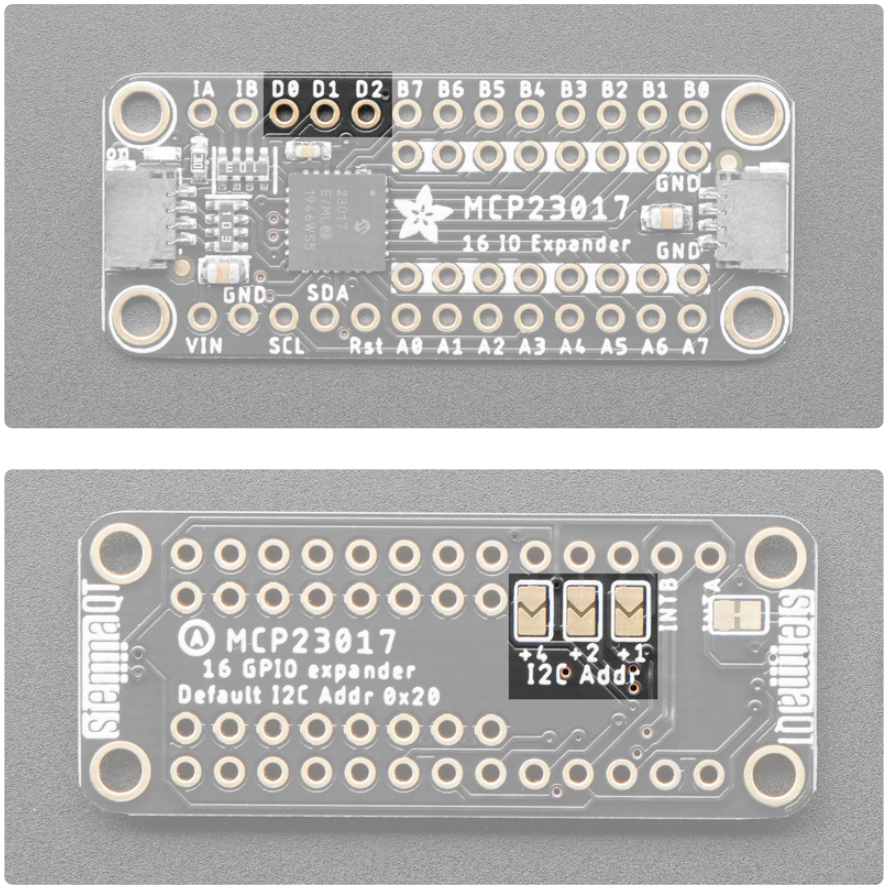
The **MCP23017**, the square chip located in the left-center on the front of the board, provides general purpose parallel I/O expansion over I2C.

Interrupt Pins



- **IA (INTA)** - configurable for what inputs to keep track of on PORTA. It can be configured as active-high, active-low or open-drain.
- **IB (INTB)** - configurable for what inputs to keep track of on PORTB. It can be configured as active-high, active-low or open-drain.

Address Pins



On the back of the board are **three address jumpers**, labeled **+1**, **+2**, and **+4,** above the **I2C Addr** label on the board silk. These jumpers allow you to chain up to 8 of these boards on the same pair of I2C clock and data pins. To do so, you solder the jumpers "closed" by connecting the two pads.

On the front of the board are three address pins, labeled **D0**, **D1** and **D2**. Just like the jumpers, these pins allow you to change the I2C address to connect multiple boards by connecting them to **VIN**.

The default I2C address is **0x20**. The other address options can be calculated by “adding” the **D0/D1/D2** to the base of **0x20**.

D0 sets the lowest bit with a value of **1**, **D1** sets the next bit with a value of **2** and **D2** sets the next bit with a value of **4.** The final address is **0x20** + **D2** + **D1** + **D0** which would be **0x27**.

So for example if **D2** is soldered closed and **D0** is soldered closed, the address is **0x20** + **4** + **1** = **0x25**.

If only **D0** is soldered closed, the address is **0x20** + **1** = **0x21**

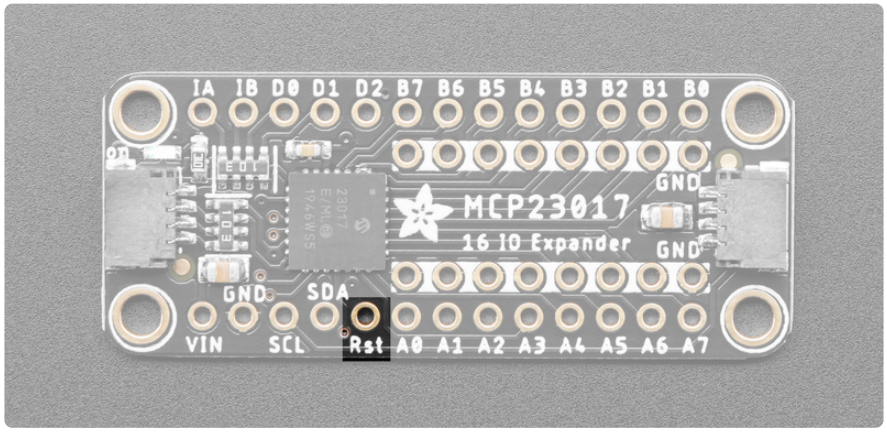
If only **D1** is soldered closed, the address is **0x20** + **2** = **0x22**

If only **D2** is soldered closed, the address is **0x20** + **4** = **0x24**

The table below shows all possible addresses, and whether the pin(s) should be high (closed) or low (open).

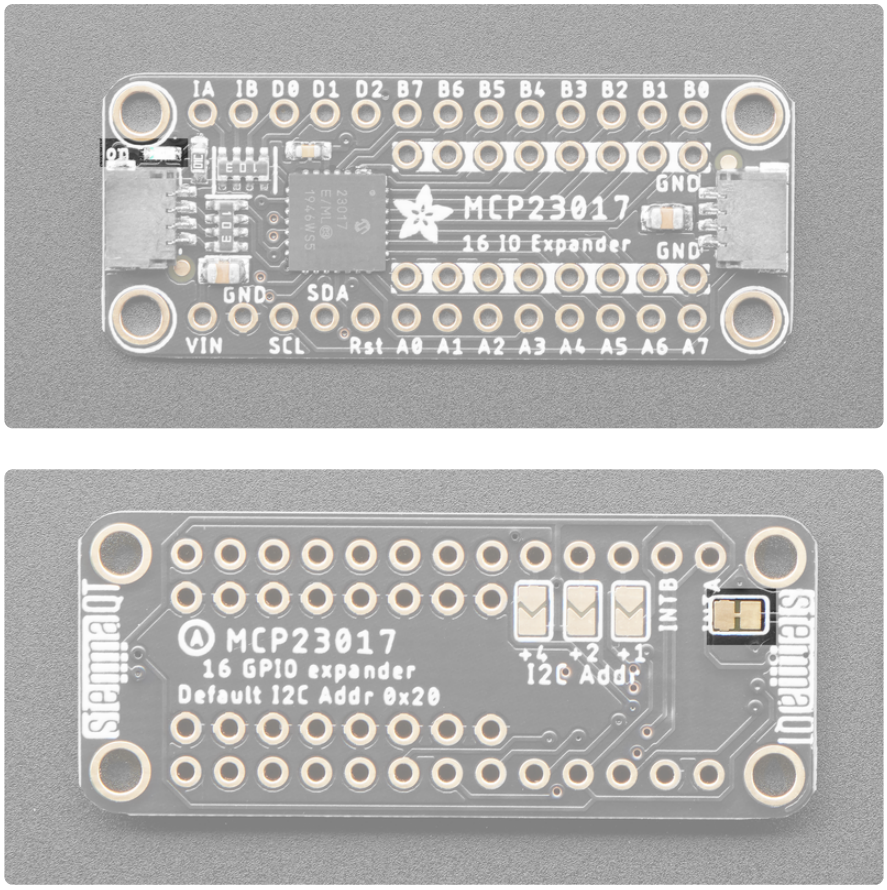
ADDR	D0	D1	D2	ADDR	D0	D1	D2
0x20	L	L	L	0x24	L	L	H
0x21	H	L	L	0x25	H	L	H
0x22	L	H	L	0x26	L	H	H
0x23	H	H	L	0x27	H	H	H

RST Pin



- **RST** - This is the reset pin. Connect this pin to ground to reset the board.

Power LED and Jumper



- **Power LED -** In the upper left corner, above the STEMMA connector, on the front of the board, is the power LED, labeled **on**.
- **LED jumper** - On the back of the board is a jumper for the power LED. If you wish to disable the power LED, simply cut the trace on this jumper.

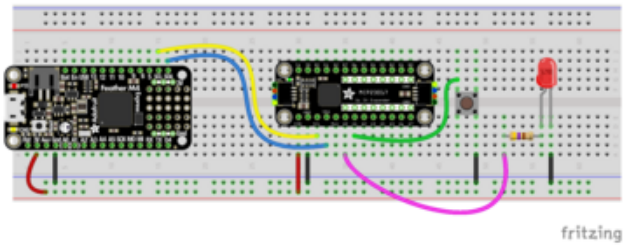
Python & CircuitPython

It's easy to use the **Adafruit MCP23017** with Python or CircuitPython with the Adafruit CircuitPython MCP23017 (https://adafru.it/Boh) module. This module allows you to easily write Python code that adds up to 16 inputs or outputs over I2C.

You can use this sensor 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

First wire up a MCP23017 to your board exactly as shown below. Here's an example of wiring a Feather M4, a button and an LED to the expander using a solderless breadboard:



MCP23017

Board 3V to **expander VIN (red wire)**
Board GND to **expander GND (black wire)**
Board SCL to **expander SCL (yellow wire)**
Board SDA to **expander SDA (blue wire)**
LED

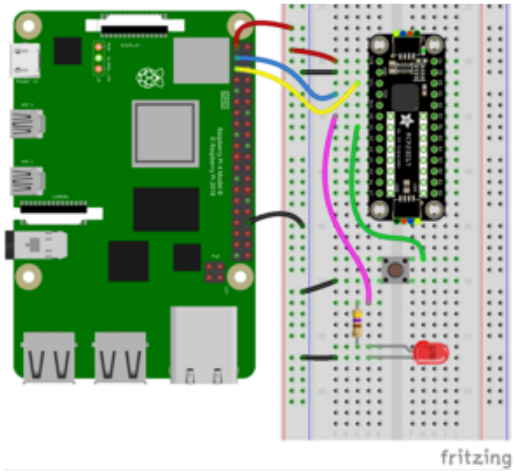
LED + to **470Ω resistor**
LED - to **Board GND (black wire)**
470Ω resistor to **expander A0 (pink wire)**
Button

Button leg to **expander A1 (green wire)**
Opposite button leg to **Board GND (black wire)**

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, a button and an LED wired to the expander using a solderless breadboard:



MCP23017

Pi 3V to **expander VIN (red wire)**
Pi GND to **expander GND (black wire)**
Pi SCL to **expander SCL (yellow wire)**
Pi SDA to **expander SDA (blue wire)**
LED

LED + to **470Ω resistor**
LED - to **Pi GND (black wire)**
470Ω resistor to **expander A0 (pink wire)**
Button

Button leg to **expander A1 (green wire)**
Opposite button leg to **Pi GND (black wire)**

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:

- `pip3 install adafruit-circuitpython-mcp230xx`

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

CircuitPython Usage

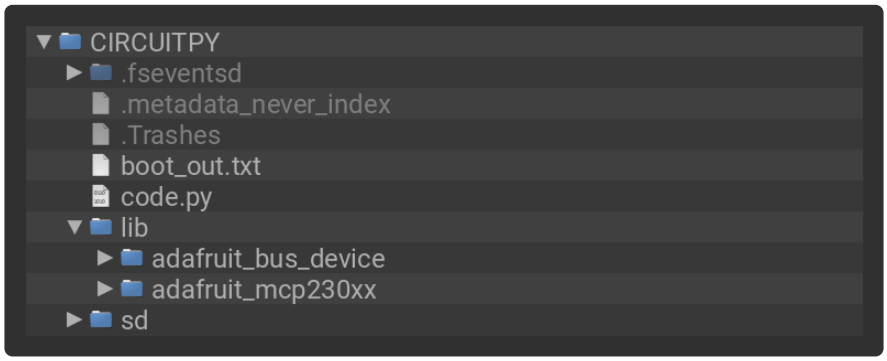
To use with CircuitPython, you need to first install the MCP230xx library, and its dependencies, into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary libraries and the **code.py** file

in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following folders:

- **adafruit_bus_device/**
- **adafruit_mcp230xx/**



Before running the code, comment out the `adafruit_mcp230xx.mcp23008` library and uncomment the `adafruit_mcp230xx.mcp23017` library to import the correct library for the MCP23017.

```
# from adafruit_mcp230xx.mcp23008 import MCP23008

from adafruit_mcp230xx.mcp23017 import MCP23017
```

Then, comment out the `mcp` instance using the `MCP23008` class, and uncomment the `mcp` instance using the `MCP23017` class.

```
# 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
```

Python Usage

Once you have the library `pip3` installed on your computer, copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

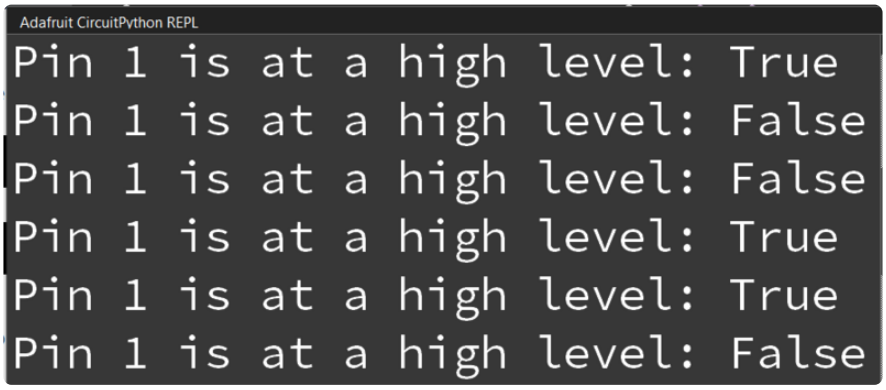
`python3 code.py`

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))
```

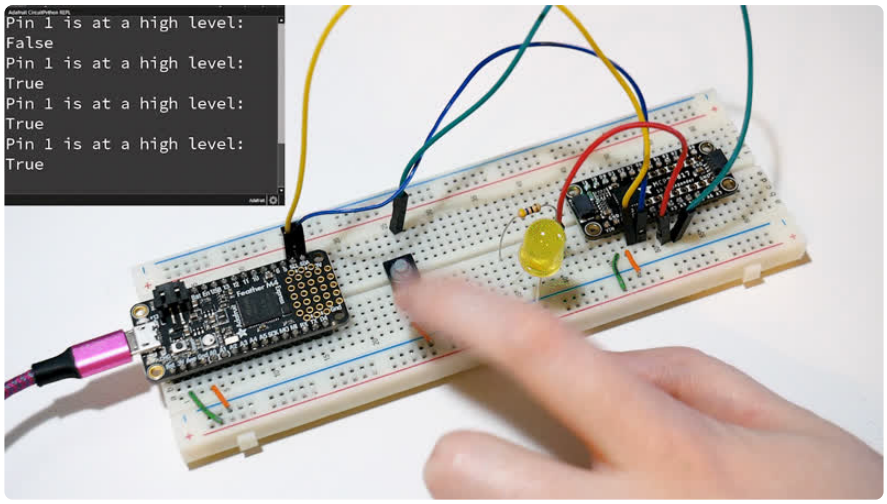
If running CircuitPython: Once everything is saved to the **CIRCUITPY** drive, connect to the serial console (https://adafru.it/Bec) to see the data printed out!

If running Python: The console output will appear wherever you are running Python.



You will see the connected LED begin to blink on and off.

When you press the button, its status will be printed to the REPL.



That's all there is to using the MCP23017 with CircuitPython!

Python Docs

Python Docs (https://adafru.it/Z8F)

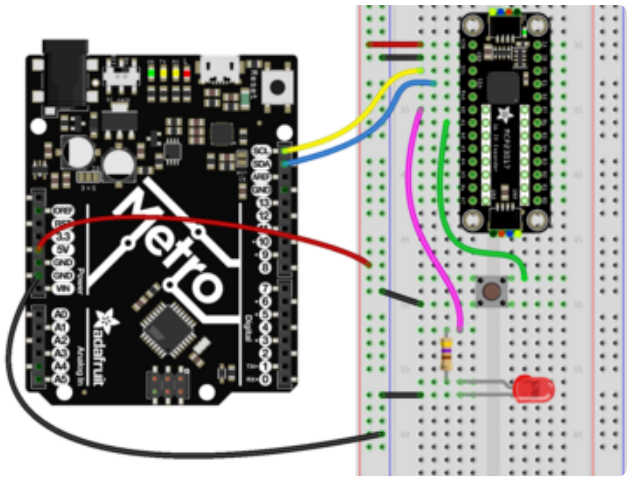
Arduino

Using the MCP23017 with Arduino involves wiring up the expander to your Arduino-compatible microcontroller, installing the Adafruit MCP23017 (https://adafru.it/jFN) library and running the provided example code.

Wiring

Wire as shown for a **5V** board like an Uno. If you are using a **3V** board, like an Adafruit Feather, wire the board's 3V pin to the MCP23017 VIN.

Here is an Adafruit Metro, a button and an LED wired up to the MCP23017 using a solderless breadboard:



MCP23017

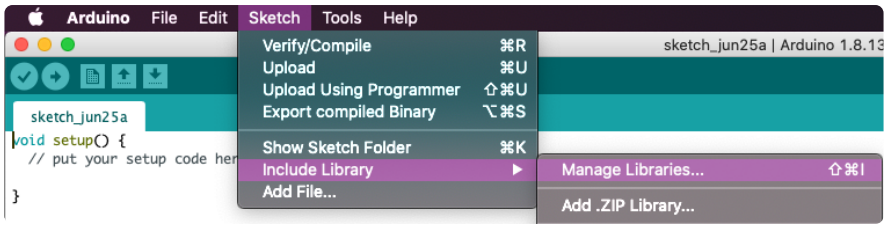
Board 5V to **expander VIN (red wire)**
Board GND to **expander GND (black wire)**
Board SCL to **expander SCL (yellow wire)**
Board SDA to **expander SDA (blue wire)**
LED

LED + to **470Ω resistor**
LED - to **Board GND (black wire)**
470Ω resistor to **expander A0 (pink wire)**
Button

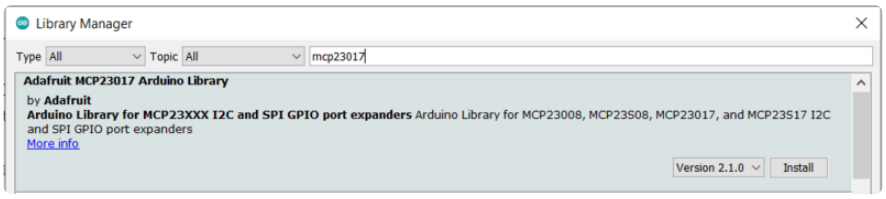
Button leg to **expander A1 (green wire)**
Opposite button leg to **Board GND (black wire)**

Library Installation

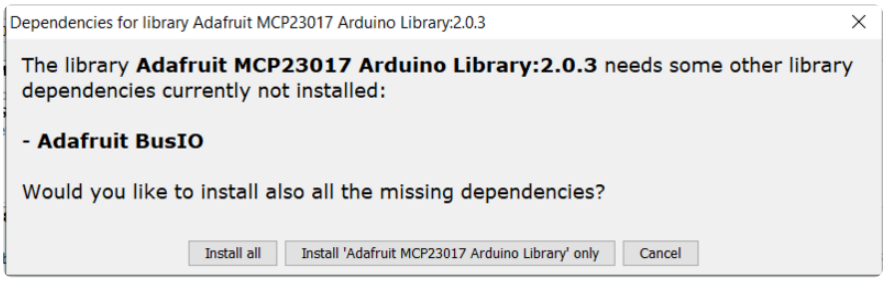
You can install the **MCP23017** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **MCP23017**, and select the **Adafruit MCP23017 Arduino Library** library:



If asked about dependencies, click "Install all".



If the "Dependencies" window does not come up, then you already have the dependencies installed.

If the dependencies are already installed, you must make sure you update them through the Arduino Library Manager before loading the example!

Load Example

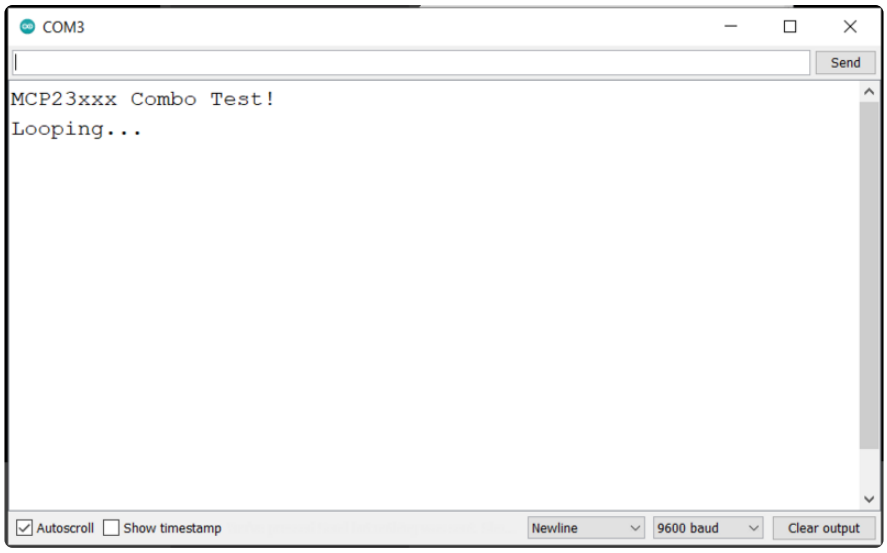
Open up **File -> Examples -> Adafruit MCP23017 Arduino Library -> mcp23xxx_combo**. Before uploading the code, comment out the `mcp` instance using the `Adafruit_MCP23X08` class, and uncomment the `mcp` instance using the `Adafruit_MCP23X17` class.

```
// uncomment appropriate line
// Adafruit_MCP23X08 mcp;
Adafruit_MCP23X17 mcp;
```

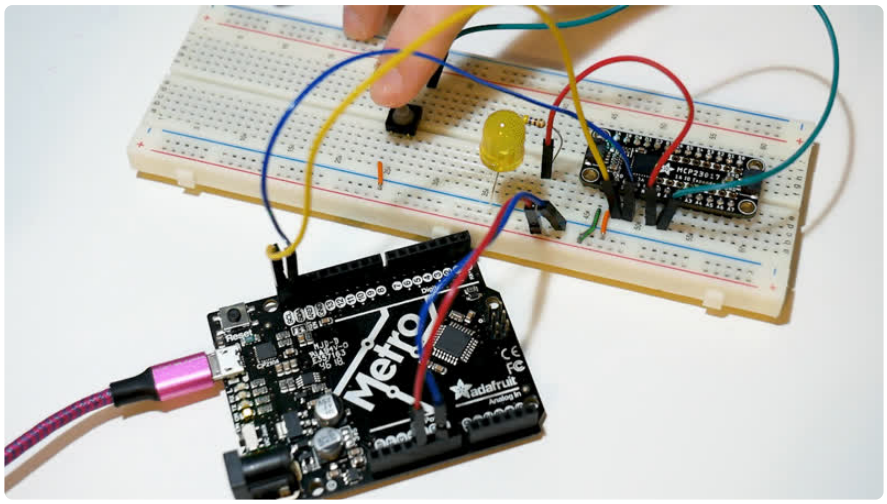
```
// Controls an LED via an attached button.

// ok to include only the one needed
// both included here to make things simple for example
#include <Adafruit_MCP23X08.h>
#include <Adafruit_MCP23X17.h>

#define LED_PIN 0     // MCP23XXX pin LED is attached to
#define BUTTON_PIN 1  // MCP23XXX pin button is attached to

// only used for SPI
#define CS_PIN 6

// uncomment appropriate line
Adafruit_MCP23X08 mcp;
//Adafruit_MCP23X17 mcp;

void setup() {
```

```
  Serial.begin(9600);
  //while (!Serial);
  Serial.println("MCP23xxx Combo Test!");

  // uncomment appropriate mcp.begin
  if (!mcp.begin_I2C()) {
  //if (!mcp.begin_SPI(CS_PIN)) {
    Serial.println("Error.");
    while (1);
  }

  // configure LED pin for output
  mcp.pinMode(LED_PIN, OUTPUT);

  // configure button pin for input with pull up
  mcp.pinMode(BUTTON_PIN, INPUT_PULLUP);

  Serial.println("Looping...");
}

void loop() {
  mcp.digitalWrite(LED_PIN, !mcp.digitalRead(BUTTON_PIN));
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 9600 baud. You should see the text "`Looping...`" in the Serial Monitor.



When you press the button, the LED will light up.



That's all there is to using the MCP23017 STEMMA breakout with Arduino!

Arduino Docs

Arduino Docs (https://adafru.it/jFN)

Downloads

Files

- MCP23017 Datasheet (https://adafru.it/Z8A)
- EagleCAD PCB Files on GitHub (https://adafru.it/Z8B)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/Z8C)

Schematic and Fab Print for MCP23017