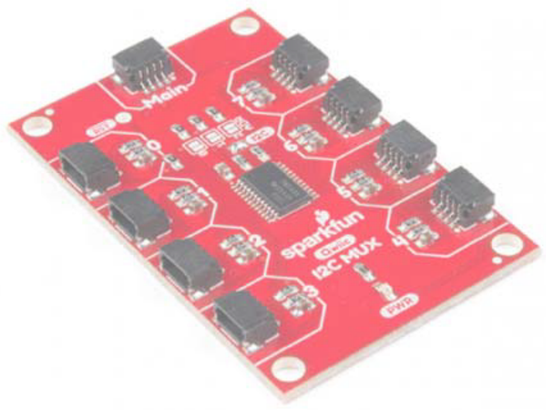
Qwiic MUX Hookup Guide

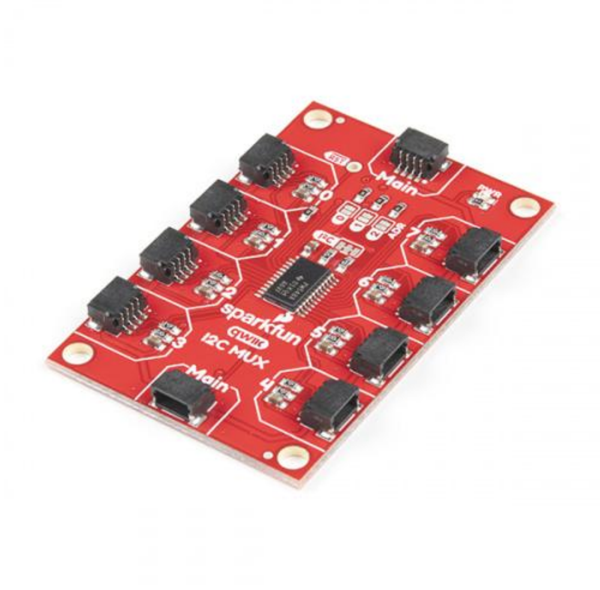
Introduction

PCA9548A and TCA9548A? The SparkX version of the Qwiic Mux breakout used the PCA9548A. The SparkFun red version uses the TCA9548A. Overall, both should be functionally the same with a few minor differences.

The Qwiic Mux - TCA9548A (v1) and (v1.1) enable communication with multiple I^2C devices that have the same address. The IC is simple to interface with and also has 8 configurable addresses of its own, this allows you to put 64 I^2C buses on a single bus!



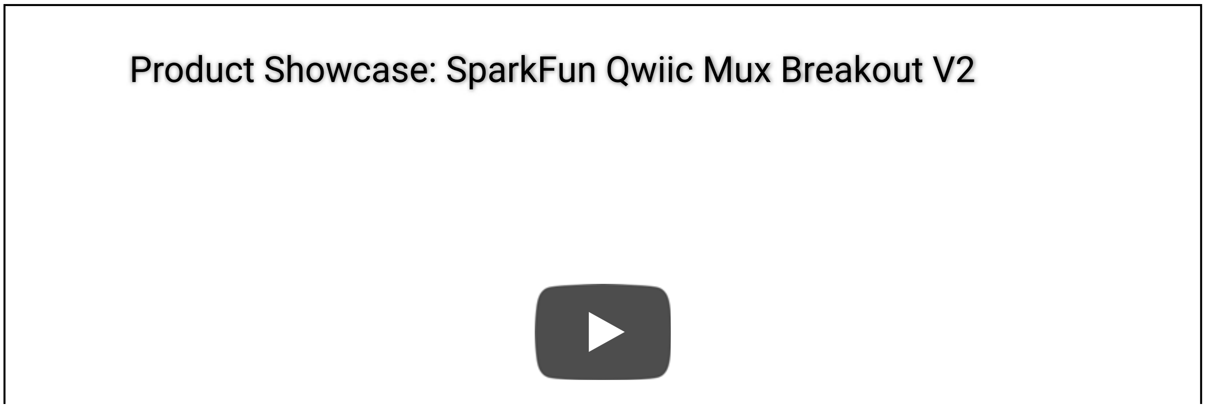
SparkFun Qwiic Mux Breakout - 8 Channel (TCA9548A)
⊘ BOB-14685



SparkFun Qwiic Mux Breakout - 8 Channel (TCA9548A)

○ BOB-16784

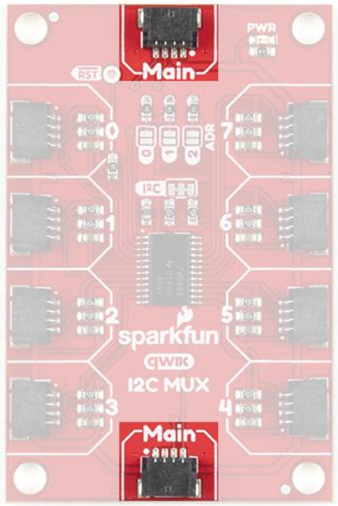
Product Showcase: Qwiic Mux



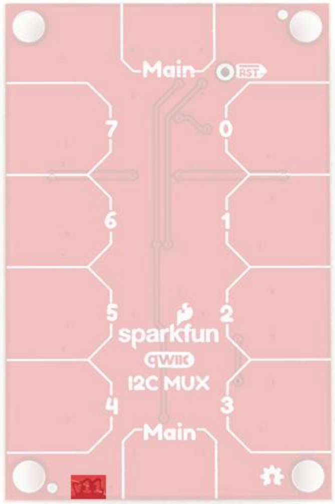
Product Showcase: SparkFun Qwiic Mux Breakout V2

Revision Update: In the latest revision of the Qwiic MUX, we have made a few changes to improve the board, listed below. If users are unsure about which version they purchased, please refer to the pictures of the most prominent changes, shown below.

- We have added a pass-through Qwiic connector.
  - The power LED was moved to accommodate the additional Qwiic connector.
- We have updated the I^{2}C pull-up resistor jumper to a cuttable trace.
- We have widened the trace for the 3.3V power.
- There is a ***v11*** label on the back of the board.
- The silk screen labels have been updated.



Added Qwiic connector to make board daisy-chainable.



*The **v11** label on the back.*

In this tutorial we'll go over how to talk to sensors on different channels of your MUX breakout board. The application of this is pretty straightforward so things won't get too fancy.

Required Materials

You'll need a few additional items to get started with the Qwiic Mux. The RedBoard Qwiic is for the Arduino examples and the Qwiic pHAT is for the Raspberry Pi example (see note below). You may already have a few of these items, so feel free to modify your cart based on your needs. Additionally, there are also alternative parts options that are available as well (*click button below to toggle options*).

SparkFun RedBoard Qwiic
◉ DEV-15123

SparkFun Qwiic Cable Kit
◉ KIT-15081

SparkFun Qwiic pHAT for Raspberry Pi

⊘ DEV-15351
Retired

ALTERNATIVE PARTS (TOGGLE)

Raspberry Pi Example: If you don't already have them, you will need a Raspberry Pi and standard peripherals. An example setup is listed below. ~~*(The Qwiic Mux and Python library have not been tested on the newly released Raspberry Pi 4 because we don't carry it in out catalog yet.*~~)

Update: This board and the Python package have been verified to work with the Raspberry Pi 4.

Tools

Depending on your setup, you may need a hobby knife, soldering iron, solder, and/or general soldering accessories.

Weller WLC100 Soldering Station
⊖ TOL-14228

Solder Lead Free - 100-gram Spool
◉ TOL-09325

Hobby Knife
◉ TOL-09200

Slice Craft Knife
⊘ TOL-14508
Retired

Suggested Reading

If you're unfamiliar with jumper pads, I^{2}C, Qwiic, or Python be sure to checkout some of these foundational tutorials.

Serial Communication

Asynchronous serial communication concepts: packets, signal levels, baud rates, UARTs and more!

Logic Levels

Learn the difference between 3.3V and 5V devices and logic levels.

I2C

An introduction to I2C, one of the main embedded communications protocols in use today.

Serial Terminal Basics

This tutorial will show you how to communicate with your serial devices using a variety of terminal emulator applications.

Raspberry Pi SPI and I2C Tutorial

Learn how to use serial I2C and SPI buses on your Raspberry Pi using the wiringPi I/O library for C/C++ and spidev/smbus for Python.

Raspberry Pi 3 Starter Kit Hookup Guide

Guide for getting going with the Raspberry Pi 3 Model B and Raspberry Pi 3 Model B+ starter kit.

Python Programming Tutorial: Getting Started with the Raspberry Pi

Qwiic pHAT for Raspberry Pi Hookup Guide

This guide will show you how to write programs on your Raspberry Pi using Python to control hardware.

Get started interfacing your Qwiic enabled boards with your Raspberry Pi. The Qwiic pHAT connects the I2C bus (GND, 3.3V, SDA, and SCL) on your Raspberry Pi to an array of Qwiic connectors.

How to Work with Jumper Pads and PCB Traces

Handling PCB jumper pads and traces is an essential skill. Learn how to cut a PCB trace, add a solder jumper between pads to reroute connections, and repair a trace with the green wire method if a trace is damaged.

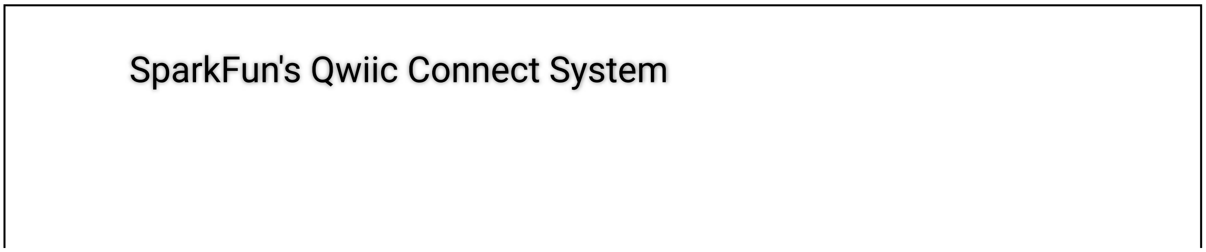
Qwiic Shield for Arduino & Photon Hookup Guide

Get started with our Qwiic ecosystem with the Qwiic shield for Arduino or Photon.

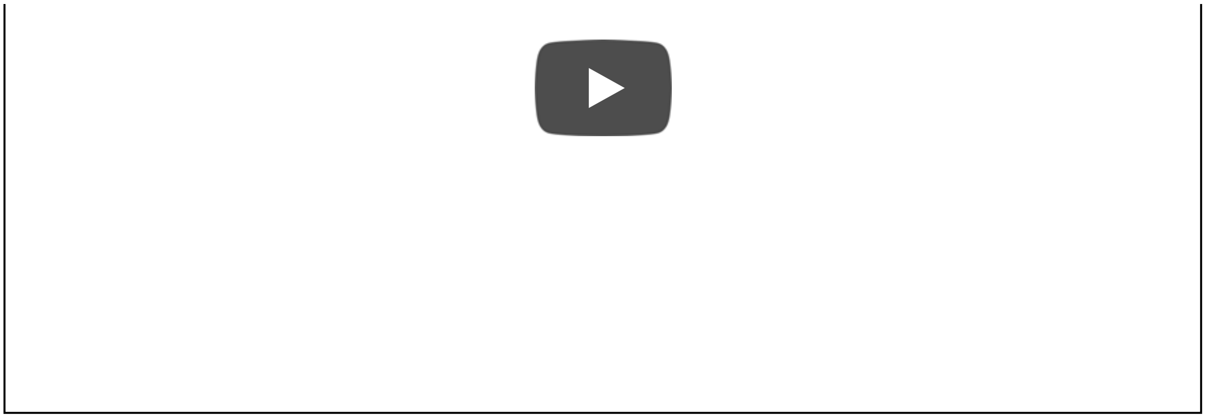
RedBoard Qwiic Hookup Guide

This tutorial covers the basic functionality of the RedBoard Qwiic. This tutorial also covers how to get started blinking an LED and using the Qwiic system.

The Qwiic Mux is intended for the Qwiic connect system. We recommend familiarizing yourself with the **Logic Levels** and **I^{2}C** tutorials before using it. Click on the banner above to learn more about our Qwiic products.



SparkFun's Qwiic Connect System



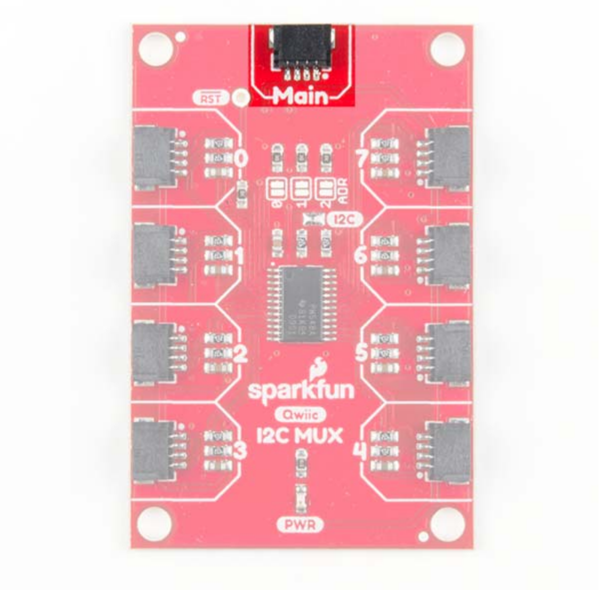
Hardware Overview

What is the difference between the PCA9548A and TCA9548A? Very little. PCA is made by NXP, TCA is made by TI. PCA can operate from 2.3 to 5.5V, TCA can operate from 1.65 to 5.5V. Everything else is identical.

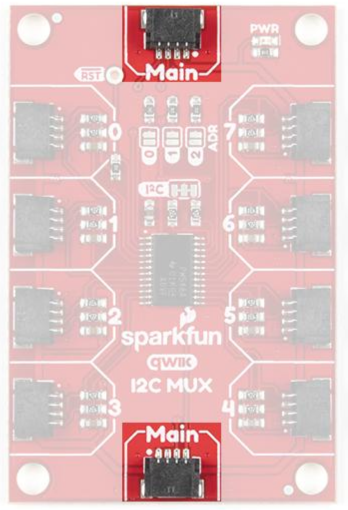
Let's look over a few characteristics of the TCA9548A so we know a bit more about how it behaves.

Characteristic	Range
Operating Voltage	**1.65V - 5.5V**
Operating Temperature	-40 - 85° C
I^{2}C Address	**0x70 (default)** up to 0x77 (see below table)

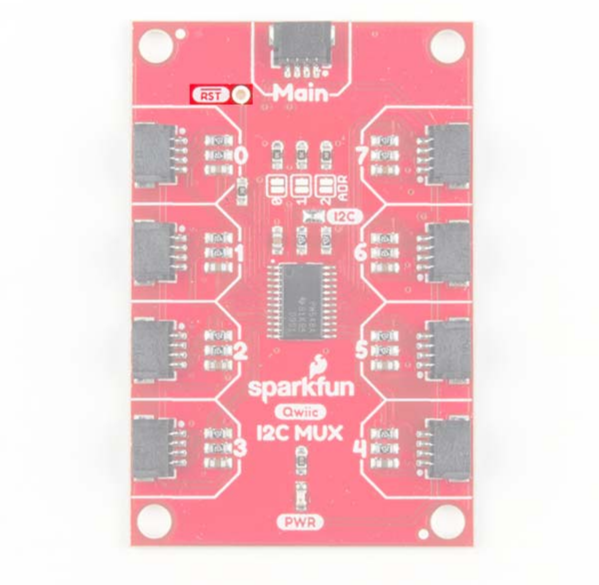
Original Release: The Qwiic input for the Mux is located at the top-center of the board, labeled `Main` , highlighted in the image below. The outputs are then located on the left and right sides of the board and are numbered accordingly.



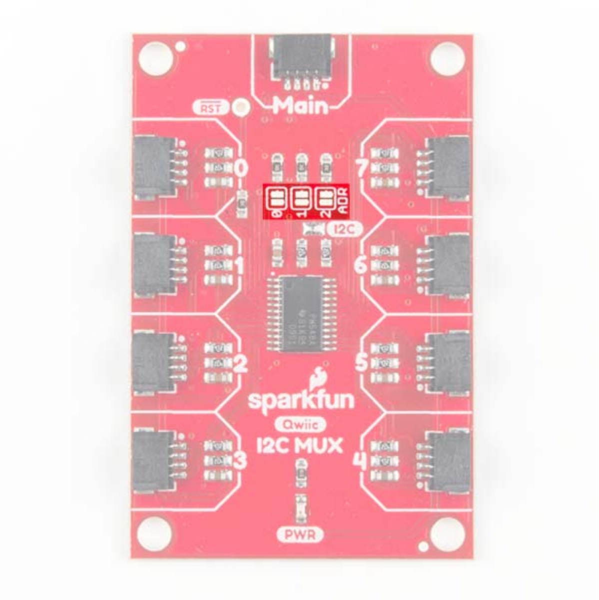
Revision Update: There are two `Main` connectors on the revision to provide a pass through for the I^{2}C bus connection. This allow the board to be daisy chained in between other Qwiic boards.



The onboard reset pin, highlighted below, is an active low input. Pulling reset low for at least 6 ns will restart the multiplexer.



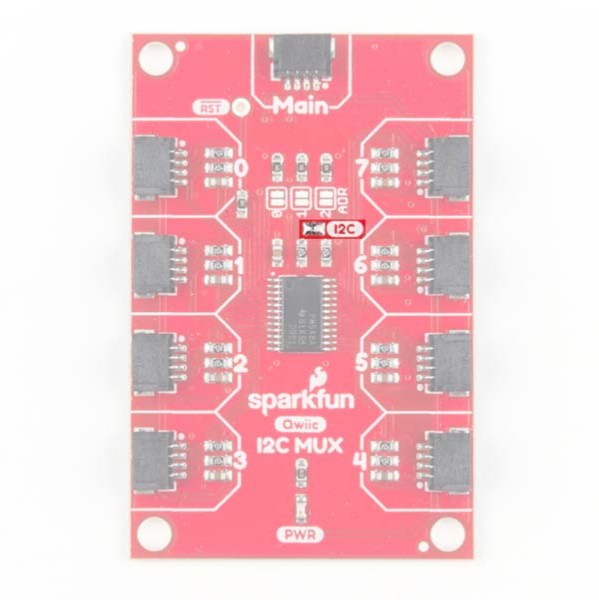
The Qwiic Mux also allows you to change the last 3 bits of the address byte, allowing for 8 jumper selectable addresses if you happen to need to put more than one Mux on the same I^{2}C port. The address can be changed by adding solder to any of the three **ADR** jumpers, shown in the image below.



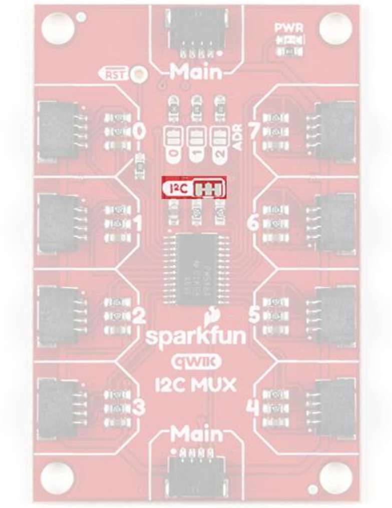
The below table shows which jumpers must be soldered together to change to the corresponding address.

I^{2}C Address	**ADR2**	**ADR1**	**ADR0**
0x70	Open	Open	Open
0x71	Open	Open	Closed
0x72	Open	Closed	Open
0x73	Open	Closed	Closed
0x74	Closed	Open	Open
0x75	Closed	Open	Closed
0x76	Closed	Closed	Open
0x77	Closed	Closed	Closed

Original Release: If you want to remove the pullup resistors from the I^{2}C bus, simply remove the solder from the jumper highlighted in the below image.



Revision Update: If you want to remove the pullup resistors from the I^{2}C bus, simply cut the jumper pad with a hobby knife. Be careful of the **3.3V** trace above the jumper pad. Cutting the trace probably won't destroy the board, as the trace also circles around through the Qwiic connector on the bottom. However, it will leave the trace exposed at the cut.



Hardware Assembly

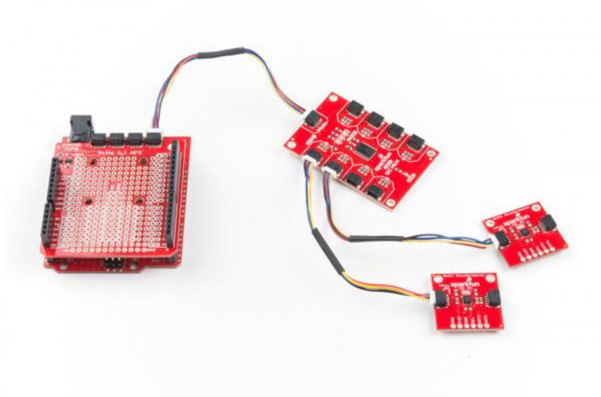
Support Tip: Make sure that you are connecting the **Main** Qwiic connector to the Arduino board/shield or Raspberry HAT.

Arduino Example

Just plug one end of the Qwiic cable into the **Main** connector on the Qwiic multiplexer breakout and the other end into Qwiic connector on your Qwiic enabled microcontroller. If it seems like it's too easy to use, but that's why we made it that way! Otherwise, if you chose to use a Qwiic shield/adapter, now would be the time to head on over to that tutorial to assemble the shield.

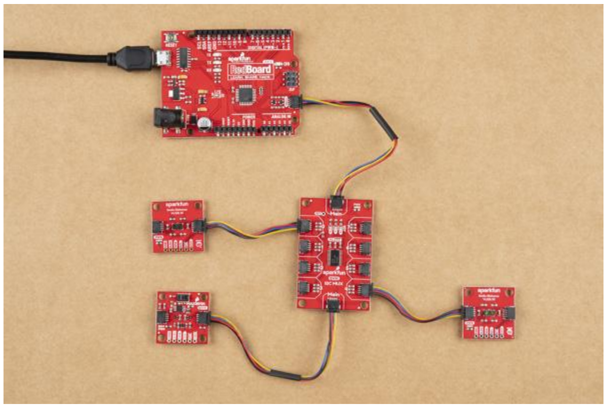
QWIIC SHIELD FOR ARDUINO PHOTON HOOKUP GUIDE

With the shield assembled, SparkFun's new Qwiic environment means that connecting the mux could not be easier. Just plug one end of the Qwiic cable into the Qwiic multiplexer breakout (***Main** connector*) and the other into the Qwiic Shield; then, you'll be ready to upload a sketch and figure out just how all those address sharing sensors are behaving.



SparkFun RedBoard with the Qwiic shield connected the Qwiic Mux with a Qwiic cable. Additionally, a few peripheral Qwiic Boards are attached to verify Qwiic Mux functionality.

Revision Update: Users can now daisy chain the Qwiic Mux in between other Qwiic boards.



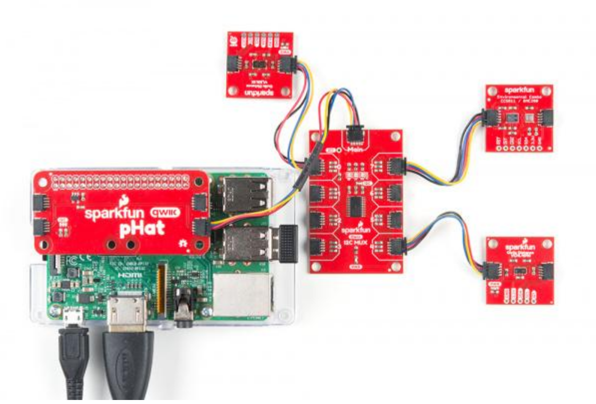
Raspberry Pi Examples

~~**Note:** This board and the Python package have not been tested on the newly released Raspberry Pi 4 because we don't carry it in out catalog yet.~~

Update: This board and the Python package have been verified to work with the Raspberry Pi 4.

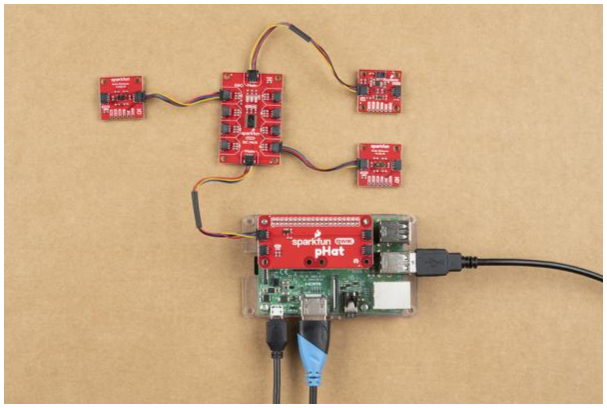
With the Qwiic connector system, assembling the hardware is simple. In addition to the Qwiic Mux, you will need: a Qwiic cables, a SparkFun Qwiic pHAT for Raspberry Pi, and a Raspberry Pi setup with the Raspbian OS, monitor, and standard peripherals. (**If you are unfamiliar with the Qwiic pHAT, you can find the Hookup Guide here*.)

Note: The peripheral Qwiic boards attached to the Qwiic Mux are not necessary for this hookup guide, but they are useful for customers looking to verify the functionality of the Qwiic Mux. Use channels 0, 4, and 7 for the examples below.

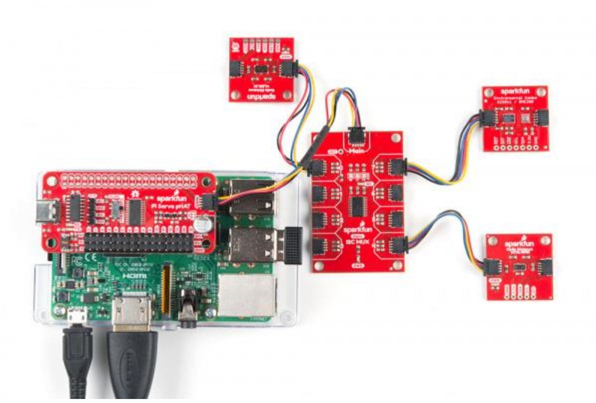


Raspberry Pi 3B connected the Qwiic Mux with a Qwiic pHAT and Qwiic cable (and a few peripheral Qwiic Boards).

Revision Update: Users can now daisy chain the Qwiic Mux in between other Qwiic boards.



Alternatively, you can also use another Qwiic adapter like the Pi Servo pHat instead.



Raspberry Pi 3 connected the Qwiic Mux with the Pi Servo pHat and Qwiic cable (and a few peripheral Qwiic Boards).

Note: This tutorial assumes you are familiar with using a Raspberry Pi and you have the latest (full... with recommended software) version of Raspbian OS your Raspberry Pi. You can download the latest version of the Raspbian OS from the Raspberry Pi Foundation website. As of Feb. 13th 2019, we recommend the **Raspbian Stretch with desktop and recommended software** option.

If this is your first time using a Raspberry Pi, please head over to the Raspberry Pi Foundation website to use their quickstart guides. We have listed a few of them here:

1. Setting up your Raspberry Pi
2. Using your Raspberry Pi
3. Documentation:
   - Setup Documentation
   - Installation Documentation
   - Raspbian Documentation
   - SD card Documentation

Arduino Example

Note: This tutorial assumes you are familiar with Arduino products and you are using the latest stable version of the Arduino IDE on your desktop. If this is your first time using the Arduino IDE, please review our tutorial on installing the Arduino IDE.

SparkFun has written some example code to enable and disable ports on the Qwiic Mux. Go ahead and download this example code here.

QWIIC MUX EXAMPLE (ZIP)

Warning! Make sure to have the **Mux_Control.ino** in the same folder when compiling the **Example1-BasicReadings.ino** sketch file. Otherwise, you may have issues uploading code.

Additionally, you will need to install the MMA8452Q Arduino library if you are using two MMA8452Q accelerometers. First, you'll need the Sparkfun MMA8452Q Arduino library. You can obtain these libraries through the Arduino Library Manager. Search for **Sparkfun MMA8452Q Accelerometer** by **Jim@SparkFun Electronics** to install the latest version. If you prefer downloading the libraries from the GitHub repository and manually installing it, you can grab them here:

DOWNLOAD SPARKFUN MMA8452Q ACCELEROMETER (ZIP)

Arduino Example Example1-BasicReadings.ino

Opening `Example1-BasicReadings` will open two tabs in the Arduino IDE, the first example, and also `Mux_Control`. Let's take a look under the hood of `Mux_Control` to get an idea of what's going on. There are two functions here, `boolean enableMuxPort(byte portNumber)` and `boolean disableMuxPort(byte portNumber)` which is pretty much all we need to specify which channels we'd like to talk to on the Mux. If we have a sensor on channel 0, we simply call `enableMuxPort(0)` to open that channel on the multiplexer. Then we'll take whatever reads and perform whatever actions we'd like to the sensor on that channel. Once finished, we have to call `disableMuxPort(0)` to close communication on that channel so we don't accidentally perform actions on the sensor on that channel. The below example code shows how to read from two MMA8452Q accelerometers.

```
#include <Wire.h>
#include <SFE_MMA8452Q.h> //From: https://github.com/sparkfun/SparkFun_MMA8452Q_Arduino_Library

MMA8452Q accel;

#define NUMBER_OF_SENSORS 2

void setup()
{
  Serial.begin(9600);
  Serial.println("Qwiic Mux Shield Read Example");

  Wire.begin();

  //Initialize all the sensors
  for (byte x = 0 ; x < NUMBER_OF_SENSORS ; x++)
  {
    enableMuxPort(x); //Tell mux to connect to port X
    accel.init(); //Init the sensor connected to this port
    disableMuxPort(x);
  }

  Serial.println("Mux Shield online");
}

void loop()
{
  for (byte x = 0 ; x < NUMBER_OF_SENSORS ; x++)
  {
    enableMuxPort(x); //Tell mux to connect to this port, and this port only

    if (accel.available())
    {
      accel.read();

      Serial.print("Accel ");
      Serial.print(x);
      Serial.print(": ");
      Serial.print(accel.cx, 2);
      Serial.print(" ");
      Serial.print(accel.cy, 2);
      Serial.print(" ");
      Serial.print(accel.cz, 2);
      Serial.print(" ");

      Serial.println(); // Print new line every time.
    }

    disableMuxPort(x); //Tell mux to disconnect from this port
  }
  delay(1); //Wait for next reading
}
```

With the example provided, you should be able to read two I^{2}C sensors with the same address on the same bus! Try opening up the Arduino Serial Monitor set to **9600** baud in order to read the sensor values.

Python Package Overview

> **Note:** This example assumes you are using the latest version of Python (2 or 3). If this is your first time using Python or I^{2}C hardware on a Raspberry Pi, please checkout our tutorial on Python Programming with the Raspberry Pi and the Raspberry Pi SPI and I2C Tutorial.
>
> **On the Raspberry Pi, Python 2 and 3 are included with the Raspbian OS (with desktop and recommended software) image.*
>
> **Support Tip:** Don't forget to double check that the hardware I^{2}C connection is enabled on your Raspberry Pi or other single board computer.

We've written a Python package to easily get setup and utilize the Qwiic Mux. However, before we jump into operating the multiplexer, let's take a closer look at the available functions in the Python package. You can install the `sparkfun-qwiic-tca9548a` Python package hosted by PyPi.

Installation

To install the Python package for this product, it is recommended that users install the SparkFun Qwiic Python package, which installs all the available Python packages for our Qwiic products and includes the required I^{2}C driver package. On systems that support PyPi installation via `pip3` (use `pip` for Python 2) is simple, using the following commands:

For **all users** (note: the user must have **sudo** privileges):

```
sudo pip3 install sparkfun-qwiic
```

For the **current user**:

```
pip3 install sparkfun-qwiic
```

> **Note:** Users can, alternatively, manually build or individually install the python package (*see instructions below*). The required I^{2}C driver package will still need to be installed as well.
>
> If you prefer to manually download and build the libraries from the GitHub repository, you can grab them here (**Please be aware of any package dependencies. You can also check out the repository documentation page, hosted on ReadtheDocs.*):
>
> **DOWNLOAD THE SPARKFUN TCA9548A PYTHON PACKAGE (ZIP)**

PyPi Installation

This repository is hosted on PyPi as the `sparkfun-qwiic-tca9548a package`. On systems that support PyPi installation via `pip3` (use `pip` for Python 2) is simple, using the following commands:

For **all users** (note: the user must have **sudo** privileges):

```
sudo pip3 install sparkfun-qwiic-tca9548a
```

For the **current user**:

```
pip3 install sparkfun-qwiic-tca9548a
```

Local Installation

To install, make sure the `setuptools` package is installed on the system.

Direct installation at the command line (use `python` for Python 2):

```
python3 setup.py install
```

To build a package for use with `pip3` :

```
python3 setup.py sdist
```

A package file is built and placed in a subdirectory called dist. This package file can be installed using `pip3` .

```
cd dist
pip3 install sparkfun_qwiic_tca9548a-<version>.tar.gz
```

Python Package Operation

Below is a description of the basic functionality of the Python package. This includes the package organization, built-in methods, and their inputs and/or outputs. For more details on how the Python package works, check out the source code and the datasheet.

Dependencies

This Python package has a very few dependencies in the code, listed below:

```
import time                         # Time access and conversion package
import qwiic_i2c                    # I2C bus driver package
```

Default Variables

The default variables, in the code, for this Python package are listed below:

```
#The name of this device
_DEFAULT_NAME = "Qwiic Mux"

_AVAILABLE_I2C_ADDRESS = [*range(0x70,0x77 + 1)]
```

Class

`QwiicTCA9548A()` or **`QwiicTCA9548A(i2caddr)`**
This Python package operates as a class object, allowing new instances of that type to be made. An `__init__()` constructor is used that creates a connection to an I^{2}C device over the I^{2}C bus using the default or specified I^{2}C address.

The Constructor

A constructor is a special kind of method used to initialize (assign values to) the data members needed by the object when it is created.

`__init__(self, address = None, debug = None, i2c_driver = None)`

Input: value

The value of the device address. If not defined, the Python package will use the default I^{2}C address (**0x29**) stored under `_AVAILABLE_I2C_ADDRESS` variable. (*The other available addresses are configured with the jumpers on the board.*)

Output: Boolean

True: Connected to I^{2}C device on the default (or specified) address.
False: No device found or connected.

Input: *i2c_driver*

Loads the specified I^{2}C driver; by default the Qwiic I^{2}C driver is used: `qwiic_i2c.getI2CDriver()`. Users should use the default I^{2}C driver and leave this field blank.

Functions

A function that is an attribute of the class, which defines a method for instances of that class. In simple terms, they are objects for the operations (or methods) of the class.

`.is_connected()`
Determine if the device is conntected to the system.

Output: Boolean

True: Connected to I^{2}C device on the default (or specified) address.
False: No device found or connected.

`.disable_all()`
This method disables the connection of all channels on the Qwiic Mux.

`.disable_channels(disable)`
This method disables the connection of specific channels on the Qwiic Mux.

Intput: Integer or List

Channel(s) to disable on the Qwiic Mux.
Range: 0 to 7 (**The method will automatically convert an individual integer into a list.*)

`.enable_all()`
This method enables the connection of specific channels on the Qwiic Mux.

`.enable_channels(enable)`
This method enables the connection of specific channels on the Qwiic Mux.

Intput: Integer or List

Channel(s) to enable on the Qwiic Mux.
Range: 0 to 7 (**The method will automatically convert an individual integer into a list.*)

`.list_channels()`
This method lists all the available channels and their current configuration (enabled or disabled) on the Qwiic Mux.

Upgrading the Package

In the future, changes to the Python package might be made. Updating the installed packages has to be done individually for each package (i.e. sub-modules and dependencies won't update automatically and must be updated manually). For the `sparkfun-qwiic-tca9548a` Python package, use the following command (use `pip` for Python 2):

For **all users** (note: the user must have **sudo** privileges):

```
sudo pip3 install --upgrade sparkfun-qwiic-tca9548a
```

For the **current user**:

```
pip3 install --upgrade sparkfun-qwiic-tca9548a
```

Python Examples

The following examples are available in the GitHub repository. To run the examples, simple download or copy the code into a file. Then, open/save the example file (if needed) and execute the code in your favorite Python IDE.

For example, with the default Python IDLE click **Run > Run Module** or use the `F5` key. To terminate the example use the `Ctrl + C` key combination.

Support Tip: To check for changes in the channels of the Qwiic Mux, users can attach various Qwiic boards to the channels (as shown in the **Hardware Assembly** section). With the `i2ctools` package installed on Raspbian, users can verify whether the channels on the Qwiic Mux are enabled with the following command (in the Terminal):

```
i2cdetect -y 1
```

A table will be printed out in the terminal, listing the addresses of the available devices. By running this command before, between, and after the following examples, users can verify the changes in the enabled channels of the Qwiic Mux.

Example 1

Users should run the first example after power up or reset of the board. In this example, channels 0 and 4 are enabled, there is a pause, and then channel 7 is enabled.

Import Dependencies

The first part of the code, imports the required dependencies to operate.

```
import qwiic
import time
```

Initialize Constructor

This line instantiates an object for the device.

```
test = qwiic.QwiicTCA9548A()
```

Test Run

This section of the code, illustrates how to enable the I^{2}C channels on the Qwiic Mux. First, it list all the channels on the Qwiic Mux and their configuration (enabled or disabled). (**On power up or reset all the channels will be disabled.*) The second part of the code, enables channels 0 and 4 using the list method. Then then the code pauses for a second before enabling channel 7. Once that task its complete, the code returns the final configuration of the channels. Users should expect to see channels 0, 4, and 7 enabled.

```
# List Channel Configuration
test.list_channels()

# Enable Channels 0 and 4
test.enable_channels([0,4])

# Pause 1 sec
time.sleep (1)

# Enable Channel 7
test.enable_channels(7)

# List Channel Configuration
test.list_channels()
```

Example 2

Users should run the first example before running this example. This example disables channels 0 and 4.

Import Dependencies

The first part of the code, imports the required dependencies to operate.

```
import qwiic
import time
```

Initialize Constructor

This line instantiates an object for the device.

```
test = qwiic.QwiicTCA9548A()
```

Test Run

This section of the code, illustrates how to disable the I^{2}C channels on the Qwiic Mux. First, it list all the channels on the Qwiic Mux and their configuration (enabled or disabled). (**From the previous example, channels 0, 4, and 7 should be enabled.*) The second part of the code, disables channels 0 and 4 using the list method. Then then the code returns the final configuration of the channels. Users should expect to see only channel 7 enabled.

```
# List Channel Configuration
test.list_channels()

# Enable Channels 0 and 4
test.disable_channels([0,4])

# List Channel Configuration
test.list_channels()
```

Troubleshooting Tips

Here are a few tips for troubleshooting this device.

No Available Devices

Double check your connections. On a Raspberry Pi, you may get this is indicated with an `OSError: [Errno 121] Remote I/O error` readout.

On a Raspberry Pi, also make sure that the I^{2}C hardware is enabled. This is usually indicated with an `Error: Failed to connect to I2C bus 1.` readout.

Checking Your I^{2}C Connection

A simple method to check if your Raspberry Pi can communicate over I^{2}C with the Qwiic Mux, is to ping the I^{2}C bus. On the latest releases of Raspbian Stretch, the `i2ctools` package should come pre-installed. If it isn't run the following command in the terminal:

```
sudo apt-get install i2ctools
```

Once the `i2ctools` package is installed, you can ping the I^{2}C bus with the following command in the terminal:

```
i2cdetect -y 1
```

You should see a table printed out in the terminal. If the Qwiic Mux is connected/working properly you should see the address space for **0x70** marked with 70.

Resources and Going Further

Now that you've successfully got your Qwiic Mux listening to all of those concurrent addresses, it's time to incorporate it into your own project!

For more information, check out the resources below:

- SparkX (R&D)
  - Schematic (PDF)
  - Eagle Files (ZIP)
  - Product GitHub Repo
- Version 1.0
  - Schematic (PDF)
  - Eagle Files (ZIP)
  - Product GitHub Repo
- Version 1.1

  - Schematic (PDF) (v1.1)
  - Eagle Files (ZIP) (v1.1)
  - Board Dimensions - (v1.1) Dimensions are in inches and the mounting holes are for M3 screws (x4).
  - Product GitHub Repo (v1.1)
  - Product Video
- Datasheet:
  - TCA9548A
  - PCA9548A
- Arduino Example Code:
  - Arduino Example (ZIP)
  - MMA8452Q Arduino Library
- TCA9548A Python Package
  - ReadtheDocs Documentation
- Product GitHub Repo
- SparkFun Product Showcase: Qwiic Mux
  - Gist: Light Spectrogram
- Qwiic Landing Page

Need even more inspiration for your next project? Check out some of these related tutorials:

Qwiic Accelerometer (MMA8452Q) Hookup Guide

Freescale’s MMA8452Q is a smart, low-power, three-axis, capacitive micro-machined accelerometer with 12-bits of resolution. It’s perfect for any project that needs to sense orientation or motion. We’ve taken that accelerometer and stuck it on a Qwiic-Enabled breakout board to make interfacing with the tiny, QFN package a bit easier.

Qwiic Proximity Sensor (VCNL4040) Hookup Guide

The SparkFun Qwiic Proximity Sensor is a great, qualitative proximity (up to 20 cm) and light sensor. This hookup guide covers a few examples to retrieve basic sensor readings.

SparkFun GPS Dead Reckoning NEO-M8U Hookup Guide

The u-blox NEO-M8U is a powerful GPS units that takes advantage of untethered dead reckoning (UDR)

SparkFun Qwiic Shield for Teensy Hookup Guide

A short guide for assembling and using the SparkFun Qwiic Shield for Teensy and Qwiic Shield for Teensy technology for navigation. The module provides continuous positioning for vehicles in urban environments and during complete signal loss (e.g. short tunnels and parking garages). We will quickly get you set up using the Qwiic ecosystem and Arduino so that you can start reading the output!

Or check out this blog post for ideas.

How I Built a Cap Touch Sound Board with the Qwiic System

AUGUST 5, 2020

Extended.