Adafruit ADS7830 8-Channel 8-Bit ADC

Created by Liz Clark

https://learn.adafruit.com/adafruit-ads7830-8-channel-8-bit-adc

Last updated on 2023-11-13 02:52:59 PM EST
# Table of Contents

## Overview

### Pinouts
- Power Pins
- I2C Logic Pins
- ADC Inputs
- Address Pins
- Reference Jumpers
- Power LED and Jumper

## CircuitPython and Python
- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- Python Installation of ADS7830 Library
- CircuitPython Usage
- Python Usage
- Example Code

## Python Docs

## Arduino
- Wiring
- Library Installation
- Example Code

## Arduino Docs

## Downloads
- Files
- Schematic and Fab Print
Overview

Many microcontrollers have ADCs these days, for reading analog/resistive sensors like potentiometers, thermistors, LDR light sensors, etc. but sometimes you need MOAR! Or maybe you're using a single board computer like a Raspberry Pi that has no ADCs at all!

The Adafruit ADS7830 8-Channel 8-Bit ADC with I2C is an affordable 8-channel ADC with an I2C interface, so it's easy to include with any platform. Wire up to the SDA/SCL pins plus power and ground, only four wires required. If you need more than 8 channels, there's 2 address pins so you could have 4 x 8 = 32 total ADC channels with chainable Stemma QT cables.
Note that this chip isn't the fanciest ADC on the market: you can get up 70 Ksamples per second (you'll need 3.4MHz I2C speed support for max speed rate), and only 8-bits of resolution, so it's good for rough sensor measurements. For example, this would definitely be a great way to add a ton of potentiometers or flex sensors to a build, but wouldn't be a good match for precision thermistor conversion.

To use it, check out our Arduino library () which will quickly let you read from any of the 8 channels as single ended inputs - or you can use differential mode which will use two adjacent pins for the positive and negative references. You can power the board from 3 or 5V power and logic, which will also act as the reference voltage and pullup voltage. Or you can change the reference voltage by cutting the jumper on the back and wiring to the breakout pad, as long as it is less than the power supply.
To get you going fast, we spun up a custom-made PCB in the [STEMMA QT form factor](https://www.adafruit.com/product/467), making it easy to interface with. The [STEMMA QT connectors](https://www.adafruit.com/product/467) on either side are compatible with the [SparkFun Qwiic](https://www.adafruit.com/product/467) I2C connectors. This allows you to make solderless connections between your development board and the ADS7830 or to chain it with a wide range of other sensors and accessories using a [compatible cable](https://www.adafruit.com/product/467).

QT Cable is not included, but we have a variety in the shop.

Comes with a bit of 0.1" standard header in case you want to use it with a breadboard or perfboard. Four mounting holes for easy attachment.

### Pinouts

The default I2C address is 0x48.
Power Pins

- VIN - this is the power pin. Since the ADC chip uses 3-5 VDC to power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V.
- REF - this is the positive reference pin. By default this is connected to VIN.
- GND - common ground for power and logic.
- COM - this is the negative reference pin. By default this is connected to GND.

I2C Logic Pins

- SCL - I2C clock pin, connect to your microcontroller I2C clock line. This pin can use 3-5V logic, and there's a 10K pullup on this pin.
- SDA - I2C data pin, connect to your microcontroller I2C data line. This pin can use 3-5V logic, and there's a 10K pullup on this pin.
- STEMMA QT () - These connectors allow you to connect to dev boards with STEMMA QT connectors or to other things with various associated accessories ()

ADC Inputs

- A0-A7 - The 8 inputs to the ADC. These inputs can get up to 70 Ksamples per second (you'll need 3.4MHz I2C speed support for max speed rate), and 8-bits of resolution. They're ideal for adding a ton of potentiometers or flex sensors to a build.

Address Pins

On the back of the board are two address jumpers, labeled AD0 and AD1. These jumpers allow you to chain up to 4 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 two address pins, labeled AD0 and AD1. 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 0x48. The other address options can be calculated by “adding” the AD0/AD1 to the base of 0x48.

AD0 sets the lowest bit with a value of 1 and AD1 sets the next bit with a value of 2. The final address is 0x48 + AD1 + AD0 which would be 0x4B.
If only AD0 is soldered closed, the address is 0x48 + 1 = 0x49

If only AD1 is soldered closed, the address is 0x48 + 2 = 0x4A

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

<table>
<thead>
<tr>
<th>ADDR</th>
<th>A0</th>
<th>A1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x48</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>0x49</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>0x4A</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>0x4B</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

Reference Jumpers

- Ext Ref - On the back of the board, directly below the board label on the silk, is the external reference jumper. Cut this jumper to change the reference voltage. The new external reference voltage must be less than the power supply connected to the board.
- Ext Com - On the back of the board, below the Ext Ref jumper, is the external negative/common reference jumper. Cut this jumper to change the negative reference.

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. It is the green LED.
- LED jumper - In the upper right corner 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.
CircuitPython and Python

It's easy to use the ADS7830 with Python or CircuitPython, and the Adafruit_CircuitPython_ADS7830() module. This module allows you to easily write Python code to read the 8 analog inputs on the ADC.

You can use this driver with any CircuitPython microcontroller board or with a computer that has GPIO and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library.

CircuitPython Microcontroller Wiring

First wire up the breakout to your board exactly as follows. The following is the breakout wired to a Feather RP2040 using the STEMMA connector along with a potentiometer connected to analog input 0 (A0):

- Board STEMMA 3V to breakout VIN (red wire)
- Board STEMMA GND to breakout GND (black wire)
- Board STEMMA SCL to breakout SCL (yellow wire)
- Board STEMMA SDA to breakout SDA (blue wire)
- Breakout VIN to potentiometer positive (red wire)
- Breakout A0 to potentiometer wiper (green wire)
- Breakout GND to potentiometer negative (black wire)

STEMMA Wired Potentiometer Breakout Board - 10K ohm Linear

For the easiest way possible to measure twists, turn to this STEMMA potentiometer breakout (hal!). This plug-n-play pot comes with a JST-PH 2mm connector and a matching

https://www.adafruit.com/product/4493
The following is the breakout wired to a Feather RP2040 using a solderless breadboard:

- Board 3V to breakout VIN (red wire)
- Board GND to breakout GND (black wire)
- Board SCL to breakout SCL (yellow wire)
- Board SDA to breakout SDA (blue wire)
- Breakout VIN to potentiometer positive (red wire)
- Breakout A0 to potentiometer wiper (green wire)
- Breakout GND to potentiometer negative (black wire)

Python Computer Wiring

Since there are 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.

Here's the Raspberry Pi wired with I2C using the STEMMA connector:

- Pi 3V to breakout VIN (red wire)
- Pi GND to breakout GND (black wire)
- Pi SCL to breakout SCL (yellow wire)
- Pi SDA to breakout SDA (blue wire)
- Breakout VIN to potentiometer positive (red wire)
- Breakout A0 to potentiometer wiper (green wire)
- Breakout GND to potentiometer negative (black wire)

Here's the Raspberry Pi wired with I2C using a solderless breadboard:
Pi 3V to breakout VIN (red wire)
Pi GND to breakout GND (black wire)
Pi SCL to breakout SCL (yellow wire)
Pi SDA to breakout SDA (blue wire)
Breakout VIN to potentiometer positive (red wire)
Breakout A0 to potentiometer wiper (green wire)
Breakout GND to potentiometer negative (black wire)

Python Installation of ADS7830 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!

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

```
    pip3 install adafruit-circuitpython-ads7830
```

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 Usage

To use with CircuitPython, you need to first install the Adafruit_CircuitPython_ADS7830 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/
```
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

If running CircuitPython: Once everything is saved to the CIRCUITPY drive, connect to the serial console to see the data printed out!

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

```python
# SPDX-FileCopyrightText: 2023 Liz Clark for Adafruit Industries
# SPDX-License-Identifier: MIT

# Simple demo to read analog input on channel 0

import time
import board
import adafruit_ads7830.ads7830 as ADC
from adafruit_ads7830.analog_in import AnalogIn

i2c = board.I2C()

# Initialize ADS7830
adc = ADC.ADS7830(i2c)
chan = AnalogIn(adc, 0)

while True:
    print(f"ADC channel 0 = {chan.value}"
    time.sleep(0.1)
```

The ADS7830 is initialized over I2C. Then in the loop, the analog input on A0 is read and printed to the serial monitor. As you adjust the potentiometer connected to A0, you'll see the values change.
To make code from different ADC's all compatible, we always return a 16-bit value (0 to 65535) even though, in this case, the underlying hardware is only 8-bit!

<table>
<thead>
<tr>
<th>CircuitPython REPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC channel 0 = 65280</td>
</tr>
<tr>
<td>ADC channel 0 = 65280</td>
</tr>
<tr>
<td>ADC channel 0 = 65280</td>
</tr>
<tr>
<td>ADC channel 0 = 60416</td>
</tr>
<tr>
<td>ADC channel 0 = 45568</td>
</tr>
<tr>
<td>ADC channel 0 = 33024</td>
</tr>
<tr>
<td>ADC channel 0 = 21248</td>
</tr>
<tr>
<td>ADC channel 0 = 11776</td>
</tr>
<tr>
<td>ADC channel 0 = 3328</td>
</tr>
<tr>
<td>ADC channel 0 = 0</td>
</tr>
<tr>
<td>ADC channel 0 = 0</td>
</tr>
<tr>
<td>ADC channel 0 = 0</td>
</tr>
<tr>
<td>ADC channel 0 = 512</td>
</tr>
<tr>
<td>ADC channel 0 = 15360</td>
</tr>
<tr>
<td>ADC channel 0 = 28928</td>
</tr>
<tr>
<td>ADC channel 0 = 50432</td>
</tr>
<tr>
<td>ADC channel 0 = 58368</td>
</tr>
<tr>
<td>ADC channel 0 = 35328</td>
</tr>
<tr>
<td>ADC channel 0 = 9728</td>
</tr>
<tr>
<td>ADC channel 0 = 2816</td>
</tr>
</tbody>
</table>

**Python Docs**

Python Docs

**Arduino**

Using the ADS7830 breakout with Arduino involves wiring up the breakout to your Arduino-compatible microcontroller with a potentiometer, installing the Adafruit_ADS7830 library, and running the provided example code.
STEMMA Wired Potentiometer Breakout Board - 10K ohm Linear

For the easiest way possible to measure twists, turn to this STEMMA potentiometer breakout (ha!). This plug-n-play pot comes with a JST-PH 2mm connector and a matching
https://www.adafruit.com/product/4493

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 breakout VIN.

Here is an Adafruit Metro wired up to the breakout using the STEMMA QT connector. A potentiometer is connected to analog input 0 (A0):

- Board 5V to breakout VIN (red wire)
- Board GND to breakout GND (black wire)
- Board SCL to breakout SCL (yellow wire)
- Board SDA to breakout SDA (blue wire)
- Breakout VIN to potentiometer positive (red wire)
- Breakout A0 to potentiometer wiper (green wire)
- Breakout GND to potentiometer negative (black wire)

Here is an Adafruit Metro wired up using a solderless breadboard:
Board 5V to breakout VIN (red wire)
Board GND to breakout GND (black wire)
Board SCL to breakout SCL (yellow wire)
Board SDA to breakout SDA (blue wire)
Breakout VIN to potentiometer positive (red wire)
Breakout A0 to potentiometer wiper (green wire)
Breakout GND to potentiometer negative (black wire)

Library Installation

You can install the Adafruit_ADS7830 library for Arduino using the Library Manager in the Arduino IDE.

Click the Manage Libraries ... menu item, search for Adafruit_ADS7830, and select the Adafruit ADS7830 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!

Example Code

```c
#include <Wire.h>
#include <Adafruit_ADS7830.h>

Adafruit_ADS7830 ad7830;

void setup() {
  Serial.begin(115200);
  while (!Serial) delay(10); // Wait for console to open

  Serial.println("Adafruit ADS7830 Test by Limor Fried/Ladyada");

  // Possible arguments to begin():
  // begin(); // Uses default I2C address 0x48 and Wire
  // begin(0x49); // Uses I2C address 0x49 and Wire
  // begin(0x48, &Wire1); // Uses I2C address 0x48 and Wire1

  if (!ad7830.begin()) {
    Serial.println("Failed to initialize ADS7830!");
    while (1);
  }
}

void loop() {
  for (uint8_t ch = 0; ch <= 7; ch++) {
    uint8_t value = ad7830.readADCsingle(ch);
    Serial.print(value);
    if (ch < 7) {
      Serial.print(",	");
    }
  }
  Serial.println();
  delay(100);
}
```
Upload the sketch to your board and open up the Serial Monitor (Tools -> Serial Monitor) at 115200 baud. You'll see the ADS7830 recognized over I2C. As you turn the potentiometer attached to A0, you'll see the values change.

The remaining 7 channels are 'floating' - not connected to anything, so you will see varying values between 0 and 255. If you want the 7 other channels to not 'float' tie them to a light pulldown (10-100K ohm) to ground.

---

**Arduino Docs**

[Arduino Docs](#)

**Downloads**

**Files**

- ADS7830 Datasheet
- EagleCAD PCB Files on GitHub
- Fritzing object in the Adafruit Fritzing Library
Schematic and Fab Print