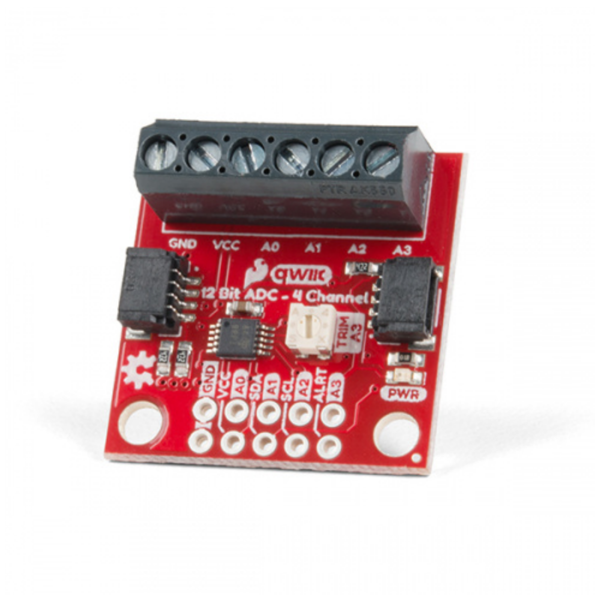
Qwiic 12-Bit ADC Hookup Guide

Introduction

An analog to digital converter (ADC) is very useful tool for converting an analog voltage to a digital signal that can be read by a microcontroller. The ability to converting from analog to digital interfaces allows users to use electronics to interface to interact with the physical world.



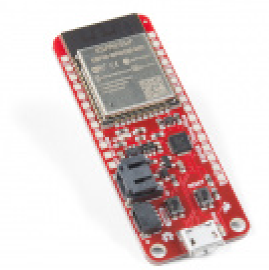
SparkFun Qwiic 12 Bit ADC - 4 Channel (ADS1015)
DEV-15334

The SparkFun Qwiic (12-bit) ADC provides four channels of I^2C controlled ADC input to your Qwiic enabled project. These four channels can be used as single-ended inputs, or in pairs for differential inputs. The ADS1015 uses its own internal voltage reference for measurements, but the ground and 3.3V power are also available on the pin outs for users.

Note: The maximum resolution of the converter is **12-bits** in differential mode and **11-bits** for single-ended inputs. Step sizes range from 125µV per count to 3mV per count depending on the full-scale range (FSR) setting.

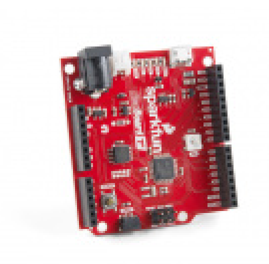
Required Materials

The SparkFun Qwiic ADC does need a few additional items for you to get started; a Qwiic enabled microcontroller, a Qwiic cable, and jewelry/precision screwdrivers (with **1.5mm** and **2.5mm flathead bits**). You may already have a few of these items, so feel free to modify your cart based on your needs.

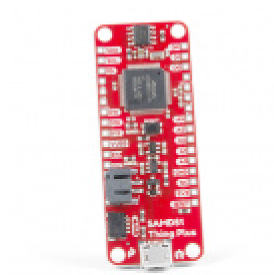


SparkFun Thing Plus - ESP32 WROOM
WRL-14689

SparkFun RedBoard Qwiic
DEV-15123



SparkFun RedBoard Turbo - SAMD21 Development Board
DEV-14812



SparkFun Thing Plus - SAMD51
DEV-14713

SparkFun Qwiic Cable Kit
KIT-15081

Magnetic Screwdriver Set (20 Piece)
TOL-15003

Suggested Reading:

If you're unfamiliar with analog to digital converters, jumper pads, or I^2C be sure to checkout some of these foundational tutorials.

Analog to Digital Conversion
The world is analog. Use analog to digital conversion to help digital devices interpret the world.

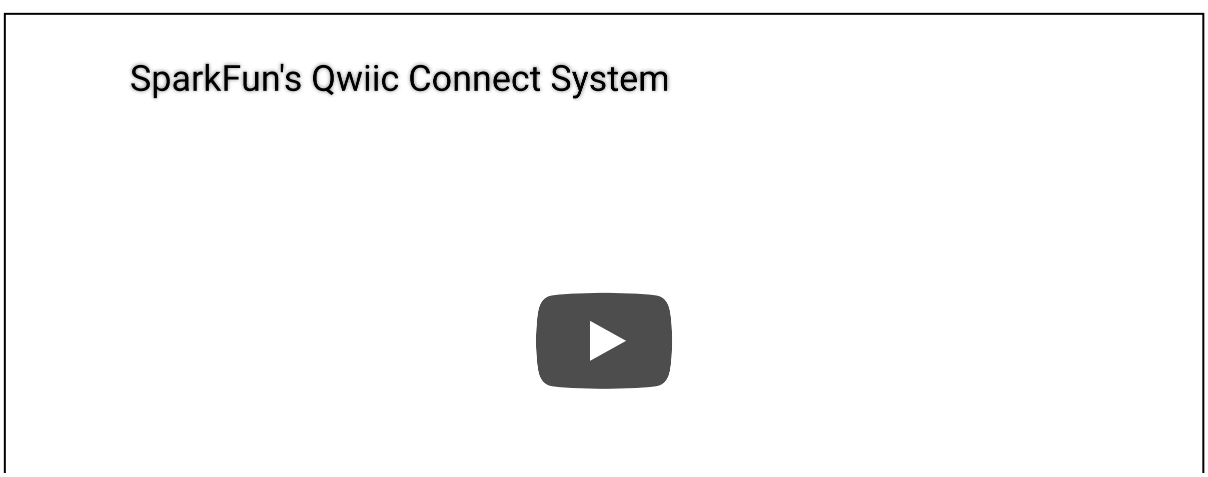
I2C
An introduction to I2C, one of the main embedded communications protocols in use today.

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.

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.

ADDITIONAL READING ON ANALOG CIRCUITS

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



Hardware Overview

Power

There is a power status LED to help make sure that your Qwiic ADC is getting power. You can power the board either through the *polarized* **Qwiic connector** system or the breakout pins (**3.3V** and **GND**) provided. This Qwiic system is meant to use **3.3V**, be sure that you are **NOT** using another voltage when using the Qwiic system.

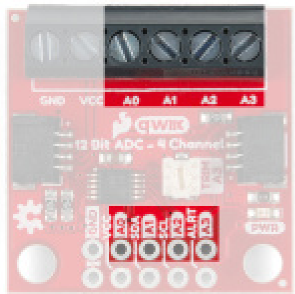
Annotated image of power LED along with VCC and GND connections.

(**If you want to remove the power LED to conserve power, you will need to de-solder or hotair rework the LED.*)

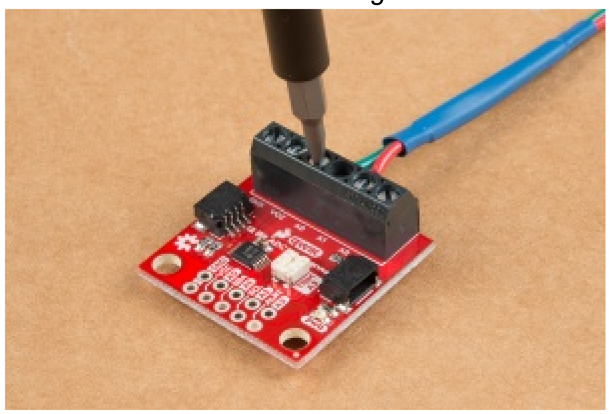
Analog Inputs

> ⚡**Caution:** The analog input voltage range is **(GND - .3V)** to **(V_{DD} + .3V)**; anything slightly higher or lower and you will damage the ADC chip. If you are using the Qwiic system, this is approximately **-.3V to 3.6V** in reference to the GND pin. If the voltages on the input pins can potentially violate these conditions, use external Schottky diodes and series resistors to limit the input current to safe values (as mentioned in the datasheet).

There are four input measurement channels for the ADS1015, labeled AIN_{0-3}, accessible through the screw pin terminals shown below. With the Qwiic system, the absolute minimum and maximum voltage for these inputs is **-.3V** and **3.6V**, respectively.



Annotated image of input connections on board.
Click to enlarge.



Example of attaching inputs through screw terminals. Click to enlarge.

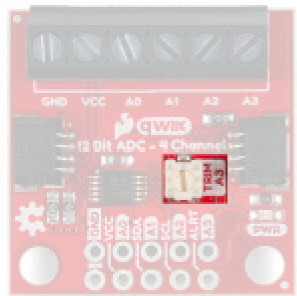
Depending on the settings for the multiplexer (MUX) in the ADS1015, the analog voltage readings (or conversions on the ADC) will be either for 4 single-ended inputs or 2 differential pairs.

Pin Label	Configuration Options
AIN_0	Single-Ended Differential Inputs: AIN1 or AIN3 w/ Programmable Comparator
AIN_1	Single-Ended Differential Ref. w/ Programmable Comparator Differential Input: AIN3 w/ Programmable Comparator
AIN_2	Single-Ended Differential Input: AIN3 w/ Programmable Comparator
AIN_3	Single-Ended Differential Ref. w/ Programmable Comparator On-board 10kΩ Potentiometer

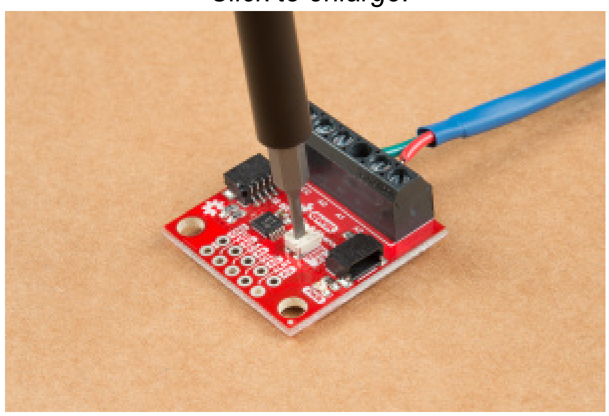
Note: For high accuracy measurements, input impedance should be taken into consideration. Additionally, any noise from the power supply to the ADC will be passed on to the output measurements.

Potentiometer

In addition, there is an onboard 10kΩ potentiometer, attached to AIN_3 that is used in the examples and can be used for testing. The potentiometer is connected directly to **3.3V** and **GND** so the voltage range will always be defined by the power input to the board.



Annotated image of potentiometer on board.
Click to enlarge.



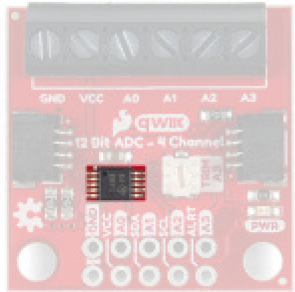
*For the potentiometer, we recommend using **1.5mm** flathead precision screwdriver. Click to enlarge.*

When using AIN_3 as the negative/reference input be sure to cut the jumper to remove the potentiometer for accurate differential readings (see Jumper section below).

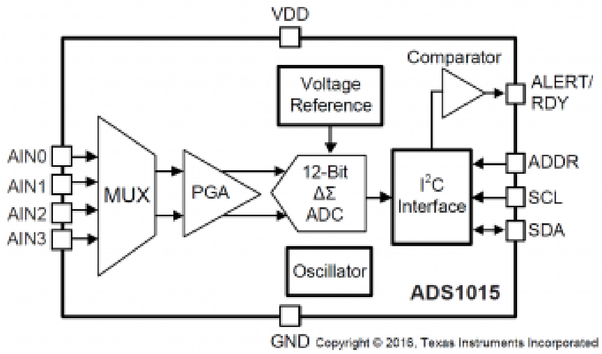
ADS1015 ADC

The ADS1015 ADC is a low-power 12-bit analog-to-digital converter (ADC), which includes a built-in integrated voltage reference and oscillator. At the core of its operation, the ADC uses a switched-capactior input stage and a delta-sigma (ΔΣ) modulator to determine the differential between AIN_P (positive analog input) and AIN_N (negative analog input). Once the conversion is completed, the digital output is accessible over the I^2C bus from the internal *conversion register*.

Interested in how ΔΣ ADCs work? Here are TI's application notes: How delta-sigma ADCs work, Part 1 and Part 2.



Annotated image of ADS1015 on board. Click to enlarge.



Functional block diagram from datasheet. Click for more details.

Note: The ADS1015 has an integrated voltage reference; **an external reference voltage cannot be used**.

The ADS1015 is a powerful tool with multiple configuration settings, set by the **Config Register**, for the analog voltage readings (or *conversions*). In the following sections, we will cover the general operation of the ADS1015. For exact details of the various configuration settings, please refer to the manufacturer datasheet. The operational characteristics of the ADS1015 are summarized in the table below.

Characteristic	**Description**
Operating Voltage (V_{DD})	2.0V to 5.5V (**Default on Qwiic System: 3.3V**)
Operating Temperature	-40°C to 125°C
Operation Modes	Single-Shot (**Default**), Continuous-Conversion, and Duty Cycling
Analog Inputs	Measurement Type: Single-Ended or Differential Input Voltage Range: **GND** to $\mathbf{V_{DD}}$ (*see* ***Caution*** *note, below*) Maximum Voltage Measurement: *Smallest of* $\mathbf{V_{DD}}$ or **FSR** Full Scale Range (FSR): ±.256V to ±6.114V (**Default: 2.048V**)
Resolution	12-bit (Differential) or 11-bit (Single-Ended) LSB size: 0.125mV - 3mV (**Default: 1 mV** *based on FSR*)
Sample Rate	128 Hz to 3.3 kHz (**Default: 1600SPS**)

Current Consumption (*Typical*)	Operating: 150µA to 200µA Power-Down: 0.5µA to 2µA
I^2C Address	**0x48** (**Default**), 0x49, 0x4A, or 0x4B

Caution: The absolute, analog input voltage range is **(GND - .3V) to (V_{DD} + .3V)**; anything slightly higher/lower may damage the ADC chip. If the voltages on the input pins can potentially violate these conditions, as specified by the datasheet, use external Schottky diodes and series resistors to limit the input current to safe values.

Operational Modes

The ADS1015 has 2 different conversion modes: *single-shot* and *continuous-conversion* with the ability to support duty cycling. Through these modes, the ADS1015 is able to optimize its performance between low power consumption and high data rates.

Single-Shot

By default, the ADS1015 operates in single-shot mode. In single-shot mode, the ADC only powers up for ~25µs to convert and store the analog voltage measurement in the conversion register before powering down. The ADS1015 only powers up again for data retrieval. The power consumption in this configuration is the lowest, but it is dependent on the frequency at which data is converted and read.

Duty Cycling

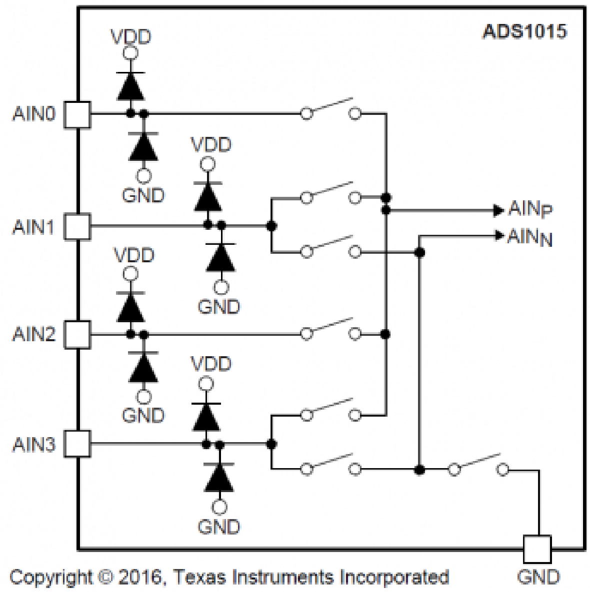
In single-shot mode, the ADS1015 can be duty cycled to periodically request high data rate readings. This emulates an intermediary configuration between the low power consumption of the single-shot mode and the high data rates of the continuous-conversion mode.

Continuous-Conversion

In this mode, the ADS1015 continuously performs conversions on analog voltage measurements and places the data in the conversion register. If the configuration settings are changed in the middle of the conversion process, the settings take effect once the current process is completed.

Input Multiplexer (MUX)

There are four input measurement channels for the ADS1015, labeled AIN_{0-3}. The input multiplexer controls which of those channels operates as the AIN_P (positive analog input) and AIN_N (negative analog input) to the ADC. Depending on the input MUX configuration, the voltage measurements will be either on single-ended inputs or as differential pairs.



Operational digram of multiplexer from datasheet.

There are 8 MUX configurations to designate the analog voltage inputs to the ADC of the ADS1015, shown in the table below. The default configuration of the ADS1015 uses inputs AIN_0 and AIN_1 as highlighted in bold in the table below.

Inputs	**MUX Configurations**							
	Differential				Single-Ended			
AIN_p	$\mathbf{AIN_0}$	AIN_0	AIN_1	AIN_2	AIN_0	AIN_1	AIN_2	AIN_3
AIN_N	$\mathbf{AIN_1}$	AIN_3	AIN_3	AIN_3	GND	GND	GND	GND

Programmable Gain Amplifier

A programmable gain amplifier (PGA) is implemented before the ΔΣ ADC. The ADS1015 has 6 programmable gain settings, which are expressed in the full-scale range (FSR) of the ADC scaling. The maximum analog measurement is then defined by the smaller of the FSR or V_{DD}. By default, the ADS1015 has a resolution of **1mV** by using an FSR of **±2.048V** as highlighted in bold in the table below.

Gain:	16	8	4	**2**	1	$^2/_3$
Resolution (LSB):	0.125 mV	0.25 mV	0.5 mV	**1 mV**	2 mV	3 mV
FSR (12-bit):	±256 mV	±512 mV	±1.024 V	**±2.048 V**	±4.048 V	±6.144 V

Analog-to-Digital Conversion

Although, it is listed as a **12-bit ADC**, the ADS1015 operates as an **11-bit ADC** when used with single-ended (individual) inputs. The 12th bit only comes into play in differential mode, as a sign (+ or -) indicator for the digital output. This allows the digital output to represent the full positive and negative range of the FSR (*see table and figure below*).

Table 3. Input Signal Versus Ideal Output Code

INPUT SIGNAL $V_{IN} = (V_{AINP} - V_{AINN})$	IDEAL OUTPUT CODE[1][1]
$\geq +FS\ (2^{11} - 1)/2^{11}$	7FF0h
$+FS/2^{11}$	0010h
0	0000h
$-FS/2^{11}$	FFF0h
$\leq -FS$	8000h

(1) Excludes the effects of noise, INL, offset, and gain errors.

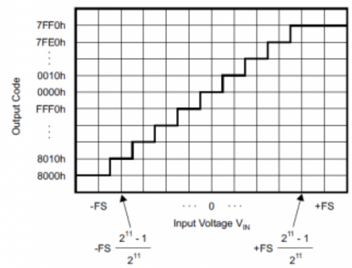


Figure 18. Code Transition Diagram

NOTE

Single-ended signal measurements, where V_{AINN} = 0 V and V_{AINP} = 0 V to +FS, only use the positive code range from 0000h to 7FF0h. However, because of device offset, the ADS101x can still output negative codes in case V_{AINP} is close to 0 V.

Explanation of how data output and sign bit work from datasheet.

Limitation of ADS1015

Since the ADS1015 only uses an internal reference voltage, the FSR is to be defined by the design of the 12-bit ADC:

FSR = LSB x 2^{12}
where, the LSB = 0.125, 0.25, 0.5, 1, 2, or 3 mV.

Due to the configuration options, it is difficult to make full use of the full-scale range of the ADS1015 with common (*useful*) voltages. See the examples below for a more detailed explanation:

If the FSR = 2.048V and V_{DD} = 3.3V:

- Resolution of the digital data: 1mV (*defined by FSR*)
- Input voltage range: 0-3.3V (*defined by V_{DD}*)
- Data Range: 0000h-7FF0h (HEX) or 0-2.048V
  The range of the input voltage that can be read by the ADC is limited by FSR. Any voltage higher than the FSR (but less than V_{DD}) reads the same maximum value in the digital output because the FSR is maxed out. In this case, you are maximizing the resolution (use of the data output), but not the full, allowable range of analog input (0-3.3V).

If the FSR = 4.096V and V_{DD} = 3.3V:

- Resolution of the digital data: 2mV (*defined by FSR*)
- Input voltage range: 0-3.3V (*defined by V_{DD}*)
- Data Range: 0000h-0672h (HEX) or 0-3.3V
  The data range is limited by V_{DD}, any higher input voltage will continue to have data up to the electrical specifications of the ADS1015 ~V_{DD} + 0.3V, where the IC gets damaged. In this case, the input voltage is being maximized to the electrical specification (0-3.3V). However, this means you are effectively only using 80% of the full resolution of the ADC (i.e. a 10.8-bit ADC at 3.3V).

Data Rate & Conversion Time

The ADS1015 offers 7 selectable output data rates of 128 SPS, 250 SPS, 490 SPS, 920 SPS, 1600 SPS, 2400 SPS, or 3300 SPS. Conversions for the ADS1015 settle within a single cycle; thus, the conversion time is equal to 1/DR.

Programmable Digital Comparator

The ADS1015 features a programmable digital comparator that can trigger the ALERT/RDY pin to indicate when conversion data is ready. By default, the comparator is disabled. However, when activated, the comparator operates in either a traditional or window mode. The upper and lower thresholds for the comparator are set by the last two registers on the ADS1015; while, the polarity (active low/high), latching or non-latching, operational mode, and enabling of the ALERT/RDY pin are set in the Config register.

Qwiic or I^2C

I^2C Address

The ADS1015 has 4 available I^2C addresses, which are set by the address pin, ADDR. On the Qwiic ADC, the default slave address of the ADS1015 is **0x48** (HEX) of 7-bit addressing, following I^2C protocol. The ADS1015 does have an additional general call address that can be used to reset all internal registers and power down the ADS1015 (see datasheet).

Default I^2C Slave Address: 0x48

I^2C Registers

The ADS1015 has four 16-bit registers, which are accessible through the I^2C bus using the Address Pointer register. The Address Pointer register is an 8-bit byte that is written immediately after the slave address byte, low R/W bit.

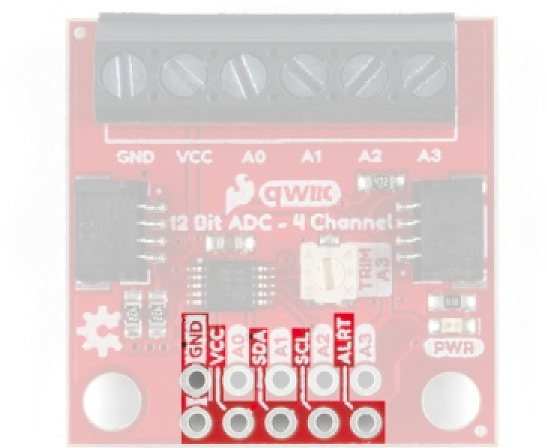
Address	Description
N/A	Address Pointer Register (8-bit): Used to grant R/W access to the four available registers on the ADS1015.
0x00	Conversion Register (16-bit): Contains result of last conversion (i.e. measurement).
0x01	Config Register (16-bit): Used the for configuration setting of the ADS1015.
0x02	Low Threshold Register (16-bit): Low threshold value for digital comparator.
0x03	High Threshold Register (16-bit): High threshold value for digital comparator.

Connections

The simplest way to use the Qwiic ADC is through the Qwiic connect system. The connectors are polarized for the I^2C connection and power. (**They are tied to their corresponding breakout pins.*)

Annotated image of the Qwiic connectors.

However, the board also provides five labeled breakout pins. You can connect these lines to the I^2C bus of your microcontroller and power pins (**3.3V** and **GND**), if it doesn't have a Qwiic connector. The ALERT/RDY pin is broken out to use for triggered events (*requires pull-up resistor*).



Annotated image of the breakout pins.

Pin Label	Pin Function	Input/Output	Notes
3.3V	Power Supply	Input	**3.3V** on Qwiic system (*should be stable*)
GND	Ground	Input	Ground and Single-Ended Reference Voltage for ADC.
SDA	I^2C Data Signal	Bi-directional	Bi-directional data line. Voltage should not exceed power supply (e.g. 3.3V).
SCL	I^2C Clock Signal	Input	Master-controlled clock signal. Voltage should not exceed power supply (e.g. 3.3V).

ALERT/RDY	Alert/Interrupt	Output	Comparator

Jumpers

Caution: Be careful when cutting traces, as not to unintentionally cut other traces.

There are jumpers on the board allowing the user to select between different I^2C addresses, to remove the pull up resistors from the I^2C pins, and to disconnect the potentiometer on AIN_3. Not sure how to modify a jumper? Read here!

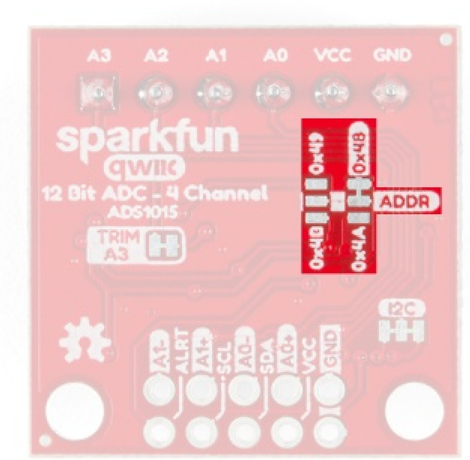
I^2C Address

The ADS1015 has four available I^2C addresses, which can be configured by the jumpers on the back of the board. The **address selection pin** is connected to the **center pad of the jumpers**, the below table shows the addresses available when the address selection pin is tied to each of the 4 available pads.

Pin	GND	VCC	SDA	SCL
Address	**0x48 (Default)**	0x49	0x4A	0x4B

Note: Make sure this jumper is only shorted on one of the four available pads. There is a pullup jumper on the 3.3V pad, but to help prevent a shorts if multiple pads are accidentally bridged.

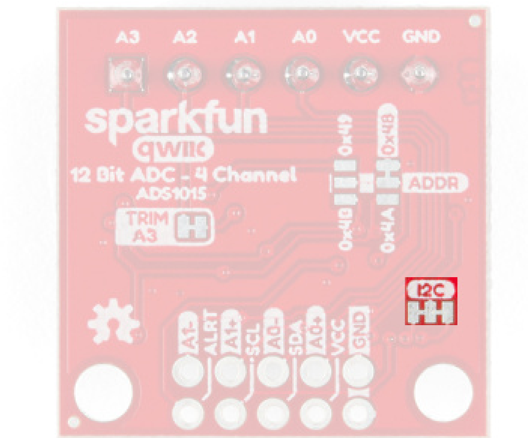
The **address selection pin**, by default is tied to **GND** on the PCB. Cutting the trace and bridging the I^2C address jumper to another pad changes the slave address from I^2C Jumper Default: **0x48**. The location of the jumpers is shown in the image below.



Annotated image of the I^2C address jumper.

I^2C Pull-up Jumper

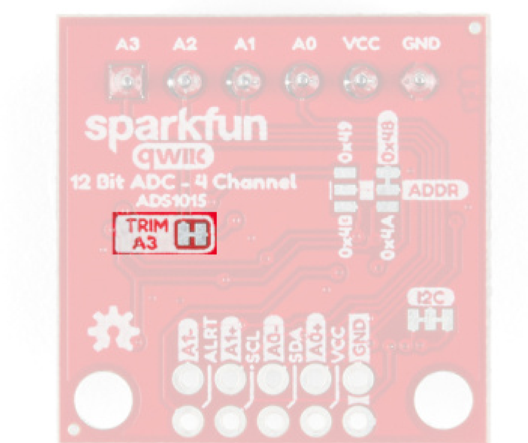
Cutting the **I^{2}C jumper** will remove the **2.2 kΩ** pull-up resistors from the I^{2}C bus. If you have multiple devices with pull-up resistors on your I^{2}C bus, you may want to cut these jumpers. (*When there are multiple devices on the bus with pull-up resistors, the equivalent parallel resistance may create too strong of a pull-up for the bus to operate correctly.*)



Annotated image of the I^{2}C pull-up jumper.

Potentiometer

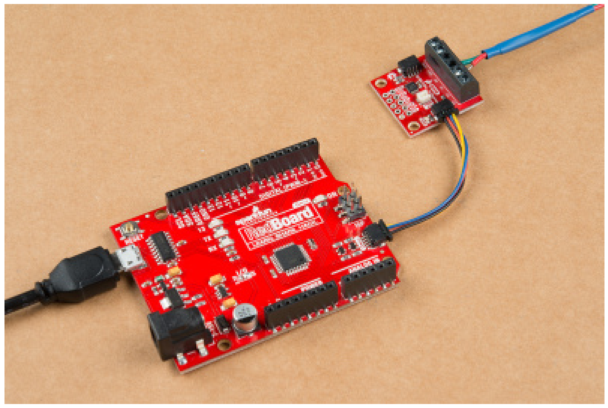
Cutting the **potentiometer jumper** will disconnect **10kΩ potentiometer** from the **AIN$_3$** input pin.



Annotated image of the potentiometer jumper.

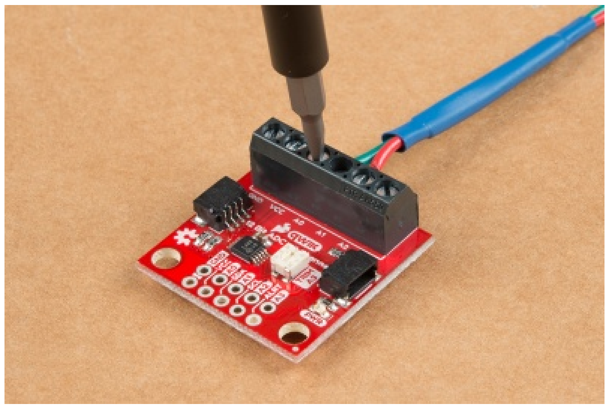
Hardware Assembly

With the Qwiic connector system, assembling the hardware is fairly simple. For the examples below, all you need to do is connect your Qwiic ADC to Qwiic enabled microcontroller with a Qwiic cable. Otherwise, you can use the I^{2}C pins, if you don't have a Qwiic connector on your microcontroller board. Just be aware of your input voltage and any logic level shifting you may need to do.



Example setup with RedBoard Qwiic.

Additionally, you can connect your input voltages to the available inputs on the screw terminals and or use the breakout pins on the board. Make sure the connections are fully inserted and that you are ground looping your inputs. Ground looping can be done by connecting the ground of your input to the ground of the ADC.



Example of attaching inputs through screw terminals.

Arduino Library

> **Note:** This example assumes you are using the latest version of the Arduino IDE on your desktop. If this is your first time using Arduino, please review our tutorial on installing the Arduino IDE. If you have not previously installed an Arduino library, please check out our installation guide.

We've written a library to easily get setup and take readings from the Qwiic 12-bit ADC. However, before we jump into getting data from the sensor, let's take a closer look at the available functions in the library. You can install this library through the Arduino Library Manager. Search for **SparkFun ADS1015 Arduino Library** and you should be able to install the latest version. If you prefer manually downloading the libraries from the GitHub repository, you can grab them here:

DOWNLOAD THE SPARKFUN ADS1015 ARDUINO LIBRARY (ZIP)

Let's get started by looking at the functions that set up the Qwiic 12-bit ADC.

Setup and Settings

`.begin()` or `.begin(i2caddr, Wire)`
Creates a connection to an I^2C device over the I^2C bus using the default or specified I^2C address.

Input: *i2caddr*

Unassigned 8-bit integer for device address. If not defined, the library will use the default I²C address stored in the I²C library (0x48). Other available addresses are 0x49, 0x4A, 0x4B (set by a 4-way solder jumper on the bottom side of the board).

Input: *Wire*

Wire port assignment. If not defined, the library will use the default Wire port. Note, this is only available to change on boards that in fact have multiple Wire Ports (like the Teensy).

Output: Boolean

True- Connected to I²C device on the default (or specified) address.

False- No device found or connected.

`.isConnected()`

Checks to see if a device over the I²C bus is connected. Returns boolean true or false depending on if the slave device has correctly ack'd to an I²C request.

Output: Boolean

True- Device present on the default (or specified) address.

False- No device found or connected.

`.setMode(uint16_t mode)`

Set the read mode of the sensor (continuous or single-shot).

Input: Unassigned 16-bit integer

0 - Continuous.

1 - Single-shot.

`.getMode()`

Returns which read mode you are currently in: continuous (0) or single-shot (1).

Output: Unassigned 16-bit integer

0 - Continuous.

1 - Single-shot.

`.setGain(uint16_t gain)`

Sets the gain of the programmable gain amplifier (PGA) inside the ADS1015. Note, this gain amplifier is located just before the ADC converter, so it greatly effects your readings and can cause your readings to "max out". Pass in one of the following `#define "variables"` from the library to easily set the gain.

Input: Unassigned 16-bit integer

ADS1015_CONFIG_PGA_TWOTHIRDS : ± 6.144V

ADS1015_CONFIG_PGA_1 : ± 4.096V

ADS1015_CONFIG_PGA_2 : ± 2.048V (default)

ADS1015_CONFIG_PGA_4 : ± 1.024V

ADS1015_CONFIG_PGA_8 : ± 0.512V

ADS1015_CONFIG_PGA_16 : ± 0.256V

`.getGain()`

Returns the gain of the programmable gain amplifier (PGA) inside the ADS1015. This returns a 16-bit hex value. The values and their corresponding gains and voltage ranges are as follows.

Output: Unassigned 16-bit integer

0x0000 - gain:2/3, input range:± 6.144V

0X0200 - gain:1, input range: ± 4.096V

0X0400 - gain:2, input range: ± 2.048V

0X0600 - gain:4, input range: ± 1.024V
0X0800 - gain:8, input range: ± 0.512V
0X0A00 - gain:16, input range: ± 0.256V

`.setSampleRate(uint16_t sampleRate)`
Sets the sample rate of the ADS1015. Use the following #define variables.

Input: Unassigned 16-bit integer
ADS1015_CONFIG_RATE_128HZ
ADS1015_CONFIG_RATE_250HZ
ADS1015_CONFIG_RATE_490HZ
ADS1015_CONFIG_RATE_920HZ
ADS1015_CONFIG_RATE_1600HZ (default)
ADS1015_CONFIG_RATE_2400HZ
ADS1015_CONFIG_RATE_3300HZ

`.setComparatorSingleEnded(uint8_t channel, int16_t threshold)`
Sets up a single ended comparator that will effect the ALERT pin on the ADS1015 (active LOW by default) when a reading above the threshold is read. Note, you must call .getLastConversionResults() for this to reset. See Example5_Alert.ino in the library for more info.

Input: *channel*
Unassigned 8-bit integer. Available values are 0, 1, 2, and 3 (for the corresponding A0, A1, A2, and A3).

Input: *threshold*
Signed 16-bit integer. Possible values are decimal 0-2047. 1000 is about 3V.

`.getSampleRate()`
Returns the sample rate of the ADS1015 as an unassigned 16-bit integer value. The values and their corresponding sample rates are as follows.

Output: Unassigned 16-bit integer
0x0000 : 128HZ
0X0200 : 250HZ
0X0400 : 490HZ
0X0600 : 920HZ
0X0800 : 1600HZ
0X0A00 : 2400HZ
0X0A00 : 3300HZ

These functions are used primarily for use with the Qwiic Flex Glove Controller (which uses the same Arduino library). Please see the Qwiic Flex Glove Controller hookup guide for more information.

- `.getAnalogData()`
- `.getScaledAnalogData()`
- `.calibrate()`
- `.setCalibration()`
- `.getCalibration()`
- `.resetCalibration()`

Readings

`.getSingleEnded(uint8_t channel);`
Returns the single ended analog value from the sensor on the specified channel.

Input: *channel*

Unasssigned 8-bit integer for channel you'd like to read. Available channels are 0, 1, 2, 3. These correspond to the channels on the board labeled A0, A1, A2, and A3.

Output: Unasssigned 16-bit integer

This return values from decimal 0 - 2047 (or HEX 0x0000 - 0x07FF). Note, this is only 11 bits of precision on a single-ended input. 12-bit resolution is only available on differential inputs.

`.getDifferential()` or **`.getDifferential(uint8_t channel)`**
Returns a signed differential analog value from the sensor on the specified pairs of channels. If no argument is passed, then it will use the default pair of channels (A0 and A1).

Input: *channel*

ADS1015_CONFIG_MUX_DIFF_P0_N1 (default)
ADS1015_CONFIG_MUX_DIFF_P0_N3
ADS1015_CONFIG_MUX_DIFF_P1_N3
ADS1015_CONFIG_MUX_DIFF_P2_N3

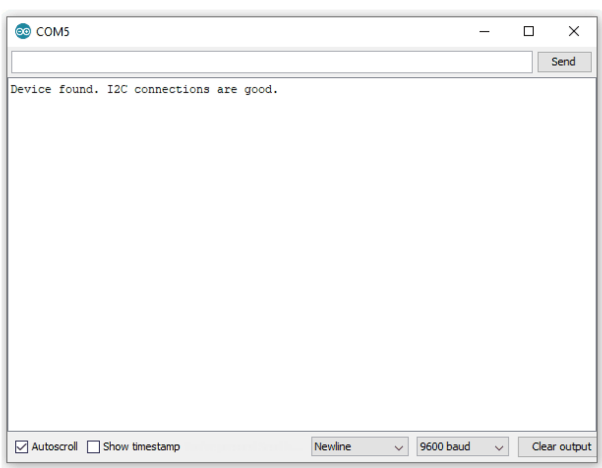
Output: Signed 16-bit integer

This return signed values from decimal -2047 up to +2047.

Arduino Examples

Example 1: Read Basic

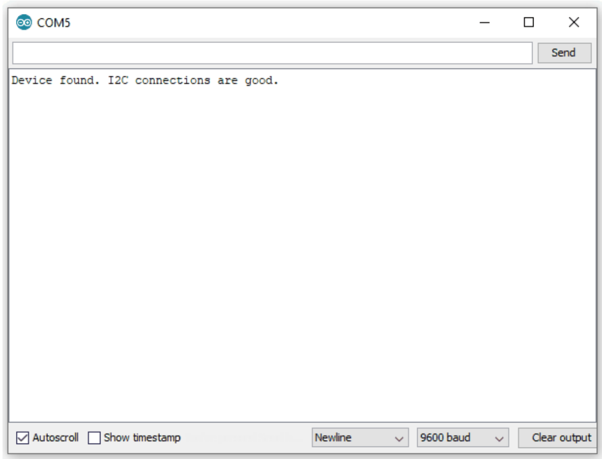
The code for **Example1_ReadBasic.ino** connects to the Qwiic 12-bit ADC and prints out the value of channel A3 over the Serial Monitor at `BAUD 9600`. Below is a sample readout from the Serial Monitor rotating the onboard trimpot from MIN (all the way CCW) to MAX (all the way CW), and then back to MIN again. Notice how it "maxes out" at 2047, this is because the trimpot is inputing voltages from 0 to 3.3V. And with the default gain of 2 the input range of the ADC is only up to 2.048V.



Serial Monitor readout for Example 1.

Example 2: Change Gain

The code for **Example2_ChangeGain.ino** operates exactly like the **Example1_ReadBasic.ino**, except the gain is set to 1. This means that it can now accept voltages from 0 to 4.096V with a resolution of 2mV. Below is a sample readout. Notice how when rotating the trimpot (0-3.3V), I can only get the reading up to 1651. This is because we are not reaching the top of the ADC's range (at this gain of 1).

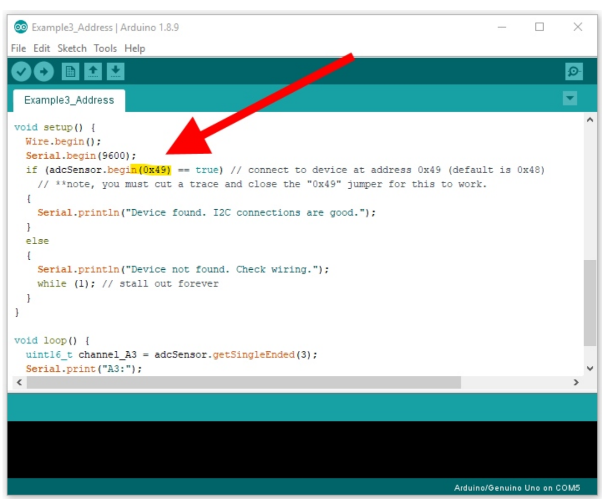


Serial Monitor readout for Example 2.

Example 3: Address

> **Caution:** This example will not work if multiple I^2C address jumpers are closed. Be sure to only close one of them.

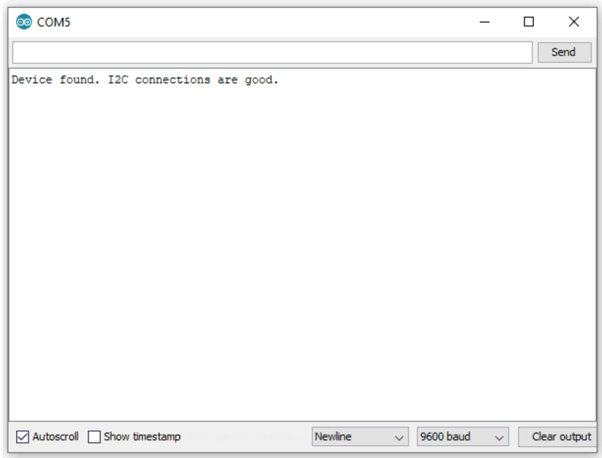
The code for **Example3_Address.ino** connects to your Qwiic ADC on a specific address. You must modify the address jumper located on the bottom side of the board. Cut the trace connecting the default jumper, and then close the jumper that is labeled with the desired address. **Example3_Address.ino** works the same as **Example1_ReadBasic.ino**, however it passes an address argument to the `.begin()` function like so: `adcSensor.begin(0x49);`



Code screenshot with highlight on address argument for Example 3.

Example 4: Differential

The forth example, **Example4_Differential.ino**, shows how to read the ADC in differential mode using channels A0 and A1. For this example, I am using two trimpots (tied to GND and 3.3V) to supply voltage sources to A0 and A1. I have the A0 trimpot set to 2.5V, and it will stay there as a reference. In the readout below, I am rotating the A1 trimpot from 0V all the way up to 3.3V. You can see that I actually start at a negative value, because my rotating trimpot is actually sitting at 0V (or negative 2.5 Volts in relation to the other input).

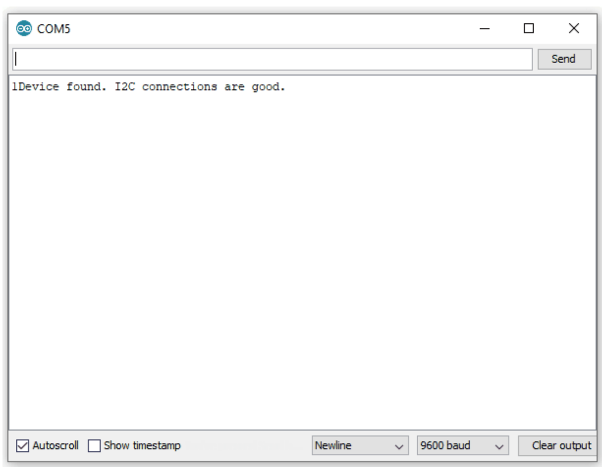


Serial Monitor readout for Example 4.

Example 5: Alert

> **Note:** This example is requires soldering. A wire must be soldered to the ALERT pin and connected to a digital pin on your microcontroller. As written, the example assumes you are using a Arduino UNO based board, like our RedBoard. The sketch designates Pin 2 as the read pin.

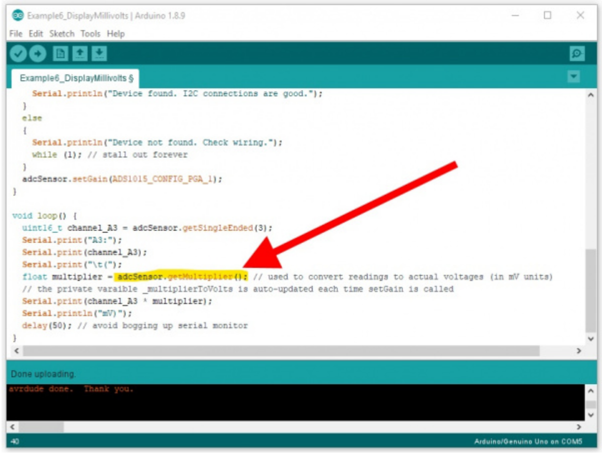
The code for **Example5_Alert.ino** sets up a single ended comparator in the ADS1015 chip. It is watching A3, and it if it sees a value about the threshold of 1000 (about 3V), it will drop the ALERT pin. The serial monitor below shows state of the alert pin (as read from D2) and the value from the ADC. This shows that when you cross 1000, then the alert pin changes accordingly.



Serial Monitor readout for Example 5.

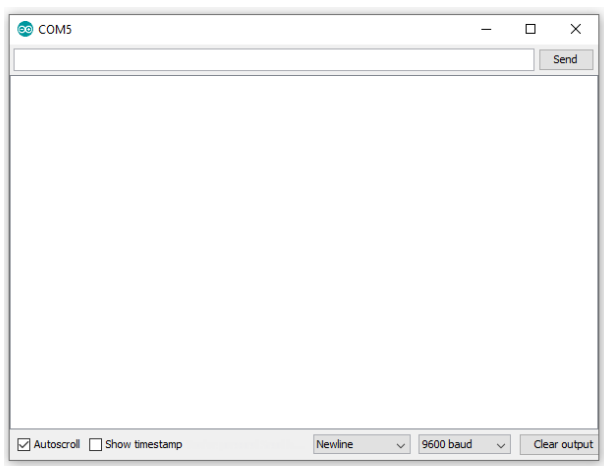
Example 6: Display MilliVolts

The last example, **Example6_DisplayMillivolts.ino**, will work much like **Example1_ReadBasic.ino**, however it also will print out the reading in millivolts. To convert from the raw reading values from the ADC, you must use a multiplier to get to millivolts. This changes depending on what gain setting you are using. For ease of use, we have included a function in the library that looks at your current gain settings, and then returns the correct multiplier you need. It is named `.getMultiplier()` .



Code highlight of .getMultiplier() in Example 6.

In the following serial monitor readout, you can see me turning the on-board trimpot on A3 (from 0V up to 3.3V) as it prints out both the return values and millivolts.



Serial Monitor readout for Example 6.

Troubleshooting

Here are a few tips for troubleshooting this device.

Power

If you are not using the Qwiic system, make sure your supply voltage is within the electrical specifications of the ADS1015.

Connections

Make sure your inputs have proper contacts and that you are grounding looping them. If you are using the inputs in differential mode, be sure to cut the potentiometer jumper for accurate readings.

Inputs Voltages

If the input voltage was outside **(GND - .3)V - (V_{DD} + .3) V**, it is very likely that the ADS1015 was damaged and you probably need a new board.

Not Getting a Full 12-bits

If you are expecting to use the full resolution of the 12-bit for the ADC, but you are having issues; check out the brief explanation of the data in the hardware section. The ADS1015 is effectively a 10.6-bit ADC when powered at 3.3V and using a 4.096 FSR.

No Available Devices in Examples

Usually an issue in attempt to connect to the Qwiic ADC.

- Check the I^{2}C address jumper on the board and the defined I^{2}C address in the code.
- Check Qwiic cables and make sure they are fully inserted. Try a another cable if possible.
- If you aren't using the Qwiic system, double check your logic levels. You should be using a logic level converter between different logic levels.
- Try the I^{2}C scanner code to search for devices on the I^{2}C bus. You may have changed the I^{2}C address (code or hardware) and forgotten.

If you still have questions or issues with this product, please create a post on our forum.

Resources and Going Further

For more product information, check out the resources below:

- Schematic (PDF)
- Eagle Files (ZIP)
- ADS1015 Datasheet
- SparkFun ADS1015 Arduino Library
- GitHub Product Repo

Need some inspiration for your next project? Check out some of these other Qwiic product tutorials:

Qwiic Adapter Hookup Guide

Get started with your Qwiic adapter board. This adapter breaks out the I2C pins from the Qwiic connectors to pins that you can easily solder with your favorite I2C enabled device.

Qwiic HAT for Raspberry Pi Hookup Guide

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

RedBoard Edge Hookup Guide
The RedBoard Edge is a RedBoard that's been rebuilt around the idea that projects are eventually put into an enclosure to help clean up their look.

Qwiic UV Sensor (VEML6075) Hookup Guide
Learn how to connect your VEML6075 UV Sensor and figure out just when you should put some sunscreen on.