

---

**Adafruit**  
**CIRCUITPYTHON**<sub>A</sub>*DS1X15LibraryDocumentati*  
**Release 1.0**

**Carter Nelson**

**Aug 06, 2018**



---

## Contents

---

<b>1</b>	<b>Dependencies</b>	<b>3</b>
<b>2</b>	<b>Usage Example</b>	<b>5</b>
2.1	Single Ended . . . . .	5
2.2	Differential . . . . .	5
<b>3</b>	<b>Contributing</b>	<b>7</b>
<b>4</b>	<b>Building locally</b>	<b>9</b>
4.1	Sphinx documentation . . . . .	9
<b>5</b>	<b>Table of Contents</b>	<b>11</b>
5.1	Simple test . . . . .	11
5.2	adafruit_ads1x15 . . . . .	12
5.3	adafruit_ads1x15.differential . . . . .	13
5.4	adafruit_ads1x15.single_ended . . . . .	13
<b>6</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>



Support for the ADS1x15 series of analog-to-digital converters. Available in 12-bit (ADS1015) and 16-bit (ADS1115) versions.



# CHAPTER 1

---

## Dependencies

---

This driver depends on:

- [Adafruit CircuitPython](#)
- [Bus Device](#)

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the [Adafruit library and driver bundle](#).





## CHAPTER 2

---

### Usage Example

---

#### 2.1 Single Ended

```
import board
import busio
from adafruit_ads1x15.single_ended import ADS1015

i2c = busio.I2C(board.SCL, board.SDA)
adc = ADS1015(i2c)
while True:
    # channel 0
    print(adc[0].value, adc[0].volts)
```

#### 2.2 Differential

```
import board
import busio
from adafruit_ads1x15.differential import ADS1015

i2c = busio.I2C(board.SCL, board.SDA)
adc = ADS1015(i2c)
while True:
    # channel 0 - channel 1
    print(adc[(0,1)].value, adc[(0,1)].volts)
```



## CHAPTER 3

---

### Contributing

---

Contributions are welcome! Please read our [Code of Conduct](#) before contributing to help this project stay welcoming.



## CHAPTER 4

---

### Building locally

---

To build this library locally you'll need to install the `circuitpython-build-tools` package.

```
python3 -m venv .env
source .env/bin/activate
pip install circuitpython-build-tools
```

Once installed, make sure you are in the virtual environment:

```
source .env/bin/activate
```

Then run the build:

```
circuitpython-build-bundles --filename_prefix adafruit-circuitpython-ads1x15 --
↳library_location .
```

### 4.1 Sphinx documentation

Sphinx is used to build the documentation based on rST files and comments in the code. First, install dependencies (feel free to reuse the virtual environment from above):

```
python3 -m venv .env
source .env/bin/activate
pip install Sphinx sphinx-rtd-theme
```

Now, once you have the virtual environment activated:

```
cd docs
sphinx-build -E -W -b html . _build/html
```

This will output the documentation to `docs/_build/html`. Open the `index.html` in your browser to view them. It will also (due to `-W`) error out on any warning like Travis will. This is a good way to locally verify it will pass.



## 5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/ads1115\_single\_ended\_simpletest.py

```

1  import time
2  import board
3  import busio
4  from adafruit_ads1x15.single_ended import ADS1115
5
6  # Create the I2C bus
7  i2c = busio.I2C(board.SCL, board.SDA)
8
9  # Create the ADC object using the I2C bus
10 adc = ADS1115(i2c)
11
12 # Print header
13 print("      CHAN 0      CHAN 1      CHAN 2      CHAN 3")
14 print("{:>5}\t{:>5}\t{:>5}\t{:>5}\t{:>5}\t{:>5}\t{:>5}\t{:>5}")
15     .format('raw', 'v', 'raw', 'v', 'raw', 'v', 'raw', 'v'))
16
17 while True:
18     # Get raw readings for each channel
19     r0 = adc[0].value
20     r1 = adc[1].value
21     r2 = adc[2].value
22     r3 = adc[3].value
23
24     # Get voltage readings for each channel
25     v0 = adc[0].volts
26     v1 = adc[1].volts
27     v2 = adc[2].volts

```

(continues on next page)

(continued from previous page)

```

28     v3 = adc[3].volts
29
30     # Print results
31     print("{:>5}\t{:>5.3f}\t{:>5}\t{:>5.3f}\t{:>5}\t{:>5.3f}\t{:>5}\t{:>5.3f}"
32           .format(r0, v0, r1, v1, r2, v2, r3, v3))
33
34     # Sleep for a bit
35     time.sleep(0.5)

```

Listing 2: examples/ads1115\_differential\_simpletest.py

```

1  import time
2  import board
3  import busio
4  from adafruit_ads1x15.differential import ADS1115
5
6  # Create the I2C bus
7  i2c = busio.I2C(board.SCL, board.SDA)
8
9  # Create the ADC object using the I2C bus
10 adc = ADS1115(i2c)
11
12 # Print header
13 print("CHAN 0 - CHAN 1")
14 print("{:>5}\t{:>5.3f}".format('raw', 'v'))
15
16 while True:
17     # Get raw reading for differential input between channel 0 and 1
18     raw = adc[(0, 1)].value
19
20     # Get voltage reading for differential input between channel 0 and 1
21     volts = adc[(0, 1)].volts
22
23     # Print results
24     print("{:>5}\t{:>5.3f}".format(raw, volts))
25
26     # Sleep for a bit
27     time.sleep(0.5)

```

## 5.2 adafruit\_ads1x15

CircuitPython driver for ADS1015/1115 ADCs.

- Author(s): Carter Nelson

**class** adafruit\_ads1x15.adafruit\_ads1x15.**ADC\_Channel** (*adc, channel*)  
Provides per channel access to ADC readings.

**value**  
ADC reading in raw counts.

**volts**  
ADC reading in volts.

**class** adafruit\_ads1x15.adafruit\_ads1x15.**ADS1x15** (*i2c, address=72*)  
Base functionality for ADS1x15 analog to digital converters.



**get\_last\_result()**

Read the last conversion result when in continuous conversion mode. Will return a signed integer value.

**stop\_adc()**

Stop all continuous ADC conversions (either normal or difference mode).

## 5.3 adafruit\_ads1x15.differential

Differential driver for ADS1015/1115 ADCs.

- Author(s): Carter Nelson

**class** `adafruit_ads1x15.differential.ADS1015(*args, **kwargs)`

ADS1015 12-bit differential analog to digital converter instance.

**class** `adafruit_ads1x15.differential.ADS1115(*args, **kwargs)`

ADS1115 16-bit differential analog to digital converter instance.

**class** `adafruit_ads1x15.differential.ADS1x15_Differential(i2c, address=72)`

Base functionality for ADS1x15 analog to digital converters operating in differential mode.

**read\_adc\_difference** (*differential, gain=1, data\_rate=None*)

Read the difference between two ADC channels and return the ADC value as a signed integer result. Differential must be one of: - 0 = Channel 0 minus channel 1 - 1 = Channel 0 minus channel 3 - 2 = Channel 1 minus channel 3 - 3 = Channel 2 minus channel 3

**read\_volts\_difference** (*differential, gain=1, data\_rate=None*)

Read the difference between two ADC channels and return the voltage value as a floating point result. Differential must be one of: - 0 = Channel 0 minus channel 1 - 1 = Channel 0 minus channel 3 - 2 = Channel 1 minus channel 3 - 3 = Channel 2 minus channel 3

**start\_adc\_difference** (*differential, gain=1, data\_rate=None*)

Start continuous ADC conversions between two ADC channels. Differential must be one of: - 0 = Channel 0 minus channel 1 - 1 = Channel 0 minus channel 3 - 2 = Channel 1 minus channel 3 - 3 = Channel 2 minus channel 3 Will return an initial conversion result, then call the `get_last_result()` function continuously to read the most recent conversion result. Call `stop_adc()` to stop conversions.

## 5.4 adafruit\_ads1x15.single\_ended

Single-ended driver for ADS1015/1115 ADCs.

- Author(s): Carter Nelson

**class** `adafruit_ads1x15.single_ended.ADS1015(*args, **kwargs)`

ADS1015 12-bit single ended analog to digital converter instance.

**class** `adafruit_ads1x15.single_ended.ADS1115(*args, **kwargs)`

ADS1115 16-bit single ended analog to digital converter instance.

**class** `adafruit_ads1x15.single_ended.ADS1x15_SingleEnded(i2c, address=72)`

Base functionality for ADS1x15 analog to digital converters operating in single ended mode.

**read\_adc** (*channel, gain=1, data\_rate=None*)

Read a single ADC channel and return the ADC value as a signed integer result. Channel must be a value within 0-3.

**read\_volts** (*channel*, *gain=1*, *data\_rate=None*)

Read a single ADC channel and return the voltage value as a floating point result. Channel must be a value within 0-3.

**start\_adc** (*channel*, *gain=1*, *data\_rate=None*)

Start continuous ADC conversions on the specified channel (0-3). Will return an initial conversion result, then call the `get_last_result()` function to read the most recent conversion result. Call `stop_adc()` to stop conversions.

## CHAPTER 6

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### a

`adafruit_ads1x15.adafruit_ads1x15`, [12](#)  
`adafruit_ads1x15.differential`, [13](#)  
`adafruit_ads1x15.single_ended`, [13](#)



## A

adafruit\_ads1x15.adafruit\_ads1x15 (module), [12](#)  
adafruit\_ads1x15.differential (module), [13](#)  
adafruit\_ads1x15.single\_ended (module), [13](#)  
ADC\_Channel (class in adafruit\_ads1x15.adafruit\_ads1x15), [12](#)  
ADS1015 (class in adafruit\_ads1x15.differential), [13](#)  
ADS1015 (class in adafruit\_ads1x15.single\_ended), [13](#)  
ADS1115 (class in adafruit\_ads1x15.differential), [13](#)  
ADS1115 (class in adafruit\_ads1x15.single\_ended), [13](#)  
ADS1x15 (class in adafruit\_ads1x15.adafruit\_ads1x15), [12](#)  
ADS1x15\_Differential (class in adafruit\_ads1x15.differential), [13](#)  
ADS1x15\_SingleEnded (class in adafruit\_ads1x15.single\_ended), [13](#)

## G

get\_last\_result() (adafruit\_ads1x15.adafruit\_ads1x15.ADS1x15 method), [12](#)

## R

read\_adc() (adafruit\_ads1x15.single\_ended.ADS1x15\_SingleEnded method), [13](#)  
read\_adc\_difference() (adafruit\_ads1x15.differential.ADS1x15\_Differential method), [13](#)  
read\_volts() (adafruit\_ads1x15.single\_ended.ADS1x15\_SingleEnded method), [13](#)  
read\_volts\_difference() (adafruit\_ads1x15.differential.ADS1x15\_Differential method), [13](#)

## S

start\_adc() (adafruit\_ads1x15.single\_ended.ADS1x15\_SingleEnded method), [14](#)  
start\_adc\_difference() (adafruit\_ads1x15.differential.ADS1x15\_Differential method), [13](#)  
stop\_adc() (adafruit\_ads1x15.adafruit\_ads1x15.ADS1x15 method), [13](#)

## V

value (adafruit\_ads1x15.adafruit\_ads1x15.ADC\_Channel attribute), [12](#)  
volts (adafruit\_ads1x15.adafruit\_ads1x15.ADC\_Channel attribute), [12](#)