

---

**Adafruit**  
**CIRCUITPYTHON***ADS1X15LibraryDocumentation*  
**Release 1.0**

**Carter Nelson**

**Mar 20, 2020**



---

## Contents

---

<b>1</b>	<b>Installation &amp; Dependencies</b>	<b>3</b>
1.1	Installing from PyPI . . . . .	3
<b>2</b>	<b>Usage Example</b>	<b>5</b>
2.1	Single Ended . . . . .	5
<b>3</b>	<b>Contributing</b>	<b>7</b>
<b>4</b>	<b>Documentation</b>	<b>9</b>
<b>5</b>	<b>Table of Contents</b>	<b>11</b>
5.1	Simple test . . . . .	11
5.2	ads1x15 . . . . .	12
5.3	ads1015 . . . . .	13
5.4	ads1115 . . . . .	13
5.5	analog_in . . . . .	13
<b>6</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>
	<b>Index</b>	<b>19</b>



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



---

## Installation & Dependencies

---

This driver depends on:

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

Please ensure all dependencies are available on the CircuitPython filesystem. This can be most easily achieved by downloading and installing [the Adafruit library and driver bundle](#) on your device.

### 1.1 Installing from PyPI

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally [from PyPI](#). To install for current user:

```
pip3 install adafruit-circuitpython-ads1x15
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-ads1x15
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .env
source .env/bin/activate
pip3 install adafruit-circuitpython-ads1x15
```





### 2.1 Single Ended

```
import time
import board
import busio
import adafruit_ads1x15.ads1015 as ADS
from adafruit_ads1x15.analog_in import AnalogIn

# Create the I2C bus
i2c = busio.I2C(board.SCL, board.SDA)

# Create the ADC object using the I2C bus
ads = ADS.ADS1015(i2c)

# Create single-ended input on channel 0
chan = AnalogIn(ads, ADS.P0)

# Create differential input between channel 0 and 1
#chan = AnalogIn(ads, ADS.P0, ADS.P1)

print("{:>5}\t{:>5}".format('raw', 'v'))

while True:
    print("{:>5}\t{:>5.3f}".format(chan.value, chan.voltage))
    time.sleep(0.5)
```



## CHAPTER 3

---

### Contributing

---

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



## CHAPTER 4

---

### Documentation

---

For information on building library documentation, please check out [this guide](#).



## 5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/ads1x15\_ads1015\_simpletest.py

```
1 import time
2 import board
3 import busio
4 import adafruit_ads1x15.ads1015 as ADS
5 from adafruit_ads1x15.analog_in import AnalogIn
6
7 # Create the I2C bus
8 i2c = busio.I2C(board.SCL, board.SDA)
9
10 # Create the ADC object using the I2C bus
11 ads = ADS.ADS1015(i2c)
12
13 # Create single-ended input on channel 0
14 chan = AnalogIn(ads, ADS.P0)
15
16 # Create differential input between channel 0 and 1
17 # chan = AnalogIn(ads, ADS.P0, ADS.P1)
18
19 print("{:>5}\t{:>5}".format("raw", "v"))
20
21 while True:
22     print("{:>5}\t{:>5.3f}".format(chan.value, chan.voltage))
23     time.sleep(0.5)
```

Listing 2: examples/ads1x15\_ads1115\_simpletest.py

```

1 import time
2 import board
3 import busio
4 import adafruit_ads1x15.ads1115 as ADS
5 from adafruit_ads1x15.analog_in import AnalogIn
6
7 # Create the I2C bus
8 i2c = busio.I2C(board.SCL, board.SDA)
9
10 # Create the ADC object using the I2C bus
11 ads = ADS.ADS1115(i2c)
12 # you can specify an I2C address instead of the default 0x48
13 # ads = ADS.ADS1115(i2c, address=0x49)
14
15 # Create single-ended input on channel 0
16 chan = AnalogIn(ads, ADS.P0)
17
18 # Create differential input between channel 0 and 1
19 # chan = AnalogIn(ads, ADS.P0, ADS.P1)
20
21 print("{:>5}\t{:>5}".format("raw", "v"))
22
23 while True:
24     print("{:>5}\t{:>5.3f}".format(chan.value, chan.voltage))
25     time.sleep(0.5)

```

## 5.2 ads1x15

CircuitPython base class driver for ADS1015/1115 ADCs.

- Author(s): Carter Nelson

**class** adafruit\_ads1x15.ads1x15.**ADS1x15**(i2c, gain=1, data\_rate=None, mode=256, address=72)

Base functionality for ADS1x15 analog to digital converters.

**data\_rate**

The data rate for ADC conversion in samples per second.

**gain**

The ADC gain.

**gains**

Possible gain settings.

**get\_last\_result**(fast=False)

Read the last conversion result when in continuous conversion mode. Will return a signed integer value. If fast is True, the register pointer is not updated as part of the read. This reduces I2C traffic and increases possible read rate.

**mode**

The ADC conversion mode.

**rate\_config**

Rate configuration masks.



**rates**

Possible data rate settings.

**read** (*pin, is\_differential=False*)

I2C Interface for ADS1x15-based ADCs reads.

**params:**

**param pin** individual or differential pin.

**param bool is\_differential** single-ended or differential read.

**class** adafruit\_ads1x15.ads1x15.**Mode**

An enum-like class representing possible ADC operating modes.

## 5.3 ads1015

CircuitPython driver for ADS1015 ADCs.

- Author(s): Carter Nelson

**class** adafruit\_ads1x15.ads1015.**ADS1015** (*i2c, gain=1, data\_rate=None, mode=256, address=72*)

Class for the ADS1015 12 bit ADC.

**bits**

The ADC bit resolution.

**rate\_config**

Rate configuration masks.

**rates**

Possible data rate settings.

## 5.4 ads1115

CircuitPython driver for 1115 ADCs.

- Author(s): Carter Nelson

**class** adafruit\_ads1x15.ads1115.**ADS1115** (*i2c, gain=1, data\_rate=None, mode=256, address=72*)

Class for the ADS1115 16 bit ADC.

**bits**

The ADC bit resolution.

**rate\_config**

Rate configuration masks.

**rates**

Possible data rate settings.

## 5.5 analog\_in

AnalogIn for single-ended and differential ADC readings.

- Author(s): Carter Nelson, adapted from MCP3xxx original by Brent Rubell

**class** adafruit\_ads1x15.analog\_in.**AnalogIn** (*ads, positive\_pin, negative\_pin=None*)  
AnalogIn Mock Implementation for ADC Reads.

**value**

Returns the value of an ADC pin as an integer.

**voltage**

Returns the voltage from the ADC pin as a floating point value.

## CHAPTER 6

---

### Indices and tables

---

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



### a

`adafruit_ads1x15.ads1015`, [13](#)  
`adafruit_ads1x15.ads1115`, [13](#)  
`adafruit_ads1x15.ads1x15`, [12](#)  
`adafruit_ads1x15.analog_in`, [13](#)



## A

`adafruit_ads1x15.ads1015` (module), 13  
`adafruit_ads1x15.ads1115` (module), 13  
`adafruit_ads1x15.ads1x15` (module), 12  
`adafruit_ads1x15.analog_in` (module), 13  
`ADS1015` (class in `adafruit_ads1x15.ads1015`), 13  
`ADS1115` (class in `adafruit_ads1x15.ads1115`), 13  
`ADS1x15` (class in `adafruit_ads1x15.ads1x15`), 12  
`AnalogIn` (class in `adafruit_ads1x15.analog_in`), 14

## B

`bits` (`adafruit_ads1x15.ads1015.ADS1015` attribute), 13  
`bits` (`adafruit_ads1x15.ads1115.ADS1115` attribute), 13

## D

`data_rate` (`adafruit_ads1x15.ads1x15.ADS1x15` attribute), 12

## G

`gain` (`adafruit_ads1x15.ads1x15.ADS1x15` attribute), 12  
`gains` (`adafruit_ads1x15.ads1x15.ADS1x15` attribute), 12  
`get_last_result()` (`adafruit_ads1x15.ads1x15.ADS1x15` method), 12

## M

`mode` (`adafruit_ads1x15.ads1x15.ADS1x15` attribute), 12  
`Mode` (class in `adafruit_ads1x15.ads1x15`), 13

## R

`rate_config` (`adafruit_ads1x15.ads1015.ADS1015` attribute), 13  
`rate_config` (`adafruit_ads1x15.ads1115.ADS1115` attribute), 13

`rate_config` (`adafruit_ads1x15.ads1x15.ADS1x15` attribute), 12  
`rates` (`adafruit_ads1x15.ads1015.ADS1015` attribute), 13  
`rates` (`adafruit_ads1x15.ads1115.ADS1115` attribute), 13  
`rates` (`adafruit_ads1x15.ads1x15.ADS1x15` attribute), 12  
`read()` (`adafruit_ads1x15.ads1x15.ADS1x15` method), 13

## V

`value` (`adafruit_ads1x15.analog_in.AnalogIn` attribute), 14  
`voltage` (`adafruit_ads1x15.analog_in.AnalogIn` attribute), 14