Adafruit CIRCUITPYTHON $_{ADS1X15}$ Library Documentati

Release 1.0

Carter Nelson

Dec 03, 2018

Contents

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

CHAPTER 1

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
```

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.

CHAPTER 5

Table of Contents

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/ads1115_single_ended_simpletest.py

```
import time
import board
import busio
from adafruit_ads1x15.single_ended import ADS1115

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

# Create the ADC object using the I2C bus
adc = ADS1115(i2c)

# Print header
print("    CHAN 0          CHAN 1          CHAN 2          CHAN 3")
print("{:>5}\t{:>5}\t{:>5}\t{:>5}\t{:>5}\t{:>5}\t{:>5}\t{:>5}"
      .format('raw', 'v', 'raw', 'v', 'raw', 'v', 'raw', 'v'))

while True:
    # Get raw readings for each channel
    r0 = adc[0].value
    r1 = adc[1].value
    r2 = adc[2].value
    r3 = adc[3].value

    # Get voltage readings for each channel
    v0 = adc[0].volts
    v1 = adc[1].volts
    v2 = adc[2].volts
```

(continues on next page)

(continued from previous page)

```
    v3 = adc[3].volts

    # Print results
    print("{:>5}\t{:>5.3f}\t{:>5}\t{:>5.3f}\t{:>5}\t{:>5.3f}\t{:>5}\t{:>5.3f}"
          .format(r0, v0, r1, v1, r2, v2, r3, v3))

    # Sleep for a bit
    time.sleep(0.5)
```

Listing 2: examples/ads1115_differential_simpletest.py

```
import time
import board
import busio
from adafruit_ads1x15.differential import ADS1115

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

# Create the ADC object using the I2C bus
adc = ADS1115(i2c)

# Print header
print("CHAN 0 - CHAN 1")
print("{:>5}\t{:>5}".format('raw', 'v'))

while True:
    # Get raw reading for differential input between channel 0 and 1
    raw = adc[(0, 1)].value

    # Get voltage reading for differential input between channel 0 and 1
    volts = adc[(0, 1)].volts

    # Print results
    print("{:>5}\t{:>5.3f}".format(raw, volts))

    # Sleep for a bit
    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

Python Module Index

a

Index