Adafruit MAX31865 Library Documentation

Release 1.0

Tony DiCola

Apr 26, 2021

Contents

1 Dependencies **3**

2 Installing from PyPI **5**

3 Usage Example **7**

4 Contributing **9**

5 Documentation **11**

6 Table of Contents **13**

6.1 Simple test 13

6.2 `adafruit_max31865` 14

6.2.1 Implementation Notes 14

7 Indices and tables **17**

Python Module Index **19**

Index **21**

CircuitPython module for the MAX31865 thermocouple amplifier.

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

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-max31865
```

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

```
sudo pip3 install adafruit-circuitpython-max31865
```

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-max31865
```

CHAPTER 3

Usage Example

See examples/max31865_simpletest.py for a demo of the usage.

CHAPTER 4

Contributing

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER 5

Documentation

For information on building library documentation, please check out this guide.

CHAPTER 6

Table of Contents

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/max31865_simpletest.py

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

# Simple demo of the MAX31865 thermocouple amplifier.
# Will print the temperature every second.
import time
import board
import digitalio
import adafruit_max31865


# Create sensor object, communicating over the board's default SPI bus
spi = board.SPI()
cs = digitalio.DigitalInOut(board.D5)  # Chip select of the MAX31865 board.
sensor = adafruit_max31865.MAX31865(spi, cs)
# Note you can optionally provide the thermocouple RTD nominal, the reference
# resistance, and the number of wires for the sensor (2 the default, 3, or 4)
# with keyword args:
# sensor = adafruit_max31865.MAX31865(spi, cs, rtd_nominal=100, ref_resistor=430.0,␣
↪wires=2)

# Main loop to print the temperature every second.
while True:
    # Read temperature.
    temp = sensor.temperature
    # Print the value.
    print("Temperature: {0:0.3f}C".format(temp))
```

(continues on next page)

(continued from previous page)

```
    # Delay for a second.
    time.sleep(1.0)
```

6.2 adafruit_max31865

CircuitPython module for the MAX31865 platinum RTD temperature sensor. See examples/simpletest.py for an example of the usage.

- Author(s): Tony DiCola

6.2.1 Implementation Notes

Hardware:

- Adafruit Universal Thermocouple Amplifier MAX31856 Breakout (Product ID: 3263)
- Adafruit PT100 RTD Temperature Sensor Amplifier - MAX31865 (Product ID: 3328)
- Adafruit PT1000 RTD Temperature Sensor Amplifier - MAX31865 (Product ID: 3648)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://circuitpython.org/downloads
- Adafruit's Bus Device library: https://github.com/adafruit/Adafruit_CircuitPython_BusDevice

class adafruit_max31865.**MAX31865**(*spi*, *cs*, **, *rtd_nominal=100*, *ref_resistor=430.0*, *wires=2*, *filter_frequency=60*)

Driver for the MAX31865 thermocouple amplifier.

Parameters

- **spi** (*SPI*) – SPI device
- **cs** (*DigitalInOut*) – Chip Select
- **rtd_nominal** (*int*) – RTD nominal value. Defaults to `100`
- **ref_resistor** (*int*) – Reference resistance. Defaults to `430.0`
- **wires** (*int*) – Number of wires. Defaults to `2`
- **filter_frequency** (*int*) – . Filter frequency. Default to `60`

Quickstart: Importing and using the MAX31865

Here is an example of using the *MAX31865* class. First you will need to import the libraries to use the sensor

```
import board
from digitalio import DigitalInOut, Direction
import adafruit_max31865
```

Once this is done you can define your `board.SPI` object and define your sensor object

```
spi = board.SPI()
cs = digitalio.DigitalInOut(board.D5)  # Chip select of the MAX31865␣
↪board.
sensor = adafruit_max31865.MAX31865(spi, cs)
```

Now you have access to the `temperature` attribute

```
temperature = sensor.temperature
```

auto_convert
The state of the sensor's automatic conversion mode (True/False).

bias
The state of the sensor's bias (True/False).

clear_faults()
Clear any fault state previously detected by the sensor.

fault
The fault state of the sensor. Use `clear_faults()` to clear the fault state. Returns a 6-tuple of boolean values which indicate if any faults are present:

- HIGHTHRESH
- LOWTHRESH
- REFINLOW
- REFINHIGH
- RTDINLOW
- OVUV

read_rtd()
Perform a raw reading of the thermocouple and return its 15-bit value. You'll need to manually convert this to temperature using the nominal value of the resistance-to-digital conversion and some math. If you just want temperature use the temperature property instead.

resistance
Read the resistance of the RTD and return its value in Ohms.

temperature
Read the temperature of the sensor and return its value in degrees Celsius.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

adafruit_max31865, 14

Index

A

`adafruit_max31865` (*module*), 14
`auto_convert` (*adafruit_max31865.MAX31865 attribute*), 15

B

`bias` (*adafruit_max31865.MAX31865 attribute*), 15

C

`clear_faults()` (*adafruit_max31865.MAX31865 method*), 15

F

`fault` (*adafruit_max31865.MAX31865 attribute*), 15

M

`MAX31865` (*class in adafruit_max31865*), 14

R

`read_rtd()` (*adafruit_max31865.MAX31865 method*), 15
`resistance` (*adafruit_max31865.MAX31865 attribute*), 15

T

`temperature` (*adafruit_max31865.MAX31865 attribute*), 15