Adafruit MAX31865 Library Documentation

Release 1.0

Tony DiCola

Jan 15, 2019

Contents

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

Usage Example

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

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-max31865 --
↪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/max31865_simpletest.py

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

import board
import busio
import digitalio

import adafruit_max31865


# Initialize SPI bus and sensor.
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
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)

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

5.2 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

5.2.1 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 ESP8622 and M0-based boards: https://github.com/adafruit/circuitpython/releases
- 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*)
Driver for the MAX31865 thermocouple amplifier.

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 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index