

---

# **Adafruit MCP9808 Library Documentation**

***Release 1.0***

**Phillip Moyer**

**Jun 10, 2021**



---

## Contents

---

<b>1</b>	<b>Dependencies</b>	<b>3</b>
<b>2</b>	<b>Installing from PyPI</b>	<b>5</b>
<b>3</b>	<b>Usage Notes</b>	<b>7</b>
<b>4</b>	<b>Contributing</b>	<b>9</b>
<b>5</b>	<b>Documentation</b>	<b>11</b>
<b>6</b>	<b>Table of Contents</b>	<b>13</b>
6.1	Simple test . . . . .	13
6.2	Temperature Limit test . . . . .	13
6.3	adafruit_mcp9808 . . . . .	14
6.3.1	Implementation Notes . . . . .	14
<b>7</b>	<b>Indices and tables</b>	<b>17</b>
	<b>Python Module Index</b>	<b>19</b>
	<b>Index</b>	<b>21</b>



The MCP9808 is an awesome, high accuracy temperature sensor that communicates over I2C. Its available on [Adafruit](#) as a [breakout](#).



# 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-mcp9808
```

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

```
sudo pip3 install adafruit-circuitpython-mcp9808
```

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-mcp9808
```



## CHAPTER 3

---

### Usage Notes

---

Getting the temperature in Celsius is easy! First, import all of the pins from the board, `board.I2C()` for native I2C communication and the thermometer library itself.

```
import board
import adafruit_mcp9808
```

Next, initialize the I2C bus in a `with` statement so it always gets shut down ok. Then, construct the thermometer class:

```
# Do one reading
with i2c = board.I2C() as i2c:
    t = adafruit_mcp9808.MCP9808(i2c)

# Finally, read the temperature property and print it out
print(t.temperature)
```



## 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](#).





## 6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/mcp9808\_simpletest.py

```
1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  import time
5  import board
6  import adafruit_mcp9808
7
8  i2c = board.I2C() # uses board.SCL and board.SDA
9
10 # To initialise using the default address:
11 mcp = adafruit_mcp9808.MCP9808(i2c)
12
13 # To initialise using a specified address:
14 # Necessary when, for example, connecting A0 to VDD to make address=0x19
15 # mcp = adafruit_mcp9808.MCP9808(i2c_bus, address=0x19)
16
17 while True:
18     tempC = mcp.temperature
19     tempF = tempC * 9 / 5 + 32
20     print("Temperature: {} C {} F ".format(tempC, tempF))
21     time.sleep(2)
```

## 6.2 Temperature Limit test

Show the MCP9808 to setup different temperature values

Listing 2: examples/mcp9808\_temperature\_limits.py

```
1  # SPDX-FileCopyrightText: 2021 Jose David M.
2  # SPDX-License-Identifier: MIT
3
4  """
5  Show the MCP9808 to setup different temperature values
6  """
7
8  import time
9  import board
10 import adafruit_mcp9808
11
12 i2c = board.I2C() # uses board.SCL and board.SDA
13 mcp = adafruit_mcp9808.MCP9808(i2c)
14
15 # Change the values according to the desired values
16 print("Setting Temperature Limits")
17 mcp.upper_temperature = 23
18 mcp.lower_temperature = 10
19 mcp.critical_temperature = 100
20
21 # To verify the limits we need to read the temperature value
22 print(mcp.temperature)
23 time.sleep(0.3) # This is the time temperature conversion at maximum resolution
24
25 # Showing temperature Limits
26 while True:
27     if mcp.below_lt:
28         print("too cold!")
29     if mcp.above_ut:
30         print("getting hot!")
31     if mcp.above_ct:
32         print("Above critical temp!")
```

## 6.3 adafruit\_mcp9808

CircuitPython library to support MCP9808 high accuracy temperature sensor.

- Author(s): Scott Shawcroft, Jose David M.

### 6.3.1 Implementation Notes

#### Hardware:

- Adafruit MCP9808 High Accuracy I2C Temperature Sensor Breakout (Product ID: 1782)

#### Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://github.com/adafruit/circuitpython/releases>
- Adafruit's Bus Device library: [https://github.com/adafruit/Adafruit\\_CircuitPython\\_BusDevice](https://github.com/adafruit/Adafruit_CircuitPython_BusDevice)
- Adafruit's Register library: [https://github.com/adafruit/Adafruit\\_CircuitPython\\_Register](https://github.com/adafruit/Adafruit_CircuitPython_Register)

#### Notes:

1. Datasheet: <http://www.adafruit.com/datasheets/MCP9808.pdf>

**class** adafruit\_mcp9808.**MCP9808** (*i2c\_bus*, *address=24*)  
Interface to the MCP9808 temperature sensor.

#### Parameters

- **i2c\_bus** (*I2C*) – The I2C bus the MCP9808 is connected to.
- **address** (*int*) – The I2C address of the device. Defaults to 0x18

**MCP9808 Settings** You could set the MCP9808 with different temperature limits and compare them with the ambient temperature *Ta*

- **above\_ct** this value will be set to `True` when *Ta* is above this limit
- **above\_ut**: this value will be set to `True` when *Ta* is above this limit
- **below\_lt**: this value will be set to `True` when *Ta* is below this limit

To get this value, you will need to read the temperature, and then access the attribute

#### Quickstart: Importing and using the MCP9808

Here is an example of using the `MCP9808` class. First you will need to import the libraries to use the sensor

```
import board
import adafruit_mcp9808
```

Once this is done you can define your `board.I2C` object and define your sensor object

```
i2c = board.I2C() # uses board.SCL and board.SDA
mcp = adafruit_mcp9808.MCP9808(i2c_bus)
```

Now you have access to the change in temperature using the `temperature` attribute. This temperature is in Celsius.

```
temperature = mcp.temperature
```

#### **above\_critical**

True when the temperature is above the currently set critical temperature. False Otherwise

#### **above\_upper**

True when the temperature is above the currently set high temperature. False Otherwise

#### **below\_lower**

True when the temperature is below the currently set lower temperature. False Otherwise

#### **critical\_temperature**

Critical alarm temperature in Celsius

#### **lower\_temperature**

Lower alarm temperature in Celsius

#### **resolution**

Temperature Resolution in Celsius

Value	Resolution	Reading Time
0	0.5°C	30 ms
1	0.25°C	65 ms
2	0.125°C	130 ms
3	0.0625°C	250 ms

**temperature**

Temperature in Celsius. Read-only.

**upper\_temperature**

Upper alarm temperature in Celsius

## CHAPTER 7

---

### Indices and tables

---

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



### a

adafruit\_mcp9808, [14](#)





## A

`above_critical` (*adafruit\_mcp9808.MCP9808* attribute), [15](#)

`above_upper` (*adafruit\_mcp9808.MCP9808* attribute), [15](#)

`adafruit_mcp9808` (module), [14](#)

## B

`below_lower` (*adafruit\_mcp9808.MCP9808* attribute), [15](#)

## C

`critical_temperature`  
(*adafruit\_mcp9808.MCP9808* attribute),  
[15](#)

## L

`lower_temperature` (*adafruit\_mcp9808.MCP9808* attribute), [15](#)

## M

`MCP9808` (class in *adafruit\_mcp9808*), [15](#)

## R

`resolution` (*adafruit\_mcp9808.MCP9808* attribute),  
[15](#)

## T

`temperature` (*adafruit\_mcp9808.MCP9808* attribute), [16](#)

## U

`upper_temperature` (*adafruit\_mcp9808.MCP9808* attribute), [16](#)