
Adafruit MLX90393 Library Documentation

Release 1.0

Kevin Townsend

Oct 15, 2020

Contents

1	Dependencies	3
1.1	Installing from PyPI	3
2	Usage Example	5
3	Contributing	7
4	Documentation	9
5	Table of Contents	11
5.1	Simple test	11
5.2	adafruit_mlx90393	11
5.2.1	Implementation Notes	12
6	Indices and tables	13
	Python Module Index	15
	Index	17

Adafruit CircuitPython driver for the MLX90393 3-axis magnetometer.

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](#).

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-mlx90939
```

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

```
sudo pip3 install adafruit-circuitpython-mlx90939
```

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-mlx90939
```


CHAPTER 2

Usage Example

```
import time
import busio
import board

import adafruit_mlx90393

I2C_BUS = busio.I2C(board.SCL, board.SDA)
SENSOR = adafruit_mlx90393.MLX90393(I2C_BUS, gain=adafruit_mlx90393.GAIN_1X)

while True:
    MX, MY, MZ = SENSOR.magnetic
    print("[{}]" .format(time.monotonic()))
    print("X: {} uT".format(MX))
    print("Y: {} uT".format(MY))
    print("Z: {} uT".format(MZ))
    # Display the status field if an error occurred, etc.
    if SENSOR.last_status > adafruit_mlx90393.STATUS_OK:
        SENSOR.display_status()
    time.sleep(1.0)
```


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/mlx90393_simpletest.py

```
1 import time
2 import busio
3 import board
4
5 import adafruit_mlx90393
6
7 I2C_BUS = busio.I2C(board.SCL, board.SDA)
8 SENSOR = adafruit_mlx90393.MLX90393(I2C_BUS, gain=adafruit_mlx90393.GAIN_1X)
9
10 while True:
11     MX, MY, MZ = SENSOR.magnetic
12     print("{} {}".format(time.monotonic(), MX))
13     print("X: {} uT".format(MX))
14     print("Y: {} uT".format(MY))
15     print("Z: {} uT".format(MZ))
16     # Display the status field if an error occurred, etc.
17     if SENSOR.last_status > adafruit_mlx90393.STATUS_OK:
18         SENSOR.display_status()
19     time.sleep(1.0)
```

5.2 adafruit_mlx90393

This is a breakout for the Adafruit MLX90393 magnetometer sensor breakout.

- Author(s): ktown

5.2.1 Implementation Notes

Hardware:

- Adafruit [MLX90393 Magnetometer Sensor Breakout Board](#) (Product ID: 4022)

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

class `adafruit_mlx90393.MLX90393` (*i2c_bus*, *address=12*, *gain=7*, *resolution=0*, *filt=7*, *oversampling=3*, *debug=False*)

Driver for the MLX90393 magnetometer. :param `i2c_bus`: The `busio.I2C` object to use. This is the only required parameter. :param `int address`: (optional) The I2C address of the device. :param `int gain`: (optional) The gain level to apply. :param `bool debug`: (optional) Enable debug output.

display_status ()

Prints out the content of the last status byte in a human-readable format.

filter

The filter level.

gain

The gain setting for the device.

last_status

The last status byte received from the sensor.

magnetic

The processed magnetometer sensor values. A 3-tuple of X, Y, Z axis values in microteslas that are signed floats.

oversampling

The oversampling level.

read_data

Reads a single X/Y/Z sample from the magnetometer.

read_reg (*reg*)

Gets the current value of the specified register.

reset ()

Performs a software reset of the sensor.

resolution_x

The X axis resolution.

resolution_y

The Y axis resolution.

resolution_z

The Z axis resolution.

write_reg (*reg*, *value*)

Writes the 16-bit value to the supplied register.

CHAPTER 6

Indices and tables

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

a

adafruit_mlx90393, [11](#)

A

`adafruit_mlx90393` (*module*), 11

D

`display_status()` (*adafruit_mlx90393.MLX90393*
method), 12

F

`filter` (*adafruit_mlx90393.MLX90393* *attribute*), 12

G

`gain` (*adafruit_mlx90393.MLX90393* *attribute*), 12

L

`last_status` (*adafruit_mlx90393.MLX90393* *at-*
tribute), 12

M

`magnetic` (*adafruit_mlx90393.MLX90393* *attribute*),
12

`MLX90393` (*class in adafruit_mlx90393*), 12

O

`oversampling` (*adafruit_mlx90393.MLX90393*
attribute), 12

R

`read_data` (*adafruit_mlx90393.MLX90393* *attribute*),
12

`read_reg()` (*adafruit_mlx90393.MLX90393* *method*),
12

`reset()` (*adafruit_mlx90393.MLX90393* *method*), 12

`resolution_x` (*adafruit_mlx90393.MLX90393*
attribute), 12

`resolution_y` (*adafruit_mlx90393.MLX90393*
attribute), 12

`resolution_z` (*adafruit_mlx90393.MLX90393*
attribute), 12

W

`write_reg()` (*adafruit_mlx90393.MLX90393*
method), 12