Adafruit MLX90393 Library Documentation

Release 1.0

Kevin Townsend

Apr 29, 2021

Contents

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 board
import adafruit_mlx90393

i2c = board.I2C()  # uses board.SCL and board.SDA
SENSOR = adafruit_mlx90393.MLX90393(i2c, 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 occured, 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.

CHAPTER 5

Table of Contents

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/mlx90393_simpletest.py

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

import time
import board
import adafruit_mlx90393

i2c = board.I2C()  # uses board.SCL and board.SDA
SENSOR = adafruit_mlx90393.MLX90393(i2c, 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 occured, etc.
    if SENSOR.last_status > adafruit_mlx90393.STATUS_OK:
        SENSOR.display_status()
    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://circuitpython.org/downloads
- Adafruit's Bus Device library: https://github.com/adafruit/Adafruit_CircuitPython_BusDevice
- Adafruit's Register library: https://github.com/adafruit/Adafruit_CircuitPython_Register

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

Driver for the MLX90393 magnetometer.

Parameters

- **i2c_bus** – The I2C bus the device is connected to
- **address** (*int*) – The I2C device address. Defaults to `0x0C`
- **gain** (*int*) – The gain level to apply. Defaults to `GAIN_1X`
- **debug** (*bool*) – Enable debug output. Defaults to `False`

Quickstart: Importing and using the device

Here is an example of using the *MLX90393* class. First you will need to import the libraries to use the sensor

```
import board
import adafruit_mlx90393
```

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
SENSOR = adafruit_mlx90393.MLX90393(i2c)
```

Now you have access to the *magnetic* attribute

```
MX, MY, MZ = SENSOR.magnetic
```

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

Python Module Index

a

Index