AdafruitDPS310 Library Documentation

Release 1.0

Bryan Siepert

May 13, 2020

Contents

Library for the DPS310 Precision Barometric Pressure Sensor

CHAPTER 1

Dependencies

This driver depends on:

- Adafruit CircuitPython
- Bus Device
- Register

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-dps310
```

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

```
sudo pip3 install adafruit-circuitpython-dps310
```

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-dps310
```

CHAPTER 3

Usage Example

```
import time
import board
import busio
import adafruit_dps310

i2c = busio.I2C(board.SCL, board.SDA)

dps310 = adafruit_dps310.DPS310(i2c)

while True:
    print("Temperature = %.2f *C"%dps310.temperature)
    print("Pressure = %.2f hPa"%dps310.pressure)
    print("")
    time.sleep(1.0)
```

Caveat: by default the library initializes the IC with constant temperature and pressure measurements at 64Hz with 64 samples. It is not possible to change the IC's mode, temperature_oversample_count or pressure_oversample_count on-the-fly so resetting the IC and operation parameteres is required. For instance, to set the mode to continuous pressure measurement at 1Hz with 2 samples:

```
dps310.reset()
dps310.pressure_oversample_count = adafruit_dps310.SampleCount.COUNT_2
dps310.pressure_rate = adafruit_dps310.Rate.RATE_1_HZ
dps310.mode = adafruit_dps310.Mode.CONT_PRESSURE
dps310.wait_pressure_ready()
```

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/dps310_simpletest.py

```
import time
import board
import busio
import adafruit_dps310

i2c = busio.I2C(board.SCL, board.SDA)

dps310 = adafruit_dps310.DPS310(i2c)

while True:
    print("Temperature = %.2f *C" % dps310.temperature)
    print("Pressure = %.2f hPa" % dps310.pressure)
    print("")
    time.sleep(1.0)
```

6.2 adafruit_dps310

Library for the DPS310 Precision Barometric Pressure Sensor

- Author(s): Bryan Siepert

6.2.1 Implementation Notes

Hardware:

- Adafruit's DPS310 Breakout: https://www.adafruit.com/product/4494

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_dps310.**CV**
struct helper

classmethod add_values(*value_tuples*)
Add CV values to the class

classmethod is_valid(*value*)
Validate that a given value is a member

class adafruit_dps310.**DPS310**(*i2c_bus*, *address=119*)
Library for the DPS310 Precision Barometric Pressure Sensor.

Parameters

- **i2c_bus** (*I2C*) – The I2C bus the DPS310 is connected to.
- **address** – The I2C slave address of the sensor

initialize()
Initialize the sensor to continuous measurement

mode
The measurement mode. Must be a *Mode*. See the *Mode* documentation for details

pressure
Returns the current pressure reading in kPA

pressure_oversample_count
The number of samples taken per pressure measurement. Must be a *SampleCount*

pressure_rate
Configure the pressure measurement rate. Must be a *Rate*

pressure_ready
Returns true if pressure readings are ready

reset()
Reset the sensor

temperature
The current temperature reading in degrees C

temperature_oversample_count
The number of samples taken per temperature measurement. Must be a *SampleCount*

temperature_rate
Configure the temperature measurement rate. Must be a *Rate*

temperature_ready
Returns true if there is a temperature reading ready

wait_pressure_ready()
Wait until a pressure measurement is available

To avoid waiting indefinitely this function raises an error if the sensor isn't configured for pressure measurements, ie. `Mode.ONE_PRESSURE`, `Mode.CONT_PRESSURE` or `Mode.CONT_PRESTEMP` See the *Mode* documentation for details.

wait_temperature_ready()
Wait until a temperature measurement is available.

To avoid waiting indefinitely this function raises an error if the sensor isn't configured for temperate measurements, ie. `Mode.ONE_TEMPERATURE`, `Mode.CONT_TEMP` or `Mode.CONT_PRESTEMP`. See the *Mode* documentation for details.

class adafruit_dps310.**Mode**
Options for `mode`

Mode	Description
`Mode.IDLE`	Puts the sensor into a shutdown state
`Mode.ONE_PRESSURE`	Setting *mode* to `Mode.ONE_PRESSURE` takes a single pressure measurement then switches to `Mode.IDLE`
`Mode.ONE_TEMPERATURE`	Setting *mode* to `Mode.ONE_TEMPERATURE` takes a single temperature measurement then switches to `Mode.IDLE`
`Mode.CONT_PRESSURE`	Take pressure measurements at the current *pressure_rate*. *temperature* will not be updated
`Mode.CONT_TEMP`	Take temperature measurements at the current *temperature_rate*. *pressure* will not be updated
`Mode.CONT_PRESTEMP`	Take temperature and pressure measurements at the current *pressure_rate* and *temperature_rate*

class adafruit_dps310.**Rate**
Options for *pressure_rate* and *temperature_rate*

class adafruit_dps310.**SampleCount**
Options for *temperature_oversample_count* and *pressure_oversample_count*

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index