
sparkfun_qwiic_adxl313

Release 0.0.6

SparkFun Electronics

Jun 24, 2021

CONTENTS:

1	Contents	3
2	Dependencies	5
3	Documentation	7
4	Installation	9
4.1	PyPi Installation	9
4.2	Local Installation	9
5	Example Use	11
6	Table of Contents	13
6.1	API Reference	13
6.1.1	qwiic_adxl313	13
6.2	Example 1: Basic Readings	18
6.3	Example 2: Set Range	20
6.4	Example 3: Auto Sleep	22
6.5	Example 4: Low Power Mode	25
6.6	Example 5: Standby	27
6.7	Example 6: interrupt	29
6.8	Example 7: FIFO	32
7	Indices and tables	39
	Python Module Index	41
	Index	43

Python module for the [SparkFun 3-Axis Digital Accelerometer Breakout - ADXL313 \(Qwiic\)](#)

This python package is a port of the existing [SparkFun ADXL313 Arduino Library](#)

This package can be used in conjunction with the overall [SparkFun qwiic Python Package](#)

New to qwiic? Take a look at the entire [SparkFun qwiic ecosystem](#).

CONTENTS

- *Dependencies*
- *Installation*
- *Documentation*
- *Example Use*

DEPENDENCIES

This driver package depends on the qwiic I2C driver: [Qwiic_I2C_Py](#)

DOCUMENTATION

The SparkFun qwiic Adxl313 module documentation is hosted at [ReadTheDocs](#)

INSTALLATION

4.1 PyPi Installation

This repository is hosted on PyPi as the [sparkfun-qwiic-adx1313](#) package. On systems that support PyPi installation via pip, this library is installed using the following commands

For all users (note: the user must have sudo privileges):

```
sudo pip install sparkfun-qwiic-adx1313
```

For the current user:

```
pip install sparkfun-qwiic-adx1313
```

4.2 Local Installation

To install, make sure the setuptools package is installed on the system.

Direct installation at the command line:

```
python setup.py install
```

To build a package for use with pip:

```
python setup.py sdist
```

A package file is built and placed in a subdirectory called dist. This package file can be installed using pip.

```
cd dist  
pip install sparkfun_qwiic_adx1313-<version>.tar.gz
```


EXAMPLE USE

See the examples directory for more detailed use examples.

```
from __future__ import print_function
import qwiic_adxl313
import time
import sys

def runExample():

    print("\nSparkFun Adxl313 Example 1 - Basic Readings\n")
    myAdxl = qwiic_adxl313.QwiicAdxl313()

    if myAdxl.connected == False:
        print("The Qwiic ADXL313 device isn't connected to the system. Please check your
↳connection", \
            file=sys.stderr)
        return
    else:
        print("Device connected successfully.")

    myAdxl.measureModeOn()

    while True:
        if myAdxl.dataReady():
            myAdxl.readAccel() # read all axis from sensor, note this also updates all
↳instance variables
            print(\
                '{: 06d}'.format(myAdxl.x)\
                , '\t', '{: 06d}'.format(myAdxl.y)\
                , '\t', '{: 06d}'.format(myAdxl.z)\
                )
            time.sleep(0.03)
        else:
            print("Waiting for data")
            time.sleep(0.5)

if __name__ == '__main__':
    try:
        runExample()
    except (KeyboardInterrupt, SystemExit) as exErr:
```

(continues on next page)

(continued from previous page)

```
print("\nEnding Example 1")
sys.exit(0)
```


TABLE OF CONTENTS

6.1 API Reference

6.1.1 qwiic_adxl313

Python module for the [SparkFun Triple Axis Accelerometer Breakout - ADXL313 (QWIIC)](<https://www.sparkfun.com/products/17241>)

This python package is a port of the existing [SparkFun ADXL313 Arduino Library](https://github.com/sparkfun/SparkFun_ADXL313_Arduino_Library)

This package can be used in conjunction with the overall [SparkFun qwiic Python Package](https://github.com/sparkfun/Qwiic_Py)

New to qwiic? Take a look at the entire [SparkFun qwiic ecosystem](<https://www.sparkfun.com/qwiic>).

```
class qwiic_adxl313.QwiicAdxl313(address=None, i2c_driver=None)
```

Parameters

- **address** – The I2C address to use for the device. If not provided, the default address is used.
- **i2c_driver** – An existing i2c driver object. If not provided a driver object is created.

Returns The ADXL313 device object.

Return type Object

ActivityINT(*state*)

Enables or disables the activity interrupt

Parameters **state** – 1 = enabled, 0 = disabled

Returns Returns true if the function was completed, otherwise False.

Return type bool

DataReadyINT(*state*)

Enables or disables the dataready interrupt :param state: 1 = enabled, 0 = disabled

Returns Returns true if the function was completed, otherwise False.

Return type bool

InactivityINT(*state*)

Enables or disables the inactivity interrupt :param state: 1 = enabled, 0 = disabled

Returns Returns true if the function was completed, otherwise False.

Return type bool

OverrunINT(*state*)

Enables or disables the overrun interrupt ;param state: 1 = enabled, 0 = disabled

Returns Returns true of the function was completed, otherwise False.

Return type bool

WatermarkINT(*state*)

Enables or disables the watermark interrupt ;param state: 1 = enabled, 0 = disabled

Returns Returns true of the function was completed, otherwise False.

Return type bool

autosleepOff()

Turns Autosleep off.

Returns Returns true of the function was completed, otherwise False.

Return type bool

autosleepOn()

Turns Autosleep on.

Returns Returns true of the function was completed, otherwise False.

Return type bool

begin()

Initialize the operation of the module

Returns Returns true of the initialization was successful, otherwise False.

Return type bool

clearFifo()

Clears all FIFO data by bypassing FIFO and re-entering previous mode

Returns Returns true of the function was completed, otherwise False.

Return type bool

property connected

Determine if a device is connected to the system..

Returns True if the device is connected, otherwise False.

Return type bool

dataReady()

Reads INT Source register, returns dataready bit status (0 or 1)

Returns Status of dataready bit within the int source register

Return type bool

getActivityThreshold()

Gets the Threshold Value for Detecting Activity.

Returns activity detection threshold

Return type byte

getFifoEntriesAmount()

Get FIFO entries amount (0-32)

Returns FIFO entries amount (0-32)

Return type byte

getFifoMode()

Get the current FIFO mode (0=bypass,1=fifo,2=stream,3=trigger)

Returns FIFO mode (0=bypass,1=fifo,2=stream,3=trigger)

Return type byte

getFifoSamplesThreshold()

Get FIFO samples threshold (0-32)

Returns FIFO samples threshold (0-32)

Return type byte

getInactivityThreshold()

Gets the Threshold Value for Detecting Inactivity.

Returns inactivity detection threshold

Return type byte

getRange()

Reads the current range setting on the device

Returns range setting of the device (from in DATA_FORMAT register)

Return type float

getRegisterBit(*regAddress, bitPos*)

Gets the bit status of specified register

Parameters

- **regAddress** – The address of the register you'd like to read
- **bitPos** – The specific bit of the register you'd like to read

return Status of bit specified within the register (0 or 1)

rtype bool

getTimeInactivity()

Gets time requirement below inactivity threshold to detect inactivity

Returns inactivity detection time requirement

Return type byte

isConnected()

Determine if a device is connected to the system..

Returns True if the device is connected, otherwise False.

Return type bool

isInterruptEnabled(*interruptBit*)

Get status of whether an interrupt is enabled or disabled. :param interruptBit: the desired int bit you'd like to read

Returns Returns true if the interrupt bit is enabled, otherwise false

Return type bool

limit(*num*, *minimum=1*, *maximum=255*)

Limits input 'num' between minimum and maximum values. Default minimum value is 1 and maximum value is 255.

Parameters

- **num** – the number you'll like to limit
- **minimum** – the min (default 1)
- **maximum** – the max (default 255)

Returns your new limited number within min and max

Return type int

measureModeOn()

sets the measure bit, putting decive in measure mode, ready for reading data

Returns Returns true of the function was completed, otherwise False.

Return type bool

readAccel()

Reads Acceleration into Three Class Variables: x, y and z

Returns Returns true of the function was completed, otherwise False.

Return type bool

setActivityThreshold(*activityThreshold*)

Sets the Threshold Value for Detecting Activity. :param activityThreshold: 0-255

Returns Returns true of the function was completed, otherwise False.

Return type bool

setActivityX(*state*)

Enalbes or disables X axis participattion in activity detection :param state: 1 = enabled, 0 = disabled

Returns Returns true of the function was completed, otherwise False.

Return type bool

setActivityY(*state*)

Enalbes or disables Y axis participattion in activity detection :param state: 1 = enabled, 0 = disabled

Returns Returns true of the function was completed, otherwise False.

Return type bool

setActivityZ(*state*)

Enalbes or disables Z axis participattion in activity detection :param state: 1 = enabled, 0 = disabled

Returns Returns true of the function was completed, otherwise False.

Return type bool

setFifoMode(*mode*)

Set FIFO mode

Parameters **mode** – FIFO mode (ADXL313_FIFO_MODE_BYPASS, ADXL313_FIFO_MODE_FIFO, ADXL313_FIFO_MODE_STREAM, ADXL313_FIFO_MODE_TRIGGER)

Returns Returns true of the function was completed, otherwise False.

Return type bool

setFifoSamplesThreshold(*samples*)

Set FIFO samples threshold (0-32)

Parameters **mode** – FIFO samples threshold (0-32)

Returns Returns true of the function was completed, otherwise False.

Return type bool

setInactivityThreshold(*inactivityThreshold*)

Sets the Threshold Value for Detecting Inactivity. :param inactivityThreshold: 0-255

Returns Returns true of the function was completed, otherwise False.

Return type bool

setInactivityX(*state*)

Enalbes or disables X axis participation in inactivity detection :param state: 1 = enabled, 0 = disabled

Returns Returns true of the function was completed, otherwise False.

Return type bool

setInactivityY(*state*)

Enalbes or disables Y axis participation in inactivity detection :param state: 1 = enabled, 0 = disabled

Returns Returns true of the function was completed, otherwise False.

Return type bool

setInactivityZ(*state*)

Enalbes or disables Z axis participation in inactivity detection :param state: 1 = enabled, 0 = disabled

Returns Returns true of the function was completed, otherwise False.

Return type bool

setInterrupt(*interruptBit, state*)

Sets the enable bit (0 or 1) for one desired int inside the ADXL313_INT_ENABLE register :param interruptBit: the desired int bit you'd like to change :param state: 1 = enabled, 0 = disabled

Returns Returns true of the function was completed, otherwise False.

Return type bool

setInterruptMapping(*interruptBit, interruptPin*)

Maps the desired interrupt bit (from intsource) to the desired hardware interrupt pin :param interruptBit: the desired int bit you'd like to map :param interruptPin: ADXL313_INT1_PIN or ADXL313_INT2_PIN

Returns Returns true of the function was completed, otherwise False.

Return type bool

setRange(*new_range*)

Sets the range setting on the device

Parameters **range** – range value desired (ADXL313_RANGE_05_G, ADXL313_RANGE_1_G, etc)

Returns Returns true of the function was completed, otherwise False.

Return type bool

setRegisterBit(*regAddress, bitPos, state*)

Sets or clears bit of specified register

Parameters

- **regAddress** – The address of the register you’d like to affect.
 - **bitPos** – The specific bit of the register you’d like to affect.
 - **state** – The condition of the bit you’d like to set/clear (0 or 1).
- return** Returns true of the function was completed, otherwise False.
- rtype** bool

setTimeInactivity(*timeInactivity*)

Sets time requirement below inactivity threshold to detect inactivity :param timeInactivity: 0-255

Returns Returns true of the function was completed, otherwise False.

Return type bool

standby()

clears the measure bit, putting decive in standby mode, ready for configuration

Returns Returns true of the function was completed, otherwise False.

Return type bool

updateIntSourceStatuses()

Reads int Source Register once and updates all individual calss statuses

Returns Returns true of the function was completed, otherwise False.

Return type bool

6.2 Example 1: Basic Readings

Listing 1: examples/ex1_qwiic_adxl313_basic_readings.py

```

1  #!/usr/bin/env python
2  #-----
3  # ex1_qwiic_adxl313_basic_readings.py
4  #
5  # Simple Example for the Qwiic ADXL313 Device
6  # Read values of x/y/z axis of the ADXL313 (via I2C), print them to terminal.
7  # This uses default configuration (1G range, full resolution, 100Hz datarate).
8  #-----
9  #
10 # Written by SparkFun Electronics, October 2020
11 #
12 # This python library supports the SparkFun Electroncis qwiic
13 # qwiic sensor/board ecosystem on a Raspberry Pi (and compatable) single
14 # board computers.
15 #
16 # More information on qwiic is at https://www.sparkfun.com/qwiic
17 #
18 # Do you like this library? Help support SparkFun. Buy a board!
19 #

```

(continues on next page)

(continued from previous page)

```

20 #=====
21 # Copyright (c) 2019 SparkFun Electronics
22 #
23 # Permission is hereby granted, free of charge, to any person obtaining a copy
24 # of this software and associated documentation files (the "Software"), to deal
25 # in the Software without restriction, including without limitation the rights
26 # to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
27 # copies of the Software, and to permit persons to whom the Software is
28 # furnished to do so, subject to the following conditions:
29 #
30 # The above copyright notice and this permission notice shall be included in all
31 # copies or substantial portions of the Software.
32 #
33 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
34 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
35 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
36 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
37 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
38 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
39 # SOFTWARE.
40 #=====
41 # Example 1
42 #
43
44 from __future__ import print_function
45 import qwiic_adxl313
46 import time
47 import sys
48
49 def runExample():
50
51     print("\nSparkFun Adxl313 Example 1 - Basic Readings\n")
52     myAdxl = qwiic_adxl313.QwiicAdxl313()
53
54     if myAdxl.connected == False:
55         print("The Qwiic ADXL313 device isn't connected to the system. Please
↳check your connection", \
56             file=sys.stderr)
57         return
58     else:
59         print("Device connected successfully.")
60
61     myAdxl.measureModeOn()
62
63     while True:
64         if myAdxl.dataReady():
65             myAdxl.readAccel() # read all axis from sensor, note this also
↳updates all instance variables
66             print(\
67                 '{: 06d}'.format(myAdxl.x)\
68                 , '\t', '{: 06d}'.format(myAdxl.y)\
69                 , '\t', '{: 06d}'.format(myAdxl.z)\

```

(continues on next page)

(continued from previous page)

```

70         )
71         time.sleep(0.03)
72     else:
73         print("Waiting for data")
74         time.sleep(0.5)
75
76 if __name__ == '__main__':
77     try:
78         runExample()
79     except (KeyboardInterrupt, SystemExit) as exErr:
80         print("\nEnding Example 1")
81         sys.exit(0)
82
83

```

6.3 Example 2: Set Range

Listing 2: examples/ex2_qwiic_adxl313_set_range.py

```

1  #!/usr/bin/env python
2  #-----
3  # ex2_qwiic_adxl313_set_range.py
4  #
5  # Simple Example for the Qwiic ADXL313 DeviceSet range of the sensor to 2G.
6  # Then read values of x/y/z axis of the ADXL313 (via I2C), print them to terminal.
7  # Note, other range options are: 0.5G, 1G[default], 2G or 4 G.
8  # Except for custom range, this example uses default configuration (full resolution,
9  ↪ 100Hz datarate).
10 #-----
11 #
12 # Written by SparkFun Electronics, October 2020
13 #
14 # This python library supports the SparkFun Electronics qwiic
15 # qwiic sensor/board ecosystem on a Raspberry Pi (and compatible) single
16 # board computers.
17 #
18 # More information on qwiic is at https://www.sparkfun.com/qwiic
19 #
20 # Do you like this library? Help support SparkFun. Buy a board!
21 #
22 #=====
23 # Copyright (c) 2019 SparkFun Electronics
24 #
25 # Permission is hereby granted, free of charge, to any person obtaining a copy
26 # of this software and associated documentation files (the "Software"), to deal
27 # in the Software without restriction, including without limitation the rights
28 # to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
29 # copies of the Software, and to permit persons to whom the Software is
30 # furnished to do so, subject to the following conditions:

```

(continues on next page)

(continued from previous page)

```

31 # The above copyright notice and this permission notice shall be included in all
32 # copies or substantial portions of the Software.
33 #
34 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
35 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
36 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
37 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
38 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
39 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
40 # SOFTWARE.
41 #=====
42 # Example 2
43 #
44
45 from __future__ import print_function
46 import qwiic_adxl313
47 import time
48 import sys
49
50 def runExample():
51
52     print("\nSparkFun Adxl313 Example 2 - Set Range\n")
53     myAdxl = qwiic_adxl313.QwiicAdxl313()
54
55     if myAdxl.connected == False:
56         print("The Qwiic ADXL313 device isn't connected to the system. Please_
↳check your connection", \
57             file=sys.stderr)
58         return
59     else:
60         print("Device connected successfully.")
61
62     myAdxl.standby();           # Must be in standby before changing settings.
63                               # This is here just in case we already_
↳had sensor powered and/or
64                               # configured from a previous setup.
65
66     myAdxl.setRange(myAdxl.ADXL313_RANGE_2_G);
67
68     # Try some other range settings by uncommented your choice below
69     #myAdxl.setRange(myAdxl.ADXL313_RANGE_05_G);
70     #myAdxl.setRange(myAdxl.ADXL313_RANGE_1_G);
71     #myAdxl.setRange(myAdxl.ADXL313_RANGE_2_G);
72     #myAdxl.setRange(myAdxl.ADXL313_RANGE_4_G);
73
74     myAdxl.measureModeOn()
75
76     while True:
77         if myAdxl.dataReady():
78             myAdxl.readAccel() # read all axis from sensor, note this also_
↳updates all instance variables
79             print(\

```

(continues on next page)

(continued from previous page)

```

80         '{: 06d}'.format(myAdxl.x)\
81         , '\t', '{: 06d}'.format(myAdxl.y)\
82         , '\t', '{: 06d}'.format(myAdxl.z)\
83         )
84         time.sleep(0.03)
85     else:
86         print("Waiting for data")
87         time.sleep(0.5)
88
89 if __name__ == '__main__':
90     try:
91         runExample()
92     except (KeyboardInterrupt, SystemExit) as exErr:
93         print("\nEnding Example 1")
94         sys.exit(0)
95
96

```

6.4 Example 3: Auto Sleep

Listing 3: examples/ex3_qwiic_adxl313_auto_sleep.py

```

1  #!/usr/bin/env python
2  #-----
3  # ex3_qwiic_adxl313_auto_sleep.py
4  #
5  # Simple Example for the Qwiic ADXL313 DeviceSet that shows how to use Autosleep feature.
6  # First, setup THRESH_INACT, TIME_INACT, and participating axis.
7  # These settings will determine when the unit will go into autosleep mode and save power!
8  # We are only going to use the x-axis (and are disabling y-axis and z-axis).
9  # This is so you can place the board "flat" inside your project,
10 # and we can ignore gravity on z-axis.
11 #-----
12 #
13 # Written by SparkFun Electronics, October 2020
14 #
15 # This python library supports the SparkFun Electronics qwiic
16 # qwiic sensor/board ecosystem on a Raspberry Pi (and compatible) single
17 # board computers.
18 #
19 # More information on qwiic is at https://www.sparkfun.com/qwiic
20 #
21 # Do you like this library? Help support SparkFun. Buy a board!
22 #
23 #=====
24 # Copyright (c) 2019 SparkFun Electronics
25 #
26 # Permission is hereby granted, free of charge, to any person obtaining a copy
27 # of this software and associated documentation files (the "Software"), to deal
28 # in the Software without restriction, including without limitation the rights

```

(continues on next page)

(continued from previous page)

```

29 # to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
30 # copies of the Software, and to permit persons to whom the Software is
31 # furnished to do so, subject to the following conditions:
32 #
33 # The above copyright notice and this permission notice shall be included in all
34 # copies or substantial portions of the Software.
35 #
36 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
37 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
38 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
39 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
40 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
41 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
42 # SOFTWARE.
43 #=====
44 # Example 3
45 #
46
47 from __future__ import print_function
48 import qwiic_adxl313
49 import time
50 import sys
51
52 def runExample():
53
54     print("\nSparkFun Adxl313 Example 3 - Setup Autosleep and then only print
↳ values when it's awake.\n")
55     myAdxl = qwiic_adxl313.QwiicAdxl313()
56
57     if myAdxl.connected == False:
58         print("The Qwiic ADXL313 device isn't connected to the system. Please
↳ check your connection", \
59             file=sys.stderr)
60         return
61     else:
62         print("Device connected successfully.")
63
64     myAdxl.standby()           # Must be in standby before changing settings.
65                               # This is here just in case we already
↳ had sensor powered and/or
66                               # configured from a previous setup.
67
68     myAdxl.setRange(myAdxl.ADXL313_RANGE_4_G)
69
70     # setup activity sensing options
71     myAdxl.setActivityX(True)           # enable x-axis participation in
↳ detecting activity
72     myAdxl.setActivityY(False)        # disable y-axis participation in
↳ detecting activity
73     myAdxl.setActivityZ(False)        # disable z-axis participation in
↳ detecting activity
74     myAdxl.setActivityThreshold(10)    # 0-255 (62.5mg/LSB)

```

(continues on next page)

(continued from previous page)

```

75
76     # setup inactivity sensing options
77     myAdxl.setInactivityX(True)                # enable x-axis participation
↳in detecting inactivity
78     myAdxl.setInactivityY(False)             # disable y-axis participation in
↳detecting inactivity
79     myAdxl.setInactivityZ(False)             # disable z-axis participation in
↳detecting inactivity
80     myAdxl.setInactivityThreshold(10)        # 0-255 (62.5mg/LSB)
81     myAdxl.setTimeInactivity(5)              # 0-255 (1sec/LSB)
82
83     myAdxl.ActivityINT(1)
84     myAdxl.InactivityINT(1)
85
86     myAdxl.autosleepOn()
87
88     myAdxl.measureModeOn()
89
90     while True:
91         myAdxl.updateIntSourceStatuses(); # this will update all INTSOURCE
↳statuses.
92
93         if myAdxl.ADXL313_INTSOURCE_INACTIVITY:
94             print("Inactivity detected.")
95             time.sleep(1)
96         if myAdxl.ADXL313_INTSOURCE_DATAREADY:
97             myAdxl.readAccel() # read all axis from sensor, note this also
↳updates all instance variables
98             print(\
99                 '{: 06d}'.format(myAdxl.x)\
100                , '\t', '{: 06d}'.format(myAdxl.y)\
101                , '\t', '{: 06d}'.format(myAdxl.z)\
102                )
103         else:
104             print("Device is asleep (dataReady is reading false)")
105             time.sleep(0.05)
106
107 if __name__ == '__main__':
108     try:
109         runExample()
110     except (KeyboardInterrupt, SystemExit) as exErr:
111         print("\nEnding Example 1")
112         sys.exit(0)
113
114

```

6.5 Example 4: Low Power Mode

Listing 4: examples/ex4_qwiic_adxl313_low_power_mode.py

```

1  #!/usr/bin/env python
2  #-----
3  # ex4_qwiic_adxl313_low_power_mode.py
4  #
5  # Shows how to use Low Power feature.
6  # In addition to turning on low power mode, you will also want to consider
7  # bandwidth rate. This will affect your results in low power land.
8  # In this example, we will turn on low power mode and set BW to 12.5Hz.
9  # Then we will only take samples at or above 12.5Hz (so we don't miss samples)
10 #
11 #-----
12 #
13 # Written by SparkFun Electronics, October 2020
14 #
15 # This python library supports the SparkFun Electroncis qwiic
16 # qwiic sensor/board ecosystem on a Raspberry Pi (and compatable) single
17 # board computers.
18 #
19 # More information on qwiic is at https://www.sparkfun.com/qwiic
20 #
21 # Do you like this library? Help support SparkFun. Buy a board!
22 #
23 #=====
24 # Copyright (c) 2019 SparkFun Electronics
25 #
26 # Permission is hereby granted, free of charge, to any person obtaining a copy
27 # of this software and associated documentation files (the "Software"), to deal
28 # in the Software without restriction, including without limitation the rights
29 # to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
30 # copies of the Software, and to permit persons to whom the Software is
31 # furnished to do so, subject to the following conditions:
32 #
33 # The above copyright notice and this permission notice shall be included in all
34 # copies or substantial portions of the Software.
35 #
36 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
37 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
38 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
39 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
40 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
41 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
42 # SOFTWARE.
43 #=====
44 # Example 4
45 #
46
47 from __future__ import print_function
48 import qwiic_adxl313

```

(continues on next page)

(continued from previous page)

```

49 import time
50 import sys
51
52 def runExample():
53
54     print("\nSparkFun Adxl313 Example 4 - Low power mode ON with 12.5Hz bandwidth.\n
↳")
55     myAdxl = qwiic_adxl313.QwiicAdxl313()
56
57     if myAdxl.connected == False:
58         print("The Qwiic ADXL313 device isn't connected to the system. Please
↳check your connection", \
59             file=sys.stderr)
60         return
61     else:
62         print("Device connected successfully.")
63
64     myAdxl.standby()           # Must be in standby before changing settings.
65                               # This is here just in case we already
↳had sensor powered and/or
66                               # configured from a previous setup.
67
68     myAdxl.lowPowerOn()
69     #also try:
70     #myAdxl.lowPower = True
71
72     myAdxl.setBandwidth(myAdxl.ADXL313_BW_12_5)
73     #also try:
74     #myAdxl.bandwidth = myAdxl.ADXL313_BW_12_5
75
76     #12.5Hz is the best power savings.
77     #Other options possible are the following.
78     #Note, bandwidths not listed below do not cause power savings.
79     #ADXL313_BW_200           (115uA in low power)
80     #ADXL313_BW_100          (82uA in low power)
81     #ADXL313_BW_50           (64uA in low power)
82     #ADXL313_BW_25           (57uA in low power)
83     #ADXL313_BW_12_5        (50uA in low power)
84     #ADXL313_BW_6_25        (43uA in low power)
85
86     myAdxl.measureModeOn()
87
88     while True:
89         myAdxl.updateIntSourceStatuses(); # this will update all INTSOURCE
↳statuses.
90
91         if myAdxl.ADXL313_INTSOURCE_DATAREADY:
92             myAdxl.readAccel() # read all axis from sensor, note this also
↳updates all instance variables
93             print(\
94                 '{: 06d}'.format(myAdxl.x)\
95                 , '\t', '{: 06d}'.format(myAdxl.y)\

```

(continues on next page)

(continued from previous page)

```

96         , '\t', '{: 06d}'.format(myAdxl.z)\
97     )
98     else:
99         print("Waiting for data.")
100        time.sleep(0.08)
101
102 if __name__ == '__main__':
103     try:
104         runExample()
105     except (KeyboardInterrupt, SystemExit) as exErr:
106         print("\nEnding Example 4")
107         sys.exit(0)

```

6.6 Example 5: Standby

Listing 5: examples/ex5_qwiic_adxl313_standby.py

```

1  #!/usr/bin/env python
2  #-----
3  # ex5_qwiic_adxl313_standby.py
4  #
5  # Simple Example for the Qwiic ADXL313 DeviceSet that Shows how to switch the sensor
6  # between standby mode and measure mode.
7  # This example will put the device in measure mode and print 100 readings to terminal,
8  # Then enter standby mode for 5 seconds.
9  # Then loop.
10 # Note, the typical current required in each mode is as follows:
11 # Standby: 0.1uA
12 # Measure: 55-170uA
13 #-----
14 #
15 # Written by SparkFun Electronics, October 2020
16 #
17 # This python library supports the SparkFun Electronics qwiic
18 # qwiic sensor/board ecosystem on a Raspberry Pi (and compatible) single
19 # board computers.
20 #
21 # More information on qwiic is at https://www.sparkfun.com/qwiic
22 #
23 # Do you like this library? Help support SparkFun. Buy a board!
24 #
25 #=====
26 # Copyright (c) 2019 SparkFun Electronics
27 #
28 # Permission is hereby granted, free of charge, to any person obtaining a copy
29 # of this software and associated documentation files (the "Software"), to deal
30 # in the Software without restriction, including without limitation the rights
31 # to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
32 # copies of the Software, and to permit persons to whom the Software is
33 # furnished to do so, subject to the following conditions:

```

(continues on next page)

(continued from previous page)

```

34 #
35 # The above copyright notice and this permission notice shall be included in all
36 # copies or substantial portions of the Software.
37 #
38 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
39 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
40 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
41 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
42 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
43 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
44 # SOFTWARE.
45 #=====
46 # Example 5
47 #
48
49 from __future__ import print_function
50 import qwiic_adxl313
51 import time
52 import sys
53
54 def runExample():
55
56     print("\nSparkFun Adxl313 Example 5 - Standby mode and measure mode.\n")
57     myAdxl = qwiic_adxl313.QwiicAdxl313()
58
59     if myAdxl.connected == False:
60         print("The Qwiic ADXL313 device isn't connected to the system. Please
↳check your connection", \
61             file=sys.stderr)
62         return
63     else:
64         print("Device connected successfully.")
65
66     while True:
67         # enter measure mode
68         print("Entering measure mode.")
69         myAdxl.measureModeOn()
70         for i in range(100):
71
72             myAdxl.updateIntSourceStatuses(); # this will update all
↳INTSOURCE statuses.
73
74             if myAdxl.ADXL313_INTSOURCE_DATAREADY:
75                 myAdxl.readAccel() # read all axis from sensor, note
↳this also updates all instance variables
76                 print(\
77                     '{: 06d}'.format(myAdxl.x)\
78                     , '\t', '{: 06d}'.format(myAdxl.y)\
79                     , '\t', '{: 06d}'.format(myAdxl.z)\
80                 )
81             else:
82                 print("Waiting for data.")

```

(continues on next page)

(continued from previous page)

```

83         time.sleep(0.05)
84         print("Entering Standby Mode")
85         myAdxl.standby()
86         time.sleep(5) # 5 seconds of standby... really saving power during this.
87         ↪time (0.1uA)
88
89 if __name__ == '__main__':
90     try:
91         runExample()
92     except (KeyboardInterrupt, SystemExit) as exErr:
93         print("\nEnding Example 1")
94         sys.exit(0)
95
96

```

6.7 Example 6: interrupt

Listing 6: examples/ex6_qwiic_adxl313_interrupt.py

```

1  #!/usr/bin/env python
2  #-----
3  # ex6_qwiic_adxl313_interrupt.py
4  #
5  # Simple Example for the Qwiic ADXL313 DeviceSet that shows how to setup a interrupt on.
6  ↪the ADXL313.
7  # Note, for this example we will setup the interrupt and poll the interrupt register
8  # via software.
9  # We will utilize the autosleep feature of the sensor.
10 # When it senses inactivity, it will go to sleep.
11 # When it senses new activity, it will wake up and trigger the INT1 pin.
12 # We will monitor the status of the interrupt by continuing to read the
13 # interrupt register on the device.
14
15 # ///// Autosleep setup /////
16 # First, setup THRESH_INACT, TIME_INACT, and participating axis.
17 # These settings will determine when the unit will go into autosleep mode and save power!
18 # We are only going to use the x-axis (and are disabling y-axis and z-axis).
19 # This is so you can place the board "flat" inside your project,
20 # and we can ignore gravity on z-axis.
21
22 # ///// Interrupt setup /////
23 # Enable activity interrupt.
24 # Map activity interrupt to "int pin 1".
25 # This hardware interrupt pin setup could be monitored by a GPIO on the raspi,
26 # or external system, however, for this example, we will simply
27 # poll the interrupt register via software to monitor its status.
28 #-----
29 #
30 # Written by SparkFun Electronics, October 2020

```

(continues on next page)

(continued from previous page)

```

30 #
31 # This python library supports the SparkFun Electronics qwiic
32 # qwiic sensor/board ecosystem on a Raspberry Pi (and compatible) single
33 # board computers.
34 #
35 # More information on qwiic is at https://www.sparkfun.com/qwiic
36 #
37 # Do you like this library? Help support SparkFun. Buy a board!
38 #
39 #=====
40 # Copyright (c) 2019 SparkFun Electronics
41 #
42 # Permission is hereby granted, free of charge, to any person obtaining a copy
43 # of this software and associated documentation files (the "Software"), to deal
44 # in the Software without restriction, including without limitation the rights
45 # to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
46 # copies of the Software, and to permit persons to whom the Software is
47 # furnished to do so, subject to the following conditions:
48 #
49 # The above copyright notice and this permission notice shall be included in all
50 # copies or substantial portions of the Software.
51 #
52 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
53 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
54 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
55 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
56 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
57 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
58 # SOFTWARE.
59 #=====
60 # Example 6
61 #
62
63 from __future__ import print_function
64 import qwiic_adxl313
65 import time
66 import sys
67
68 def runExample():
69
70     print("\nSparkFun Adxl313 Example 6 - Setup Autosleep and interrupts, then only
↳ print values when it's awake.\n")
71     myAdxl = qwiic_adxl313.QwiicAdxl313()
72
73     if myAdxl.connected == False:
74         print("The Qwiic ADXL313 device isn't connected to the system. Please
↳ check your connection", \
75             file=sys.stderr)
76         return
77     else:
78         print("Device connected successfully.")
79

```

(continues on next page)

(continued from previous page)

```

80     myAdxl.standby()           # Must be in standby before changing settings.
81                               # This is here just in case we already
↳had sensor powered and/or
82                               # configured from a previous setup.
83
84     myAdxl.setRange(myAdxl.ADXL313_RANGE_4_G)
85
86     # setup activity sensing options
87     myAdxl.setActivityX(True)   # enable x-axis participation in
↳detecting activity
88     myAdxl.setActivityY(False)  # disable y-axis participation in
↳detecting activity
89     myAdxl.setActivityZ(False)  # disable z-axis participation in
↳detecting activity
90     myAdxl.setActivityThreshold(10) # 0-255 (62.5mg/LSB)
91
92     # setup inactivity sensing options
93     myAdxl.setInactivityX(True) # enable x-axis participation
↳in detecting inactivity
94     myAdxl.setInactivityY(False) # disable y-axis participation in
↳detecting inactivity
95     myAdxl.setInactivityZ(False) # disable z-axis participation in
↳detecting inactivity
96     myAdxl.setInactivityThreshold(10) # 0-255 (62.5mg/LSB)
97     myAdxl.setTimeInactivity(5)      # 0-255 (1sec/LSB)
98
99     # Interrupt Mapping
100    # when activity of inactivity is detected, it will effect the int1 pin on the
↳sensor
101    myAdxl.setInterruptMapping(myAdxl.ADXL313_INT_ACTIVITY_BIT, myAdxl.ADXL313_INT1_
↳PIN)
102    myAdxl.setInterruptMapping(myAdxl.ADXL313_INT_INACTIVITY_BIT, myAdxl.ADXL313_
↳INT1_PIN)
103
104    myAdxl.ActivityINT(1)
105    myAdxl.InactivityINT(1)
106    myAdxl.DataReadyINT(0)
107
108    myAdxl.autosleepOn()
109
110    myAdxl.measureModeOn()
111
112    # print int enable statuses, to verify we're setup correctly
113    print("activity int enable: ", myAdxl.isInterruptEnabled(myAdxl.ADXL313_INT_
↳ACTIVITY_BIT))
114    print("inactivity int enable: ", myAdxl.isInterruptEnabled(myAdxl.ADXL313_INT_
↳INACTIVITY_BIT))
115    print("dataReady int enable: ", myAdxl.isInterruptEnabled(myAdxl.ADXL313_INT_
↳DATA_READY_BIT))
116    time.sleep(5)
117
118    while True:

```

(continues on next page)

(continued from previous page)

```

119         myAdxl.updateIntSourceStatuses() # this will update all INTSOURCE_
↳ statuses.
120
121         if myAdxl.ADXL313_INTSOURCE_INACTIVITY:
122             print("Inactivity detected.")
123             time.sleep(1)
124         if myAdxl.ADXL313_INTSOURCE_DATAREADY:
125             myAdxl.readAccel() # read all axis from sensor, note this also_
↳ updates all instance variables
126             print(\
127                 '{: 06d}'.format(myAdxl.x)\
128                 , '\t', '{: 06d}'.format(myAdxl.y)\
129                 , '\t', '{: 06d}'.format(myAdxl.z)\
130             )
131         else:
132             print("Device is asleep (dataReady is reading false)")
133             time.sleep(0.05)
134
135 if __name__ == '__main__':
136     try:
137         runExample()
138     except (KeyboardInterrupt, SystemExit) as exErr:
139         print("\nEnding Example 1")
140         sys.exit(0)
141
142

```

6.8 Example 7: FIFO

Listing 7: examples/ex7_qwiic_adxl313_fifo.py

```

1  #!/usr/bin/env python
2  #-----
3  # ex7_qwiic_adxl313_fifo.py
4  #
5  # Simple Example for the Qwiic ADXL313 DeviceSet that shows how to setup the FIFO on the_
↳ ADXL313.
6  # One key advantage of using the FIFO is that it allows us to
7  # let the ADXL313 store up to 32 samples in it's FIFO "buffer".
8  # While it is doing this, we can use our microcontroller to do other things,
9  # Then, when the FIFO is full (or close to), we can quickly read in the 32 samples.
10
11 # In order to use the FIFO in this way, we need to set it up to fire an interrupt
12 # when it gets "almost full". This threshold of samples is called the "watermark".
13 # When the watermark level is reached, it will fire the interrupt INT1.
14 # Our raspi will be monitoring the watermark int source bit, and then quickly
15 # read whatever's in the FIFO and save it to a log file.
16 # Note, we can't print the data in real time to the terminal
17 # because python terminal is too slow.
18

```

(continues on next page)

(continued from previous page)

```

19 # Some timestamps of each stage of this cycle will also be printed.
20 # This will allow us to fine tune bandwidth and watermark settings.
21 # The "Entries" of the FIFO_STATUS register will also be printed before each read.
22 # This will tell us how many samples are currently held in the FIFO.
23 # This will allow us to read the entire contents and keep an eye on how full it is
24 # getting before each read. This will help us fine tune how much time we have
25 # between each read to do other things. (in this example, we are simply going to do
26 # a delay and print dots, but you could choose to do more useful things).
27
28 # **SPI app note**
29 # Note, this example uses I2C to communicate the the sensor.
30 # If you are going to use SPI, then you will need to add in a sufficient
31 # delay in between reads (at least 5uSec), to allow the FIFO to "pop" the next
32 # reading in the data registers. See datasheet page 16 for more info.
33
34 # ///// FIFO setup /////
35 # Stream mode
36 # Trigger INT1, Note, this example does not utilize monitoring this hardware interrupt.
37 # We will be monitoring via software by reading the int source and watching the
38 # watermark bit.
39 # Watermark Threshold (aka (samples in FIFO_CTL register)): 30
40
41 # ///// Interrupt setup /////
42 # Enable watermark interrupt.
43 # Map watermark interrupt to "int pin 1".
44 # This hardware interrupt pin setup could be monitored by a GPIO on the raspi,
45 # or external system, however, for this example, we will simply
46 # poll the interrupt register via software to monitor its status.
47
48 #-----
49 #
50 # Written by SparkFun Electronics, October 2020
51 #
52 # This python library supports the SparkFun Electronics qwiic
53 # qwiic sensor/board ecosystem on a Raspberry Pi (and compatible) single
54 # board computers.
55 #
56 # More information on qwiic is at https://www.sparkfun.com/qwiic
57 #
58 # Do you like this library? Help support SparkFun. Buy a board!
59 #
60 #=====
61 # Copyright (c) 2019 SparkFun Electronics
62 #
63 # Permission is hereby granted, free of charge, to any person obtaining a copy
64 # of this software and associated documentation files (the "Software"), to deal
65 # in the Software without restriction, including without limitation the rights
66 # to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
67 # copies of the Software, and to permit persons to whom the Software is
68 # furnished to do so, subject to the following conditions:
69 #
70 # The above copyright notice and this permission notice shall be included in all

```

(continues on next page)

(continued from previous page)

```

71 # copies or substantial portions of the Software.
72 #
73 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
74 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
75 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
76 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
77 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
78 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
79 # SOFTWARE.
80 #=====
81 # Example 7
82 #
83
84 from __future__ import print_function
85 import qwiic_adxl313
86 import time
87 import sys
88
89 lastWatermarkTime = 0 # used for printing timestamps in debug
90 fifoEntriesAmount = 0 # used to know how much is currently in the fifo and make sure to
↳read it all out.
91
92 def micros():
93     return round(time.time_ns()/1000)
94
95 # Open a log file in "append mode", We must log data here because printing to terminal
↳is too slow
96 logfile = open("log.txt","a")
97
98 def runExample():
99
100     print("\nSparkFun Adxl313 Example 7 - FIFO reading with debug info about timing.
↳\n")
101     myAdxl = qwiic_adxl313.QwiicAdxl313()
102
103     if myAdxl.connected == False:
104         print("The Qwiic ADXL313 device isn't connected to the system. Please
↳check your connection", \
105             file=sys.stderr)
106         return
107     else:
108         print("Device connected successfully.")
109
110     myAdxl.standby() # Must be in standby before changing settings.
111                    # This is here just in case we already
↳had sensor powered and/or
112                    # configured from a previous setup.
113
114     myAdxl.setRange(myAdxl.ADXL313_RANGE_4_G)
115
116     # set bandwidth
117     # note, 12.5Hz was chosen for this example to highlight the FIFO wait/read cycle

```

(continues on next page)

(continued from previous page)

```

118     # you can tweak BW and the fifo sample threshold to suit your application.
119     myAdxl.setBandwidth(myAdxl.ADXL313_BW_12_5)
120     # also try:
121     # myAdxl.bandwidth = myAdxl.ADXL313_BW_12_5
122
123     # setup activity sensing options
124     myAdxl.setActivityX(False)           # disable x-axis participation in
↳ detecting activity
125     myAdxl.setActivityY(False)         # disable y-axis participation in
↳ detecting activity
126     myAdxl.setActivityZ(False)        # disable z-axis participation in
↳ detecting activity
127
128     # setup inactivity sensing options
129     myAdxl.setInactivityX(False)       # disable x-axis participation in
↳ detecting inactivity
130     myAdxl.setInactivityY(False)      # disable y-axis participation in
↳ detecting inactivity
131     myAdxl.setInactivityZ(False)      # disable z-axis participation in
↳ detecting inactivity
132
133     # FIFO SETUP
134     myAdxl.setFifoMode(myAdxl.ADXL313_FIFO_MODE_STREAM)
135     myAdxl.setFifoSamplesThreshold(30) # can be 1-32
136
137     # Interrupt Mapping
138     # when fifo fills up to watermark level, it will effect the int1 pin on the
↳ sensor
139     myAdxl.setInterruptMapping(myAdxl.ADXL313_INT_WATERMARK_BIT, myAdxl.ADXL313_INT1_
↳ PIN)
140
141     # enable/disable interrupts
142     # note, we set them all here, just in case there were previous settings,
143     # that need to be changed for this example to work properly.
144     myAdxl.ActivityINT(0)             # disable activity
145     myAdxl.InactivityINT(0)          # disable inactivity
146     myAdxl.DataReadyINT(0)           # disable dataready
147     myAdxl.WatermarkINT(1)           # enable watermark
148
149     myAdxl.autosleepOff()             # just in case it was set from a previous
↳ setup
150
151     myAdxl.measureModeOn()            # wakes up sensor from stanby and puts
↳ into measurement mode
152
153     # print int enable statuses, to verify we're setup correctly
154     print("activity int enable: ", myAdxl.isInterruptEnabled(myAdxl.ADXL313_INT_
↳ ACTIVITY_BIT))
155     print("inactivity int enable: ", myAdxl.isInterruptEnabled(myAdxl.ADXL313_INT_
↳ INACTIVITY_BIT))
156     print("dataReady int enable: ", myAdxl.isInterruptEnabled(myAdxl.ADXL313_INT_
↳ DATA_READY_BIT))

```

(continues on next page)

(continued from previous page)

```

157     print("FIFO watermark int enable: ", myAdxl.isInterruptEnabled(myAdxl.ADXL313_
↳INT_WATERMARK_BIT))
158     print("FIFO watermark Samples Threshold: ", myAdxl.getFifoSamplesThreshold())
159     print("FIFO mode: ", myAdxl.getFifoMode())
160
161     lastWatermarkTime = micros()
162
163     myAdxl.clearFifo() # clear FIFO for a fresh start on this example.
164     # The FIFO may have been full from previous use
165     # and then would fail to cause an interrupt when starting this example.
166
167     uSecTimer = 0 # used to print some "dots" during down time in cycle
168     while True:
169         myAdxl.updateIntSourceStatuses() # this will update all INTSOURCE_
↳statuses.
170         if myAdxl.ADXL313_INTSOURCE_WATERMARK:
171             entries = myAdxl.getFifoEntriesAmount()
172             timegap_us = (micros() - lastWatermarkTime)
173             timegap_ms = round(timegap_us / 1000)
174
175             print("\nWatermark Interrupt! Time since last read: ", timegap_
↳us, "us ", timegap_ms, "ms Entries:", entries)
176             lastWatermarkTime = micros()
177             while entries > 0:
178                 myAdxl.updateIntSourceStatuses() # this will update all_
↳INTSOURCE statuses.
179                 if myAdxl.ADXL313_INTSOURCE_DATAREADY:
180                     myAdxl.readAccel() # read all axis from sensor,
↳note this also updates all instance variables
181
182                     # Gotta log data to a text file, because_
↳printing to terminal is too slow
183                     logfile.write(str(myAdxl.x))
184                     logfile.write("\t")
185                     logfile.write(str(myAdxl.y))
186                     logfile.write("\t")
187                     logfile.write(str(myAdxl.z))
188                     logfile.write("\n")
189                     entries -= 1 # we've read one more entry, so let
↳'s keep track and keep going until we're done
190                     else:
191                         print("Waiting for Data.")
192
193                     time.sleep(0.000001) # sleep 1 microsecond
194                     uSecTimer += 1
195                     if uSecTimer > 100:
196                         print(".", end = ' ')
197                         uSecTimer = 0
198
199 if __name__ == '__main__':
200     try:
201

```

(continues on next page)

(continued from previous page)

```
202         runExample()
203     except (KeyboardInterrupt, SystemExit) as exErr:
204         print("\nEnding Example 1")
205         logfile.close()
206         sys.exit(0)
207
208
```


INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

q

qwiic_adxl313, 13

INDEX

A

ActivityINT() (*qwiic_adxl313.QwiicAdxl313* method),
13
autosleepOff() (*qwiic_adxl313.QwiicAdxl313*
method), 14
autosleepOn() (*qwiic_adxl313.QwiicAdxl313* method),
14

B

begin() (*qwiic_adxl313.QwiicAdxl313* method), 14

C

clearFifo() (*qwiic_adxl313.QwiicAdxl313* method),
14
connected (*qwiic_adxl313.QwiicAdxl313* property), 14

D

dataReady() (*qwiic_adxl313.QwiicAdxl313* method),
14
DataReadyINT() (*qwiic_adxl313.QwiicAdxl313*
method), 13

G

getActivityThreshold()
(*qwiic_adxl313.QwiicAdxl313* method),
14
getFifoEntriesAmount()
(*qwiic_adxl313.QwiicAdxl313* method),
14
getFifoMode() (*qwiic_adxl313.QwiicAdxl313* method),
15
getFifoSamplesThreshold()
(*qwiic_adxl313.QwiicAdxl313* method),
15
getInactivityThreshold()
(*qwiic_adxl313.QwiicAdxl313* method),
15
getRange() (*qwiic_adxl313.QwiicAdxl313* method), 15
getRegisterBit() (*qwiic_adxl313.QwiicAdxl313*
method), 15
getTimeInactivity() (*qwiic_adxl313.QwiicAdxl313*
method), 15

I

InactivityINT() (*qwiic_adxl313.QwiicAdxl313*
method), 13
isConnected() (*qwiic_adxl313.QwiicAdxl313* method),
15
isInterruptEnabled() (*qwiic_adxl313.QwiicAdxl313*
method), 15

L

limit() (*qwiic_adxl313.QwiicAdxl313* method), 16

M

measureModeOn() (*qwiic_adxl313.QwiicAdxl313*
method), 16
module
qwiic_adxl313, 13

O

OverrunINT() (*qwiic_adxl313.QwiicAdxl313* method),
14

Q

qwiic_adxl313
module, 13
QwiicAdxl313 (class in *qwiic_adxl313*), 13

R

readAccel() (*qwiic_adxl313.QwiicAdxl313* method),
16

S

setActivityThreshold()
(*qwiic_adxl313.QwiicAdxl313* method),
16
setActivityX() (*qwiic_adxl313.QwiicAdxl313*
method), 16
setActivityY() (*qwiic_adxl313.QwiicAdxl313*
method), 16
setActivityZ() (*qwiic_adxl313.QwiicAdxl313*
method), 16
setFifoMode() (*qwiic_adxl313.QwiicAdxl313* method),
16

setFifoSamplesThreshold() (qwiic_adxl313.QwiicAdxl313 method), 17

setInactivityThreshold() (qwiic_adxl313.QwiicAdxl313 method), 17

setInactivityX() (qwiic_adxl313.QwiicAdxl313 method), 17

setInactivityY() (qwiic_adxl313.QwiicAdxl313 method), 17

setInactivityZ() (qwiic_adxl313.QwiicAdxl313 method), 17

setInterrupt() (qwiic_adxl313.QwiicAdxl313 method), 17

setInterruptMapping() (qwiic_adxl313.QwiicAdxl313 method), 17

setRange() (qwiic_adxl313.QwiicAdxl313 method), 17

setRegisterBit() (qwiic_adxl313.QwiicAdxl313 method), 17

setTimeInactivity() (qwiic_adxl313.QwiicAdxl313 method), 18

standby() (qwiic_adxl313.QwiicAdxl313 method), 18

U

updateIntSourceStatuses() (qwiic_adxl313.QwiicAdxl313 method), 18

W

WatermarkINT() (qwiic_adxl313.QwiicAdxl313 method), 14