sparkfun_qwiic_adxl313

Release 0.0.6

SparkFun Electronics

Jun 24, 2021

CONTENTS:

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.

CHAPTER ONE

CONTENTS

CHAPTER

TWO

DEPENDENCIES

This driver package depends on the qwiic I2C driver: Qwiic_I2C_Py

CHAPTER THREE

DOCUMENTATION

The SparkFun qwiic Adxl313 module documentation is hosted at ReadTheDocs

CHAPTER

FOUR

INSTALLATION

4.1 PyPi Installation

This repository is hosted on PyPi as the sparkfun-qwiic-adxl313 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-adxl313
```

For the current user:

```
pip install sparkfun-qwiic-adxl313
```

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_adxl313-<version>.tar.gz
```

CHAPTER

FIVE

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)
```

CHAPTER

SIX

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 of 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 of 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 of 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 initializtion 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 conntected 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 theshold

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

`getFifoSamplesThreshhold()`

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 theshold

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 spcified 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 conntected 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 interrruptBit: 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'l 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

`setFifoSamplesThreshhold`(*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 participattion 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 participattion 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 participattion 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 interrruptBit: 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 interrruptBit: 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

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

(continues on next page)

(continued from previous page)

```
#==================================================================================
# Copyright (c) 2019 SparkFun Electronics
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#==================================================================================
# Example 1
#

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)\
```

(continues on next page)

(continued from previous page)

```
                        )
                        time.sleep(0.03)
                else:
                        print("Waiting for data")
                        time.sleep(0.5)

if __name__ == '__main__':
        try:
                runExample()
        except (KeyboardInterrupt, SystemExit) as exErr:
                print("\nEnding Example 1")
                sys.exit(0)


```

6.3 Example 2: Set Range

Listing 2: examples/ex2_qwiic_adxl313_set_range.py

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

(continues on next page)

(continued from previous page)

```
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#==================================================================================
# Example 2
#

from __future__ import print_function
import qwiic_adxl313
import time
import sys

def runExample():

        print("\nSparkFun Adxl313  Example 2 - Set Range\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.standby();        # Must be in standby before changing settings.
                                                # This is here just in case we already had sensor powered and/or
                                                # configured from a previous setup.

        myAdxl.setRange(myAdxl.ADXL313_RANGE_2_G);

        # Try some other range settings by uncommented your choice below
        #myAdxl.setRange(myAdxl.ADXL313_RANGE_05_G);
        #myAdxl.setRange(myAdxl.ADXL313_RANGE_1_G);
        #myAdxl.setRange(myAdxl.ADXL313_RANGE_2_G);
        #myAdxl.setRange(myAdxl.ADXL313_RANGE_4_G);

        myAdxl.measureModeOn()

        while True:
                if myAdxl.dataReady():
                        myAdxl.readAccel() # read all axis from sensor, note this also updates all instance variables
                        print(\
```

(continues on next page)

(continued from previous page)

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


```

6.4 Example 3: Auto Sleep

Listing 3: examples/ex3_qwiic_adxl313_auto_sleep.py

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

(continues on next page)

(continued from previous page)

```
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#==================================================================================
# Example 3
#

from __future__ import print_function
import qwiic_adxl313
import time
import sys

def runExample():

        print("\nSparkFun Adxl313  Example 3 - Setup Autosleep and then only print values when it's awake.\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.standby()        # Must be in standby before changing settings.
                                                # This is here just in case we already had sensor powered and/or
                                                # configured from a previous setup.

        myAdxl.setRange(myAdxl.ADXL313_RANGE_4_G)

        # setup activity sensing options
        myAdxl.setActivityX(True)                # enable x-axis participation in detecting activity
        myAdxl.setActivityY(False)                # disable y-axis participation in detecting activity
        myAdxl.setActivityZ(False)                # disable z-axis participation in detecting activity
        myAdxl.setActivityThreshold(10)        # 0-255 (62.5mg/LSB)
```

(continues on next page)

(continued from previous page)

```python

        # setup inactivity sensing options
        myAdxl.setInactivityX(True)                        # enable x-axis participation in detecting inactivity
        myAdxl.setInactivityY(False)                # disable y-axis participation in detecting inactivity
        myAdxl.setInactivityZ(False)                # disable z-axis participation in detecting inactivity
        myAdxl.setInactivityThreshold(10)        # 0-255 (62.5mg/LSB)
        myAdxl.setTimeInactivity(5)                        # 0-255 (1sec/LSB)

        myAdxl.ActivityINT(1)
        myAdxl.InactivityINT(1)

        myAdxl.autosleepOn()

        myAdxl.measureModeOn()

        while True:
                myAdxl.updateIntSourceStatuses(); # this will update all INTSOURCE statuses.

                if myAdxl.ADXL313_INTSOURCE_INACTIVITY:
                        print("Inactivity detected.")
                        time.sleep(1)
                if myAdxl.ADXL313_INTSOURCE_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)\
                        )
                else:
                        print("Device is asleep (dataReady is reading false)")
                time.sleep(0.05)

if __name__ == '__main__':
        try:
                runExample()
        except (KeyboardInterrupt, SystemExit) as exErr:
                print("\nEnding Example 1")
                sys.exit(0)


```

6.5 Example 4: Low Power Mode

Listing 4: examples/ex4_qwiic_adxl313_low_power_mode.py

```python
#!/usr/bin/env python
#-----------------------------------------------------------------------------
# ex4_qwiic_adxl313_low_power_mode.py
#
#   Shows how to use Low Power feature.
#   In addition to turning on low power mode, you will also want to consider
#   bandwidth rate. This will affect your results in low power land.
#   In this example, we will turn on low power mode and set BW to 12.5Hz.
#   Then we will only take samples at or above 12.5Hz (so we don't miss samples)
#
#------------------------------------------------------------------------
#
# Written by  SparkFun Electronics, October 2020
#
# This python library supports the SparkFun Electroncis qwiic
# qwiic sensor/board ecosystem on a Raspberry Pi (and compatable) single
# board computers.
#
# More information on qwiic is at https://www.sparkfun.com/qwiic
#
# Do you like this library? Help support SparkFun. Buy a board!
#
#==================================================================================
# Copyright (c) 2019 SparkFun Electronics
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#==================================================================================
# Example 4
#

from __future__ import print_function
import qwiic_adxl313
```

(continues on next page)

(continued from previous page)

```
import time
import sys

def runExample():

        print("\nSparkFun Adxl313  Example 4 - Low power mode ON with 12.5Hz bandwidth.\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.standby()        # Must be in standby before changing settings.
                                                # This is here just in case we already had sensor powered and/or
                                                # configured from a previous setup.

        myAdxl.lowPowerOn()
        #also try:
        #myAdxl.lowPower = True

        myAdxl.setBandwidth(myAdxl.ADXL313_BW_12_5)
        #also try:
        #myAdxl.bandwidth = myAdxl.ADXL313_BW_12_5

        #12.5Hz is the best power savings.
        #Other options possible are the following.
        #Note, bandwidths not listed below do not cause power savings.
        #ADXL313_BW_200                    (115uA in low power)
        #ADXL313_BW_100                    (82uA in low power)
        #ADXL313_BW_50                    (64uA in low power)
        #ADXL313_BW_25                    (57uA in low power)
        #ADXL313_BW_12_5            (50uA in low power)
        #ADXL313_BW_6_25                  (43uA in low power)

        myAdxl.measureModeOn()

        while True:
                myAdxl.updateIntSourceStatuses(); # this will update all INTSOURCE statuses.

                if myAdxl.ADXL313_INTSOURCE_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)\
```

(continues on next page)

(continued from previous page)

```
                        , '\t', '{: 06d}'.format(myAdxl.z)\
                        )
                else:
                        print("Waiting for data.")
                time.sleep(0.08)

if __name__ == '__main__':
        try:
                runExample()
        except (KeyboardInterrupt, SystemExit) as exErr:
                print("\nEnding Example 4")
                sys.exit(0)
```

6.6 Example 5: Standby

Listing 5: examples/ex5_qwiic_adxl313_standby.py

```
#!/usr/bin/env python
#-----------------------------------------------------------------------------
# ex5_qwiic_adxl313_standby.py
#
# Simple Example for the Qwiic ADXL313 DeviceSet that Shows how to switch the sensor
# between stanby mode and measure mode.
# This example will put the device in measure mode and print 100 readings to terminal,
# Then enter standby mode for 5 seconds.
# Then loop.
# Note, the typical current required in each mode is as follows:
# Standby: 0.1uA
# Measure: 55-170uA
#------------------------------------------------------------------------
#
# Written by  SparkFun Electronics, October 2020
#
# This python library supports the SparkFun Electroncis qwiic
# qwiic sensor/board ecosystem on a Raspberry Pi (and compatable) single
# board computers.
#
# More information on qwiic is at https://www.sparkfun.com/qwiic
#
# Do you like this library? Help support SparkFun. Buy a board!
#
#==================================================================================
# Copyright (c) 2019 SparkFun Electronics
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
```

(continues on next page)

(continued from previous page)

```
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#==================================================================================
# Example 5
#

from __future__ import print_function
import qwiic_adxl313
import time
import sys

def runExample():

        print("\nSparkFun Adxl313  Example 5 - Standby mode and measure mode.\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.")

        while True:
                # enter measure mode
                print("Entering measure mode.")
                myAdxl.measureModeOn()
                for i in range(100):

                        myAdxl.updateIntSourceStatuses(); # this will update all INTSOURCE statuses.

                        if myAdxl.ADXL313_INTSOURCE_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)\
                                )
                        else:
                                print("Waiting for data.")
```

(continues on next page)

(continued from previous page)

```
                        time.sleep(0.05)
                print("Endering Standby Mode")
                myAdxl.standby()
                time.sleep(5) # 5 seconds of standby... really saving power during this time (0.1uA)


if __name__ == '__main__':
        try:
                runExample()
        except (KeyboardInterrupt, SystemExit) as exErr:
                print("\nEnding Example 1")
                sys.exit(0)


```

6.7 Example 6: interrupt

Listing 6: examples/ex6_qwiic_adxl313_interrupt.py

```
#!/usr/bin/env python
#-----------------------------------------------------------------------------
# ex6_qwiic_adxl313_interrupt.py
#
# Simple Example for the Qwiic ADXL313 DeviceSet that shows how to setup a interrupt on the ADXL313.
# Note, for this example we will setup the interrupt and poll the interrupt register
# via software.
# We will utilize the autosleep feature of the sensor.
# When it senses inactivity, it will go to sleep.
# When it senses new activity, it will wake up and trigger the INT1 pin.
# We will monitor the status of the interrupt by continuing to read the
# interrupt register on the device.

# ///// Autosleep setup //////
# First, setup THRESH_INACT, TIME_INACT, and participating axis.
# These settings will determine when the unit will go into autosleep mode and save power!
# We are only going to use the x-axis (and are disabling y-axis and z-axis).
# This is so you can place the board "flat" inside your project,
# and we can ignore gravity on z-axis.

# ///// Interrupt setup //////
# Enable activity interrupt.
# Map activity interrupt to "int pin 1".
# This harware interrupt pin setup could be monitored by a GPIO on the raspi,
# or external system, however, for this example, we will simply
# poll the interrupt register via software to monitor its status.
#------------------------------------------------------------------------
#
# Written by  SparkFun Electronics, October 2020
```

(continues on next page)

(continued from previous page)

```
#
# This python library supports the SparkFun Electroncis qwiic
# qwiic sensor/board ecosystem on a Raspberry Pi (and compatable) single
# board computers.
#
# More information on qwiic is at https://www.sparkfun.com/qwiic
#
# Do you like this library? Help support SparkFun. Buy a board!
#
#==================================================================================
# Copyright (c) 2019 SparkFun Electronics
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#==================================================================================
# Example 6
#

from __future__ import print_function
import qwiic_adxl313
import time
import sys

def runExample():

        print("\nSparkFun Adxl313  Example 6 - Setup Autosleep and interrupts, then only print values when it's awake.\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.")

```

(continues on next page)

(continued from previous page)

```
        myAdxl.standby()        # Must be in standby before changing settings.
                                                # This is here just in case we already␣
↪had sensor powered and/or
                                                # configured from a previous setup.

        myAdxl.setRange(myAdxl.ADXL313_RANGE_4_G)

        # setup activity sensing options
        myAdxl.setActivityX(True)                       # enable x-axis participation in␣
↪detecting activity
        myAdxl.setActivityY(False)                       # disable y-axis participation in␣
↪detecting activity
        myAdxl.setActivityZ(False)                       # disable z-axis participation in␣
↪detecting activity
        myAdxl.setActivityThreshold(10)        # 0-255 (62.5mg/LSB)

        # setup inactivity sensing options
        myAdxl.setInactivityX(True)                        # enable x-axis participation␣
↪in detecting inactivity
        myAdxl.setInactivityY(False)                # disable y-axis participation in␣
↪detecting inactivity
        myAdxl.setInactivityZ(False)                # disable z-axis participation in␣
↪detecting inactivity
        myAdxl.setInactivityThreshold(10)        # 0-255 (62.5mg/LSB)
        myAdxl.setTimeInactivity(5)                        # 0-255 (1sec/LSB)

        # Interrupt Mapping
        # when activity of inactivity is detected, it will effect the int1 pin on the␣
↪sensor
        myAdxl.setInterruptMapping(myAdxl.ADXL313_INT_ACTIVITY_BIT, myAdxl.ADXL313_INT1_
↪PIN)
        myAdxl.setInterruptMapping(myAdxl.ADXL313_INT_INACTIVITY_BIT, myAdxl.ADXL313_
↪INT1_PIN)

        myAdxl.ActivityINT(1)
        myAdxl.InactivityINT(1)
        myAdxl.DataReadyINT(0)

        myAdxl.autosleepOn()

        myAdxl.measureModeOn()

        # print int enable statuses, to verify we're setup correctly
        print("activity int enable: ", myAdxl.isInterruptEnabled(myAdxl.ADXL313_INT_
↪ACTIVITY_BIT))
        print("inactivity int enable: ", myAdxl.isInterruptEnabled(myAdxl.ADXL313_INT_
↪INACTIVITY_BIT))
        print("dataReady int enable: ", myAdxl.isInterruptEnabled(myAdxl.ADXL313_INT_
↪DATA_READY_BIT))
        time.sleep(5)

        while True:
```

(continues on next page)

(continued from previous page)

```
                myAdxl.updateIntSourceStatuses() # this will update all INTSOURCE␣
↪statuses.

                if myAdxl.ADXL313_INTSOURCE_INACTIVITY:
                        print("Inactivity detected.")
                        time.sleep(1)
                if myAdxl.ADXL313_INTSOURCE_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)\
                        )
                else:
                        print("Device is asleep (dataReady is reading false)")
                time.sleep(0.05)

if __name__ == '__main__':
        try:
                runExample()
        except (KeyboardInterrupt, SystemExit) as exErr:
                print("\nEnding Example 1")
                sys.exit(0)


```

6.8 Example 7: FIFO

Listing 7: examples/ex7_qwiic_adxl313_fifo.py

```
#!/usr/bin/env python
#----------------------------------------------------------------------------
# ex7_qwiic_adxl313_fifo.py
#
# Simple Example for the Qwiic ADXL313 DeviceSet that shows how to setup the FIFO on the␣
↪ADXL313.
# One key advantage of using the FIFO is that it allows us to
# let the ADXL313 store up to 32 samples in it's FIFO "buffer".
# While it is doing this, we can use our microcontroller to do other things,
# Then, when the FIFO is full (or close to), we can quickly read in the 32 samples.

# In order to use the FIFO in this way, we need to set it up to fire an interrupt
# when it gets "almost full". This threshold of samples is called the "watermark".
# When the watermark level is reached, it will fire the interrupt INT1.
# Our raspi will be monitoring the watermark int source bit, and then quickly
# read whatevers in the FIFO and save it to a log file.
# Note, we can't print the data in real time to the terminal
# because python terminal is too slow.

```

(continues on next page)

(continued from previous page)

```
# Some timestamps of each stage of this cycle will also be printed.
# This will allow us to fine tune bandwidth and watermark settings.
# The "Entries" of the FIFO_STATUS register will also be printed before each read.
# This will tell us how many samples are currently held in the FIFO.
# This will allow us to read the entire contents and keep an eye on how full it is
# getting before each read. This will help us fine tune how much time we have
# between each read to do other things. (in this example, we are simplly going to do
# a delay and print dots, but you could choose to do more useful things).

# **SPI app note***
# Note, this example uses I2C to communicate the the sensor.
# If you are going to use SPI, then you will need to add in a sufficient
# delay in between reads (at least 5uSec), to allow the FIFO to "pop" the next
# reading in the data registers. See datasheet page 16 for more info.

# ///// FIFO setup //////
# Stream mode
# Trigger INT1, Note, this example does not utilize monitoring this hardware interrupt.
# We will be monitoring via software by reading the int source and watching the
# watermark bit.
# Watermark Threshold (aka (samples in FIFO_CTL register)): 30

# ///// Interrupt setup //////
# Enable watermark interrupt.
# Map watermark interrupt to "int pin 1".
# This harware interrupt pin setup could be monitored by a GPIO on the raspi,
# or external system, however, for this example, we will simply
# poll the interrupt register via software to monitor its status.

#------------------------------------------------------------------------
#
# Written by  SparkFun Electronics, October 2020
#
# This python library supports the SparkFun Electroncis qwiic
# qwiic sensor/board ecosystem on a Raspberry Pi (and compatable) single
# board computers.
#
# More information on qwiic is at https://www.sparkfun.com/qwiic
#
# Do you like this library? Help support SparkFun. Buy a board!
#
#==================================================================================
# Copyright (c) 2019 SparkFun Electronics
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
```

(continues on next page)

(continued from previous page)

```
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#==================================================================================
# Example 7
#

from __future__ import print_function
import qwiic_adxl313
import time
import sys

lastWatermarkTime = 0 # used for printing timestamps in debug
fifoEntriesAmount = 0 # used to know how much is currently in the fifo and make sure to read it all out.

def micros():
        return round(time.time_ns()/1000)

# Open a log file in "append mode", We must log data here because printing to terminal is too slow
logfile = open("log.txt","a")

def runExample():

        print("\nSparkFun Adxl313  Example 7 - FIFO reading with debug info about timing.\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.standby()        # Must be in standby before changing settings.
                                                # This is here just in case we already had sensor powered and/or
                                                # configured from a previous setup.

        myAdxl.setRange(myAdxl.ADXL313_RANGE_4_G)

        # set bandwidth
        # note, 12.5Hz was chosen for this example to highlight the FIFO wait/read cycle
```

(continues on next page)

(continued from previous page)

```
        # you can tweak BW and the fifo sample threshhold to suit your application.
        myAdxl.setBandwidth(myAdxl.ADXL313_BW_12_5)
        # also try:
        # myAdxl.bandwidth = myAdxl.ADXL313_BW_12_5

        # setup activity sensing options
        myAdxl.setActivityX(False)                # disable x-axis participation in␣
↪detecting activity
        myAdxl.setActivityY(False)                # disable y-axis participation in␣
↪detecting activity
        myAdxl.setActivityZ(False)                # disable z-axis participation in␣
↪detecting activity

        # setup inactivity sensing options
        myAdxl.setInactivityX(False)                # disable x-axis participation in␣
↪detecting inactivity
        myAdxl.setInactivityY(False)                # disable y-axis participation in␣
↪detecting inactivity
        myAdxl.setInactivityZ(False)                # disable z-axis participation in␣
↪detecting inactivity

        # FIFO SETUP
        myAdxl.setFifoMode(myAdxl.ADXL313_FIFO_MODE_STREAM)
        myAdxl.setFifoSamplesThreshhold(30) # can be 1-32

        # Interrupt Mapping
        # when fifo fills up to watermark level, it will effect the int1 pin on the␣
↪sensor
        myAdxl.setInterruptMapping(myAdxl.ADXL313_INT_WATERMARK_BIT, myAdxl.ADXL313_INT1_
↪PIN)

        # enable/disable interrupts
        # note, we set them all here, just in case there were previous settings,
        # that need to be changed for this example to work properly.
        myAdxl.ActivityINT(0)                # disable activity
        myAdxl.InactivityINT(0)                # disable inactivity
        myAdxl.DataReadyINT(0)                # disable dataready
        myAdxl.WatermarkINT(1)                # enable watermark

        myAdxl.autosleepOff()                # just in case it was set from a previous␣
↪setup

        myAdxl.measureModeOn()                # wakes up sensor from stanby and puts␣
↪into measurement mode

        # print int enable statuses, to verify we're setup correctly
        print("activity int enable: ", myAdxl.isInterruptEnabled(myAdxl.ADXL313_INT_
↪ACTIVITY_BIT))
        print("inactivity int enable: ", myAdxl.isInterruptEnabled(myAdxl.ADXL313_INT_
↪INACTIVITY_BIT))
        print("dataReady int enable: ", myAdxl.isInterruptEnabled(myAdxl.ADXL313_INT_
↪DATA_READY_BIT))
```

(continues on next page)

(continued from previous page)

```
        print("FIFO watermark int enable: ", myAdxl.isInterruptEnabled(myAdxl.ADXL313_INT_WATERMARK_BIT))
        print("FIFO watermark Samples Threshhold: ", myAdxl.getFifoSamplesThreshhold())
        print("FIFO mode: ", myAdxl.getFifoMode())

        lastWatermarkTime = micros()

        myAdxl.clearFifo() # clear FIFO for a fresh start on this example.
        # The FIFO may have been full from previous use
        # and then would fail to cause an interrupt when starting this example.

        uSecTimer = 0 # used to print some "dots" during down time in cycle
        while True:
                myAdxl.updateIntSourceStatuses() # this will update all INTSOURCE statuses.
                if myAdxl.ADXL313_INTSOURCE_WATERMARK:
                        entries = myAdxl.getFifoEntriesAmount()
                        timegap_us = (micros() - lastWatermarkTime)
                        timegap_ms = round(timegap_us / 1000)

                        print("\nWatermark Interrupt! Time since last read: ", timegap_us, "us ", timegap_ms, "ms Entries:", entries)
                        lastWatermarkTime = micros()
                        while entries > 0:
                                myAdxl.updateIntSourceStatuses() # this will update all INTSOURCE statuses.
                                if myAdxl.ADXL313_INTSOURCE_DATAREADY:
                                        myAdxl.readAccel() # read all axis from sensor, note this also updates all instance variables

                                        # Gotta log data to a text file, because printing to terminal is too slow
                                        logfile.write(str(myAdxl.x))
                                        logfile.write("\t")
                                        logfile.write(str(myAdxl.y))
                                        logfile.write("\t")
                                        logfile.write(str(myAdxl.z))
                                        logfile.write("\n")
                                        entries -= 1 # we've read one more entry, so let's keep track and keep going until we're done
                                else:
                                        print("Waiting for Data.")

                time.sleep(0.000001) # sleep 1 microsecond
                uSecTimer += 1
                if uSecTimer > 100:
                        print(".", end = '')
                        uSecTimer = 0


if __name__ == '__main__':
        try:
```

(continues on next page)

(continued from previous page)

```
                runExample()
        except (KeyboardInterrupt, SystemExit) as exErr:
                print("\nEnding Example 1")
                logfile.close()
                sys.exit(0)


```

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX