
sparkfun_as6212_py

Release 0.0.2

SparkFun Electronics

Aug 27, 2021

CONTENTS:

- 1 Contents 3**
- 2 Supported Platforms 5**
- 3 Dependencies 7**
- 4 Documentation 9**
- 5 Installation 11**
 - 5.1 PyPi Installation 11
- 6 Example Use 13**
- 7 Table of Contents 15**
 - 7.1 API Reference 15
 - 7.1.1 qwiic_as6212 15
 - 7.2 Example 1 - Basic Readings 17
 - 7.3 Example 2 - Single Shot 20
- 8 Indices and tables 23**
- Python Module Index 25**
- Index 27**

Python module for the [SparkFun Digital Temperature Sensor Breakout - AS6212 \(Qwiic\)](#)

This python package is a port of the existing [SparkFun AS6212 Qwiic 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

- *Supported Platforms*
- *Dependencies*
- *Installation*
- *Documentation*
- *Example Use*

SUPPORTED PLATFORMS

This Python package currently supports the following platforms:

- [Raspberry Pi](#)

DEPENDENCIES

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

DOCUMENTATION

This module documentation is hosted at [ReadTheDocs](#)

INSTALLATION

5.1 PyPi Installation

This repository is hosted on PyPi as the [sparkfun-qwiic-as6212](#) 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-as6212
```

For the current user:

```
pip install sparkfun-qwiic-as6212
```

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-as6212-<version>.tar.gz
```


EXAMPLE USE

See the examples directory for more detailed use examples.

```
from __future__ import print_function
import qwiic_as6212
import time
import sys

def runExample():

    print("\nSparkFun Qwiic AS6212 Sensor Example 1\n")
    myTempSensor = qwiic_as6212.QwiicAs6212Sensor()

    if myTempSensor.is_connected == False:
        print("The Qwiic AS6212 Sensor device isn't connected to the system. Please_
↪check your connection", \
            file=sys.stderr)
        return

    myTempSensor.begin()
    time.sleep(1)

    print("Initialized.")

    # Initialize configuration settings
    # These settings are saved in the sensor, even if it loses power

    # set the number of consecutive faults before triggering alarm.
    # valid options: 1,2,3 or 4
    myTempSensor.set_consecutive_faults(1)

    # set the polarity of the Alert. (0:Active LOW, 1:Active HIGH).
    myTempSensor.set_alert_polarity(myTempSensor.AS6212_ALERT_ACTIVE_LOW)

    # set the sensor in Comparator Mode (0) or Interrupt Mode (1).
    myTempSensor.set_interrupt_mode(myTempSensor.AS6212_MODE_COMPARATOR)

    # set the Conversion Cycle Time (how quickly the sensor gets a new reading)
    myTempSensor.set_conversion_cycletime(myTempSensor.AS6212_CONVERSION_CYCLE_TIME_
↪250MS)
```

(continues on next page)

(continued from previous page)

```

# set T_HIGH, the upper limit to trigger the alert on
myTempSensor.set_high_temp_f(78.0) # set T_HIGH in F
# myTempSensor.set_high_temp_c(25.56) # set T_HIGH in C

# set T_LOW, the lower limit to shut turn off the alert
myTempSensor.set_low_temp_f(75.0) # set T_LOW in F
# myTempSensor.set_low_temp_c(23.89) # set T_LOW in C

print("TLOW F: ", myTempSensor.read_low_temp_f())
print("THIGH F: ", myTempSensor.read_high_temp_f())

while True:
    myTempSensor.set_sleep_mode(0) # turn sleep mode off (0)
    time.sleep(0.250) # allow time to wake up and complete first conversion

    temperature = myTempSensor.read_temp_f()

    # Check for alert
    alertRegisterState = myTempSensor.get_alert_status() # read the Alert
    ↪from register

    # Place sensor in sleep mode to save power.
    # Current consumption typically ~0.1uA.
    myTempSensor.set_sleep_mode(1) # turn sleep mode on (1)

    print("Temperature: ", temperature, "\tAlert Register: ", alertRegisterState)
    time.sleep(1)

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

```

TABLE OF CONTENTS

7.1 API Reference

7.1.1 qwiic_as6212

Python module for the [SparkFun Digital Temperature Sensor Breakout - AS6212 (Qwiic)](<https://www.sparkfun.com/products/18521>)

This python package is a port of the existing [SparkFun Qwiic AS6212 Sensor Arduino Library](https://github.com/sparkfun/SparkFun_TMP102_Arduino_Library/tree/master/examples)

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_as6212.QwiicAs6212Sensor(*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 AS6212 Sensor device object.

Return type Object

begin()

Initialize the operation of the Soil Moisture Sensor module :return: Returns true if the initialization was successful, otherwise False. :rtype: bool

property connected

Determine if a Soil MoistureSensor device is connected to the system.. :return: True if the device is connected, otherwise False. :rtype: bool

get_address()

Returns the device address

get_alert_polarity()

Get the polarity of Alert AS6212_ALERT_ACTIVE_HIGH (1) AS6212_ALERT_ACTIVE_LOW (0)

get_alert_status()

Get the status of the alert bit (0 or 1)

get_consecutive_faults()

Gets the number of consecutive faults that need to happen in a row before alert is changed. valid settings are 1,2,3 or 4 but this correspond to other bit values in the configuration register bits 11 and 12

get_conversion_cycletime()

Gets the conversion cycle time (aka conversion rate) in teh config reg Returns the cycle time in milliseconds: (125/250/1000/4000)

get_interrupt_mode()

Get the interrupt mode bit AS6212_MODE_COMPARATOR (0) AS6212_MODE_INTERRUPT (1)

get_single_shot_status()

gets the status of the single shot bit from the config register 0 = No conversion ongoing / conversion finished
1 = Start single shot conversion / conversion ongoing

get_sleep_mode()

gets the status of the sleep mode bit from the config register

is_connected()

Determine if a Soil MoistureSensor device is conntected to the system.. :return: True if the device is connected, otherwise False. :rtype: bool

read_2_byte_register(*register_to_read*)

Reads two bytes of data from a desired register. Combines them into a single 16 bit value Returns single value

read_high_temp_c()

Gets T_HIGH (degrees C) alert threshold

read_high_temp_f()

Reads T_HIGH register in F

read_low_temp_c()

Gets T_LOW (degrees C) alert threshold

read_low_temp_f()

Reads T_LOW register in F

read_temp_c()

Reads the results from the sensor :rtype: integer

read_temp_f()

Reads the results from the sensor :rtype: integer

set_alert_polarity(*polarity*)

Set the polarity of Alert AS6212_ALERT_ACTIVE_HIGH (1) AS6212_ALERT_ACTIVE_LOW (0)

set_consecutive_faults(*faults*)

Set the number of consecutive faults 1 - 1 fault 2 - 2 faults 3 - 3 faults 4 - 4 faults

set_conversion_cycletime(*cycletime*)

sets the conversion cylce time (aka conversion rate) in the config register valid settings are:

AS6212_CONVERSION_CYCLE_TIME_125MS AS6212_CONVERSION_CYCLE_TIME_250MS

AS6212_CONVERSION_CYCLE_TIME_1000MS AS6212_CONVERSION_CYCLE_TIME_4000MS

set_high_temp_c(*temperature*)

Sets THIGH (degrees C) alert threshold

set_high_temp_f(*temperature*)

Sets T_HIGH (degrees F) alert threshold

set_interrupt_mode(mode)

sets the interrupt mode bits in the config register

valid options are: AS6212_MODE_COMPARATOR (0) AS6212_MODE_INTERRUPT (1)

set_low_temp_c(temperature)

Sets T_LOW (degrees C) alert threshold

set_low_temp_f(temperature)

Sets T_LOW (degrees F) alert threshold

set_sleep_mode(mode)

sets the sleep mode bit (on or off) in the config register

valid options are: 0 = SLEEP MODE OFF 1 = SLEEP MODE ON

trigger_single_shot_conversion()

Sets the SS mode bit in the config register Note, you must be in sleep mode for this to work

7.2 Example 1 - Basic Readings

Listing 1: examples/Example_01_BasicReadings.py

```

1  #!/usr/bin/env python
2  #-----
3  # Example_01_BasicReadings.py
4  #
5  # Simple Example for the Qwiic AS6212 Device
6  #-----
7  #
8  # Written by Pete Lewis, SparkFun Electronics, Aug 2021
9  #
10 # Thanks to Alex Wende and Lori Croster @ SparkFun Electronics
11 # for code examples from TMP102 Python Package, May 2021
12 # (https://github.com/sparkfun/Qwiic_TMP102_Py)
13 #
14 # Thanks to Brandon Williams. This library was based off his
15 # original library created 07/15/2020 and can be found here:
16 # https://github.com/will2055/AS6212-Arduino-Library/
17 #
18 # Thanks to Madison Chodikov @ SparkFun Electronics
19 # for code examples from TMP117 Arduino Library
20 # (https://github.com/sparkfun/SparkFun_TMP117_Arduino_Library)
21 #
22 # This python library supports the SparkFun Electronics qwiic
23 # qwiic sensor/board ecosystem on a Raspberry Pi (and compatible) single
24 # board computers.
25 #
26 # This python library supports the SparkFun Electronics qwiic
27 # qwiic sensor/board ecosystem on a Raspberry Pi (and compatible) single
28 # board computers.
29 #
30 # More information on qwiic is at https://www.sparkfun.com/qwiic
31 #

```

(continues on next page)

(continued from previous page)

```

32 # Do you like this library? Help support SparkFun. Buy a board!
33 #
34 #=====
35 # Copyright (c) 2021 SparkFun Electronics
36 #
37 # Permission is hereby granted, free of charge, to any person obtaining a copy
38 # of this software and associated documentation files (the "Software"), to deal
39 # in the Software without restriction, including without limitation the rights
40 # to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
41 # copies of the Software, and to permit persons to whom the Software is
42 # furnished to do so, subject to the following conditions:
43 #
44 # The above copyright notice and this permission notice shall be included in all
45 # copies or substantial portions of the Software.
46 #
47 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
48 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
49 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
50 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
51 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
52 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
53 # SOFTWARE.
54 #=====
55 # Example 1
56 #
57
58 from __future__ import print_function
59 import qwiic_as6212
60 import time
61 import sys
62
63 def runExample():
64
65     print("\nSparkFun Qwiic AS6212 Sensor Example 1\n")
66     myTempSensor = qwiic_as6212.QwiicAs6212Sensor()
67
68     if myTempSensor.is_connected == False:
69         print("The Qwiic AS6212 Sensor device isn't connected to the system.␣
↪Please check your connection", \
70             file=sys.stderr)
71         return
72
73     myTempSensor.begin()
74     time.sleep(1)
75
76     print("Initialized.")
77
78     # Initialize configuration settings
79     # These settings are saved in the sensor, even if it loses power
80
81     # set the number of consecutive faults before triggering alarm.
82     # valid options: 1,2,3 or 4

```

(continues on next page)

(continued from previous page)

```

83     myTempSensor.set_consecutive_faults(1)
84
85     # set the polarity of the Alert. (0:Active LOW, 1:Active HIGH).
86     myTempSensor.set_alert_polarity(myTempSensor.AS6212_ALERT_ACTIVE_LOW)
87
88     # set the sensor in Comparator Mode (0) or Interrupt Mode (1).
89     myTempSensor.set_interrupt_mode(myTempSensor.AS6212_MODE_COMPARATOR)
90
91     # set the Conversion Cycle Time (how quickly the sensor gets a new reading)
92     myTempSensor.set_conversion_cycletime(myTempSensor.AS6212_CONVERSION_CYCLE_TIME_
↪ 250MS)
93
94     # set T_HIGH, the upper limit to trigger the alert on
95     myTempSensor.set_high_temp_f(78.0) # set T_HIGH in F
96     # myTempSensor.set_high_temp_c(25.56) # set T_HIGH in C
97
98     # set T_LOW, the lower limit to shut turn off the alert
99     myTempSensor.set_low_temp_f(75.0) # set T_LOW in F
100    # myTempSensor.set_low_temp_c(23.89) # set T_LOW in C
101
102    print("TLOW F: ", myTempSensor.read_low_temp_f())
103    print("THIGH F: ", myTempSensor.read_high_temp_f())
104
105    while True:
106        myTempSensor.set_sleep_mode(0) # turn sleep mode off (0)
107        time.sleep(0.250) # allow time to wake up and complete first conversion
108
109        temperature = myTempSensor.read_temp_f()
110
111        # Check for alert
112        alertRegisterState = myTempSensor.get_alert_status() #
↪ read the Alert from register
113
114        # Place sensor in sleep mode to save power.
115        # Current consumption typically ~0.1uA.
116        myTempSensor.set_sleep_mode(1) # turn sleep mode on (1)
117
118        print("Temperature: ", temperature, "\tAlert Register: ",
↪ alertRegisterState)
119        time.sleep(1)
120
121    if __name__ == '__main__':
122        try:
123            runExample()
124        except (KeyboardInterrupt, SystemExit) as exErr:
125            print("\nEnding Example 1")
126            sys.exit(0)

```

7.3 Example 2 - Single Shot

Listing 2: examples/Example_02_SingleShot.py

```

1  #!/usr/bin/env python
2  #-----
3  # Example_02_SingleShot.py
4  #
5  # Simple Example for the Qwiic AS6212 Device
6  #
7  # This example uses the Single Shot Feature of the device.
8  # It puts the sensor into sleep mode, and then in order to take
9  # each reading, it calls the trigger_single_shot_conversion() function.
10 # This allows us to take single readings on demand and really
11 # keep power use to a minimum.
12 #
13 # Note, in the basic readings example, we are "waking up" the sensor
14 # (by turning sleep mode off), and then it enters continuous reading mode,
15 # and so the sensor will continue to make conversions at the set conversion cycle time_
  ↳ (4Hz).
16 # This uses more power, but can be useful if you want to setup an alert, and can
17 # be even finer tuned by setting up the amount of desired consecutive faults.
18 #
19 # Note, using single shot readings like in this example, can also
20 # allow you to poll the SS bit (and know when the conversion is complete)
21 # SS bit = 0 = No conversion ongoing / conversion finished
22 # SS bit = 1 = Start single shot conversion / conversion ongoing
23 # This can allow you to immediate start another conversion, and increase
24 # the amount of conversions you demand.
25 #
26 # As the device exhibits a very short conversion time (~36ms-51ms), the effective_
  ↳ conversion
27 # rate can be increased by setting the single shot bit repetitively after a conversion_
  ↳ has finished.
28 # However, it has to be ensured that the additional power is limited, otherwise self-
  ↳ heating
29 # effects have to be considered.
30 #
31 #-----
32 #
33 # Written by Pete Lewis, SparkFun Electronics, Aug 2021
34 #
35 # Thanks to Alex Wende and Lori Croster @ SparkFun Electronics
36 # for code examples from TMP102 Python Package, May 2021
37 # (https://github.com/sparkfun/Qwiic\_TMP102\_Py)
38 #
39 # Thanks to Brandon Williams. This library was based off his
40 # original library created 07/15/2020 and can be found here:
41 # https://github.com/will2055/AS6212-Arduino-Library/
42 #
43 # Thanks to Madison Chodikov @ SparkFun Electronics
44 # for code examples from TMP117 Arduino Library

```

(continues on next page)

(continued from previous page)

```

45 # (https://github.com/sparkfun/SparkFun_TMP117_Arduino_Library)
46 #
47 # This python library supports the SparkFun Electronics qwiic
48 # qwiic sensor/board ecosystem on a Raspberry Pi (and compatible) single
49 # board computers.
50 #
51 # More information on qwiic is at https://www.sparkfun.com/qwiic
52 #
53 # Do you like this library? Help support SparkFun. Buy a board!
54 #
55 #=====
56 # Copyright (c) 2021 SparkFun Electronics
57 #
58 # Permission is hereby granted, free of charge, to any person obtaining a copy
59 # of this software and associated documentation files (the "Software"), to deal
60 # in the Software without restriction, including without limitation the rights
61 # to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
62 # copies of the Software, and to permit persons to whom the Software is
63 # furnished to do so, subject to the following conditions:
64 #
65 # The above copyright notice and this permission notice shall be included in all
66 # copies or substantial portions of the Software.
67 #
68 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
69 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
70 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
71 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
72 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
73 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
74 # SOFTWARE.
75 #=====
76 # Example 2
77 #
78
79 from __future__ import print_function
80 import qwiic_as6212
81 import time
82 import sys
83
84 def runExample():
85
86     print("\nSparkFun Qwiic AS6212 Sensor Example 2 - Single Shot Readings\n")
87     myTempSensor = qwiic_as6212.QwiicAs6212Sensor()
88
89     if myTempSensor.is_connected == False:
90         print("The Qwiic AS6212 Sensor device isn't connected to the system.␣
91 ↪Please check your connection", \
92             file=sys.stderr)
93         return
94
95     myTempSensor.begin()
96     time.sleep(1)

```

(continues on next page)

(continued from previous page)

```
96     print("Initialized.")
97
98     myTempSensor.set_sleep_mode(1) # turn sleep mode on (1)
99     print("Sleep mode ON")
100     time.sleep(1)
101
102     while True:
103         myTempSensor.trigger_single_shot_conversion() # trigger SS
104
105         #wait for conversion to complete (~51ms)
106         conversionTime = 0
107         while myTempSensor.get_single_shot_status() == 1:
108             conversionTime += 1
109             time.sleep(0.001) # 1ms
110
111         tempF = myTempSensor.read_temp_f()
112
113         print("Temperature: %.2fF \t Conversion time: %ims" % (tempF, ↵
↵conversionTime))
114         time.sleep(1)
115
116 if __name__ == '__main__':
117     try:
118         runExample()
119     except (KeyboardInterrupt, SystemExit) as exErr:
120         print("\nEnding Example 1")
121         sys.exit(0)
```

INDICES AND TABLES

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

PYTHON MODULE INDEX

q

`qwiic_as6212`, [15](#)

B

`begin()` (*qwiic_as6212.QwiicAs6212Sensor* method), 15

C

`connected` (*qwiic_as6212.QwiicAs6212Sensor* property), 15

G

`get_address()` (*qwiic_as6212.QwiicAs6212Sensor* method), 15

`get_alert_polarity()` (*qwiic_as6212.QwiicAs6212Sensor* method), 15

`get_alert_status()` (*qwiic_as6212.QwiicAs6212Sensor* method), 15

`get_consecutive_faults()` (*qwiic_as6212.QwiicAs6212Sensor* method), 15

`get_conversion_cycletime()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

`get_interrupt_mode()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

`get_single_shot_status()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

`get_sleep_mode()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

I

`is_connected()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

M

`module`
 qwiic_as6212, 15

Q

`qwiic_as6212`

`module`, 15

`QwiicAs6212Sensor` (class in *qwiic_as6212*), 15

R

`read_2_byte_register()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

`read_high_temp_c()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

`read_high_temp_f()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

`read_low_temp_c()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

`read_low_temp_f()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

`read_temp_c()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

`read_temp_f()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

S

`set_alert_polarity()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

`set_consecutive_faults()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

`set_conversion_cycletime()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

`set_high_temp_c()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

`set_high_temp_f()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

`set_interrupt_mode()` (*qwiic_as6212.QwiicAs6212Sensor* method), 16

`set_low_temp_c()` (*qwiic_as6212.QwiicAs6212Sensor* method), 17

`set_low_temp_f()` (*qwiic_as6212.QwiicAs6212Sensor* method), 17

`set_sleep_mode()` (*qwiic_as6212.QwiicAs6212Sensor*
method), [17](#)

T

`trigger_single_shot_conversion()`
(*qwiic_as6212.QwiicAs6212Sensor* *method*),
[17](#)