sparkfun_as6212_py

Release 0.0.2

SparkFun Electronics

Aug 27, 2021

CONTENTS:

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.

CHAPTER

ONE

CONTENTS

CHAPTER

TWO

SUPPORTED PLATFORMS

This Python package currently supports the following platforms:

- Raspberry Pi

CHAPTER

THREE

DEPENDENCIES

This driver package depends on the qwiic I2C driver: Qwiic_I2C_Py

CHAPTER

FOUR

DOCUMENTATION

This module documentation is hosted at ReadTheDocs

CHAPTER

FIVE

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
```

CHAPTER

SIX

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)
```

CHAPTER

SEVEN

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 of the initialization was successful, otherwise False. :rtype: bool

property connected
Determine if a Soil MoistureSensor device is conntected 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 convertion 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

```
#!/usr/bin/env python
#-----------------------------------------------------------------------------
# Example_01_BasicReadings.py
#
# Simple Example for the Qwiic AS6212 Device
#-----------------------------------------------------------------------
#
# Written by Pete Lewis, SparkFun Electronics, Aug 2021
#
# Thanks to Alex Wende and Lori Croster @ SparkFun Electronics
# for code examples from TMP102 Python Package, May 2021
# (https://github.com/sparkfun/Qwiic_TMP102_Py)
#
# Thanks to Brandon Williams. This library was based off his
# original library created 07/15/2020 and can be found here:
# https://github.com/will2055/AS6212-Arduino-Library/
#
# Thanks to Madison Chodikov @ SparkFun Electronics
# for code examples from TMP117 Arduino Library
# (https://github.com/sparkfun/SparkFun_TMP117_Arduino_Library)
#
# This python library supports the SparkFun Electroncis qwiic
# qwiic sensor/board ecosystem on a Raspberry Pi (and compatable) single
# board computers.
#
# 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
#
```

(continues on next page)

(continued from previous page)

```
# Do you like this library? Help support SparkFun. Buy a board!
#
#==================================================================================
# Copyright (c) 2021 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_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
```

(continues on next page)

(continued from previous page)

```
        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)

        # 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)
```

7.3 Example 2 - Single Shot

Listing 2: examples/Example_02_SingleShot.py

```
#!/usr/bin/env python
#-----------------------------------------------------------------------------
# Example_02_SingleShot.py
#
# Simple Example for the Qwiic AS6212 Device
#
# This example uses the Single Shot Feature of the device.
# It puts the sensor into sleep mode, and then in order to take
# each reading, it calls the trigger_single_shot_conversion() function.
# This allows us to take single readings on demand and really
# keep power use to a minimum.
#
# Note, in the basic readings example, we are "waking up" the sensor
# (by turning sleep mode off), and then it enters continuous reading mode,
# and so the sensor will continue to make conversions at the set conversion cycle time␣
↪(4Hz).
# This uses more power, but can be useful if you want to setup an alert, and can
# be even finer tuned by setting up the amount of desired consecutive faults.
#
# Note, using single shot readings like in this example, can also
# allow you to poll the SS bit (and know when the conversion is complete)
# SS bit = 0 = No conversion ongoing / conversion finished
# SS bit = 1 = Start single shot conversion / conversion ongoing
# This can allow you to immediate start another conversion, and increase
# the amount of conversions you demand.
#
# As the device exhibits a very short conversion time (~36ms-51ms), the effective␣
↪conversion
# rate can be increased by setting the single shot bit repetitively after a conversion␣
↪has finished.
# However, it has to be ensured that the additional power is limited, otherwise self-
↪heating
# effects have to be considered.
#
#-----------------------------------------------------------------------------
#
# Written by Pete Lewis, SparkFun Electronics, Aug 2021
#
# Thanks to Alex Wende and Lori Croster @ SparkFun Electronics
# for code examples from TMP102 Python Package, May 2021
# (https://github.com/sparkfun/Qwiic_TMP102_Py)
#
# Thanks to Brandon Williams. This library was based off his
# original library created 07/15/2020 and can be found here:
# https://github.com/will2055/AS6212-Arduino-Library/
#
# Thanks to Madison Chodikov @ SparkFun Electronics
# for code examples from TMP117 Arduino Library
```

(continues on next page)

(continued from previous page)

```
# (https://github.com/sparkfun/SparkFun_TMP117_Arduino_Library)
#
# 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) 2021 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 2
#

from __future__ import print_function
import qwiic_as6212
import time
import sys

def runExample():

        print("\nSparkFun Qwiic AS6212 Sensor Example 2 - Single Shot Readings\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)
```

(continues on next page)

(continued from previous page)

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

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

q

INDEX

T