sparkfun$_{q}wiic_{p}ca9685$

Release 0.0.1

Jul 17, 2020

Contents:

Python module for the PCA9685 PWM controller, which is part of the SparkFun Servo pHAT for Raspberry Pi and SparkFun Pi Servo HAT

This package should be used in conjunction with the overall SparkFun qwiic Python Package. New to qwiic? Take a look at the entire SparkFun qwiic ecosystem.

CHAPTER 1

Contents

CHAPTER 2

Supported Platforms

The qwiic PCA9685 Python package current supports the following platforms:

- Raspberry Pi <!– Platforms to be tested
- NVidia Jetson Nano
- Google Coral Development Board –>

CHAPTER 3

Dependencies

This package depends on the qwiic I2C driver: Qwiic_I2C_Py

CHAPTER 4

Documentation

The SparkFun qwiic PCA9685 module documentation is hosted at ReadTheDocs

CHAPTER 5

Installation

5.1 PyPi Installation

This repository is hosted on PyPi as the sparkfun-qwiic-pca9685 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-pca9685
```

For the current user:

```
pip install sparkfun-qwiic-pca9685
```

5.2 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_pca9685-<version>.tar.gz
```

CHAPTER 6

Example Use

See the examples directory for more detailed use examples.

```
import qwiic_pca9685
import time
import math
import sys

def runExample():

    print("\nSparkFun PCA9685 Example 1\n")
    mySensor = qwiic_pca9685.QwiicPCA9685()

    if mySensor.isConnected() == False:
        print("The Qwiic PCA9685 device isn't connected to the system. Please check␣
↪your connection", \
            file=sys.stderr)
        return

    mySensor.begin()

    # Sets PWM Frequency to 50 Hz
    mySensor.set_pre_scale(50)

    # Sets start time of PWM pulse on Channel 0 to 0s
    mySensor.set_channel_word(0, 1, 0)


    while True:
        # Increments start time of PWM pulse on Channel 0 to i (1ms to 2ms)
        for i in range(205, 410):
            fun.set_channel_word(0, 0, i)
            # Delay .05 s
            time.sleep(.1)
```

(continues on next page)

(continued from previous page)

```
        # Decrements start time of PWM pulse on Channel 0 to i (2ms to 1ms)
        for i in range(410, 205, -1):
            fun.set_channel_word(0, 0, i)
            # Delay .05 s
            time.sleep(.1)

        time.sleep(1)
```

CHAPTER 7

Table of Contents

7.1 API Reference

7.1.1 qwiic_pca9685

Python module for the [SparkFun Pi Servo HAT](https://www.sparkfun.com/products/14328) and [SparkFun Servo pHAT for Raspberry Pi](https://www.sparkfun.com/products/15316) This package should be used in conjunction with the i2c_driver package contained in the [SparkFun qwiic Python Package](https://github.com/sparkfun/Qwiic_Py)

class qwiic_pca9685.**QwiicPCA9685**(*address=None*, *debug=None*, *i2c_driver=None*)

SparkFunPCA9685 Initialise the PCA9685 chip at `address` with `i2c_driver`.

param address The I2C address to use for the device. If not provided, the default address is used.

param i2c_driver An existing i2c driver object. If not provided a driver object is created.

return Constructor Initialization True- Successful False- Issue loading I2C driver

rtype Bool

begin()

This method checks if there is an I2C connection then enables the Auto-Increment bit for the writing/reading of words (for the output timing).

Returns Function Operation True- Successful False- Issue in Execution

Return type Bool

clear_restart_bit()

If RESTART bit is a logic 1, it clears RESTART bit in MODE 1 register by writing a logic 1.

Returns Value of RESTART bit after changes. 0- Restart Disabled (Default) 1- Restart Enabled

Return type Integer

NOTE: Other actions that will clear the RESTART bit are:

1. Power cycle.
2. I2C Software Reset command.
3. If the MODE2 OCH bit is logic 0, write to any PWM register then issue an I2C-bus STOP.
4. If the MODE2 OCH bit is logic 1, write to all four PWM registers in any PWM channel.

Likewise, if the user does an orderly shutdown [1] of all the PWM channels before setting the SLEEP bit, the RESTART bit will be cleared. If this is done the contents of all PWM registers are invalidated and must be reloaded before reuse.

[1] Two methods can be used to do an orderly shutdown. The fastest is to write a logic 1 to bit 4 in register ALL_LED_OFF_H. The other method is to write logic 1 to bit 4 in each active PWM channel LEDn_OFF_H register.

enable_extclock_bit()

Changes value of EXTCLK bit in MODE 1 register to enable EXTCLK pin. Once enabled, it allows for an external clock signal. It also affects the refresh rate:

$$\text{refresh\_rate} = \frac{\text{EXTCLK}}{4096 \times (\text{prescale} + 1)}$$

NOTE: This EXTCLK bit is a "sticky bit", that is, it cannot be cleared by writing a logic 0 to it. The EXTCLK can only be cleared by a power cycle or software reset.

get_addr_bit(*addr_bit=None*)

Reads the value of a specified address bit in MODE 1 register.

Parameters **addr_bit** – Specify address bit. 0- ALLCALL Bit (Default) 1- SUB1 Bit 2- SUB2 Bit 3- SUB3 Bit

Returns

Value of specified address bit. 0- Normal Mode 1- Low Power Mode; Oscillator Off.

(Default)

Return type Integer

get_auto_increment_bit()

Reads the value of AI bit in MODE 1 register. When enabled, it allows users to write of multiple bytes (i.e. words).

Returns Value of AI bit. 0- Auto-Increment Disabled (Default) 1- Auto-Increment Enabled

Return type Integer

get_channel_word(*channel=None*, *on_off=None*)

Reads the ON/OFF timing for the specified PWM channel.

Parameters

- **channel** – PWM channel. 0 to 16
- **on_off** – On or Off setting. 0- OFF Start timing (end of ON timing) 1- ON Start timing (anything greater than 0 is

considered a delay)

Returns Word (2 bytes)

NOTE: There are two 12-bit registers per LED output. Both registers will hold a value from 0 to 4095. One 12-bit register will hold a value for the ON time and the other 12-bit register will hold the value for the OFF time.

The ON and OFF times are compared with the value of a 12-bit counter that will be running continuously from 0000h to 0FFFh (0 to 4095 decimal).

get_extclock_bit()

Reads the value of EXTCLK bit in MODE 1 register. When enabled, it allows for an external clock signal. It also affects the refresh rate:

EXTCLK

refresh_rate = ——————- 4096 x (prescale +1)

Returns Value of EXTCLK bit. 0- Use Internal Clock (Default) 1- Use EXTCLK Pin Clock.

Return type Integer

NOTE: This bit is a "sticky bit", that is, it cannot be cleared by writing a logic 0 to it. The EXTCLK can only be cleared by a power cycle or software reset.

get_invrt_bit()

Reads the value of INVRT bit in MODE 2 register. Determines how the outputs are driven. See Section 7.7 "Using the PCA9685 with and without external drivers" of the datasheet.

Returns Value of INVRT bits. 0- Outputs change on STOP command. 1- Outputs change on ACK.

Return type Integer

get_och_bit()

Reads the value of OCH bit in MODE 2 register. Determines when the outputs change.

Returns

Value of OCH bits. 0- Outputs change on STOP command.

NOTE: Change of the outputs at the STOP command allows synchronizing outputs of more than one PCA9685. Applicable to registers from 06h (LED0_ON_L) to 45h (LED15_OFF_H) only. 1 or more registers can be written, in any order, before STOP.

1- Outputs change on ACK. NOTE: Update on ACK requires all 4 PWM channel registers to be loaded before outputs will change on the last ACK.

Return type Integer

get_outdrv_bit()

Reads the value of OUTDRV bit in MODE 2 register. Determines how the outputs are driven.

Returns

Value of OUTDRV bits. 0- Outputs are configured with an open-drain

structure

1- Outputs are configured with a totem-pole structure

Return type Integer

get_outne_bits()
Reads the value of OUTNE bits in MODE 2 register. When the active LOW output (OE pin) is enabled, this setting allows users to enable or disable all the LED outputs at the same time.

Returns

Value of OUTNE bits. When OE = 1: 0- LEDn = 0 1- If OUTDRV = 1 then LEDn = 1

If OUTDRV = 0 then LEDn = high-impedence (same as OUTNE[1:0] = b‘10’)

2 to 3- LEDn = high-impedence

Return type Integer

get_pre_scale()
Reads the frequency at which the output are modulated. The prescale value is determined by Eq 1 (below).

Eq 1: osc_clock

prescale value = round(————————————-) - 1 (4096 * update_rate)

Returns prescale_value

Return type Integer

NOTE: Range: 24 Hz to 1526 Hz or (0x03 to 0xFF, Default: 0x1E = 200Hz).

get_restart_bit()
Reads the value of RESTART bit in MODE 1 register.

Returns Value of Restart Mode bit. 0- Restart Disabled (Default) 1- Restart Enabled

Return type Integer

get_sleep_bit()
Reads the value of SLEEP bit in MODE 1 register. When enabled, it the chip there is no PWM control.

Returns Value of SLEEP bit. 0- Normal Mode 1- Low Power Mode; Oscillator Off. (Default)

Return type Integer

is_connected()
This method checks if the “i2c_driver” can connect to the device at the specified or default address.

Returns Device Connection True- Successful False- Can’t find device

Return type Bool

restart()
Restarts the PCA9685 after the soft reset. (Clears MODE1 register.)

restart_pwm_channels()
Restarts all of the previously active PWM channels with a few I2C-bus cycles.

Returns Value of RESTART bit after changes. 0- Restart Disabled (Default) 1- Restart Enabled

Return type Integer

NOTE: Only if the PCA9685 was operating and the user put the chip to sleep (setting MODE1 bit 4) without stopping any of the PWM channels, the RESTART bit (MODE1 bit 7) will be set to logic 1 at the end of the PWM refresh cycle. The contents of each PWM register are held valid when the clock is off.

Uses the following steps:

1. Read MODE1 register.

2. Check that bit 7 (RESTART) is a logic 1. If it is, clear bit 4 (SLEEP). Allow time for the oscillator to stabilize (500us).
3. Write logic 1 to bit 7 of MODE1 register. All PWM channels will restart and the RESTART bit will clear.

Remark: The SLEEP bit must be logic 0 for at least 500us, before a logic 1 is written into the RESTART bit.

set_addr_bit(*addr_bit*, *value*)
Writes the value to a specified address bit in MODE 1 register.

Parameters

- **addr_bit** – Specify address bit. 0- ALLCALL Bit 1- SUB1 Bit 2- SUB2 Bit 3- SUB3 Bit
- **value** – Specify address bit. 0- Disables specified address (Default on

  SUB1, SUB2, SUB3)

  1- Enables specified address (Default on ALLCALL)

set_auto_increment_bit(*value=None*)
Changes value of AI bit in MODE 1 register. When enabled, it allows users to write of multiple bytes (i.e. words).

Parameters **value** – Value to set AI bit. 0- Auto-Increment Disabled (Default) 1- Auto-Increment Enabled

set_channel_word(*channel=None*, *on_off=None*, *value=None*)
Configures the on/off timing for the specified PWM channel.

Parameters

- **channel** – PWM channel. 0 to 16
- **on_off** – ON/OFF setting. 0- OFF Start timing (end of ON timing) 1- ON Start timing (anything greater than 0 is

  considered a delay)
- **value** – Value to be entered into the ON/OFF 12-bit register for the specified LED output. Word (2 bytes)

Returns Function Operation True- Successful False- Issue in Execution

Return type Bool

NOTE: There are two 12-bit registers per LED output. Both registers will hold a value from 0 to 4095. One 12-bit register will hold a value for the ON time and the other 12-bit register will hold the value for the OFF time.

The ON and OFF times are compared with the value of a 12-bit counter that will be running continuously from 0000h to 0FFFh (0 to 4095 decimal).

set_invrt_bit(*value=None*)
Configures value of INVRT bits in MODE 2 register. Configures how the outputs are driven. See Section 7.7 “Using the PCA9685 with and without external drivers” of the datasheet.

Parameters **value** – Value of INVRT bits. 0- Outputs change on STOP command. 1- Outputs change on ACK.

set_och_bit(*value=None*)
Reads the value of OCH bits in MODE 2 register. Configures when the outputs change.

Parameters **value** – Value of OCH bits. 0- Outputs change on STOP command.

NOTE: Change of the outputs at the STOP command allows synchronizing outputs of more than one PCA9685. Applicable to registers from 06h (LED0_ON_L) to 45h (LED15_OFF_H) only. 1 or more registers can be written, in any order, before STOP.

1- Outputs change on ACK. NOTE: Update on ACK requires all 4 PWM channel registers to be loaded before outputs will change on the last ACK.

set_outdrv_bit(*value=None*)
Reads value of OUTDRV bits in MODE 2 register. Configures how the outputs are driven.

Parameters **value** – Value of OUTDRV bits. 0- Outputs are configured with an open-drain

structure

1- Outputs are configured with a totem-pole structure

set_outne_bit(*value=None*)
Reads the value of OUTNE bits in MODE 2 register. When the active LOW output (OE pin) is enabled, this setting allows users to enable or disable all the LED outputs at the same time.

Parameters **value** – Value of OUTNE bits. When OE = 1: 0- LEDn = 0 1- If OUTDRV = 1 then LEDn = 1

If OUTDRV = 0 then LEDn = high-impedence (same as OUTNE[1:0] = b‘10’)

2 to 3- LEDn = high-impedence

Returns Function Operation True- Successful False- Issue in Execution

Return type Bool

set_pre_scale(*frequency=None*, *ext=None*)
Configures the ‘prescale_value’, which defines the frequency at which the output are modulated. The prescale value is determined by Eq 1 (below). Additionally, the hardware enforces a minimum value that can be loaded into this register is ‘3’.

Eq 1: osc_clock

prescale value = round(————————-) - 1 (4096 * update_rate)

Parameters

- **frequency** – PWM Frequency (Hz) Range: 24 to 1526 Hz
- **ext** – External Clock Frequency (Hz) Default = None; uses internal clock frequency (25 MHz)

Returns Function Operation True- Successful False- Issue in Execution

Return type Bool

PWM Frequency Range: 24 Hz to 1526 Hz or (0x03 to 0xFF, Default: 0x1E = 200Hz) Internal Clock: 25 MHz (Default)

set_sleep_bit(*value=None*)
Changes the value of SLEEP bit in MODE 1 register.

Parameters **value** – Value to set SLEEP bit. 0- Normal Mode 1- Low Power Mode; Oscillator Off. (Default)

soft_reset()

Software Reset Call: Allows all the devices in the I2C bus to be reset to the power-up state value through a specific formatted I2C bus command.

General Call Address

SWRST data byte 1

Start | | Stop

| | |

[S][0000 0000][A][0000 0110][A][P]

|

Acknowledge from Slave | Acknowledge from Slave

PCA9685 then resets to the default value (power-up value) and is ready to be addressed again within the specified bus free time. A failure or non-acknowledge from the PCA9685 (at any time) should be interpreted as an "SWRST Call Abort".

write_restart_bit(*value=None*)

Writes values to RESTART bit in MODE 1 register.

Parameters **value** – Value to write to Restart Mode bit. 0 or 1

Returns Value of RESTART bit after changes. 0- Restart Disabled (Default) 1- Restart Enabled

Return type Integer

NOTE: Value aren't set, just written. The bit is set by the state of the chip and its current operation.

7.2 Basic Operation

Listing 1: examples/qwiic_pca9685_ex1.py

```
import qwiic_pca9685
import time
import math
import sys

def runExample():

        print("\nSparkFun BME280 Sensor  Example 1\n")
        mySensor = qwiic_pca9685.QwiicPCA9685()

        if mySensor.isConnected() == False:
                print("The Qwiic PCA9685 device isn't connected to the system. Please check your connection", \
                        file=sys.stderr)
                return

        mySensor.begin()

```

(continues on next page)

(continued from previous page)

```
        # Sets PWM Frequency to 50 Hz
        mySensor.set_pre_scale(50)

        # Sets start time of PWM pulse on Channel 0 to 0s
        mySensor.set_channel_word(0, 1, 0)


        while True:
                # Increments start time of PWM pulse on Channel 0 to i (1ms to 2ms)
                for i in range(205, 410):
                        fun.set_channel_word(0, 0, i)
                        # Delay .05 s
                        time.sleep(.05)

                # Decrements start time of PWM pulse on Channel 0 to i (2ms to 1ms)
                for i in range(410, 205, -1):
                        fun.set_channel_word(0, 0, i)
                        # Delay .05 s
                        time.sleep(.05)

                time.sleep(1)
```

Python Module Index

q

Index

W