qwiic$_i2c$

Release 0.9.4

Aug 08, 2023

Contents:

Python package to support multi platform I2C bus integrations for the SparkFun qwiic ecosystem

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 1

Contents

CHAPTER 2

Supported Platforms

The qwiic I2C Python package current supports the following platforms:

- Raspberry Pi (Single Board Computers)
- NVidia Jetson Nano
- Google Coral Development Board

CHAPTER 3

Dependencies

The Raspberry Pi/Single Board Computer Linux driver of this package is dependent on smbus

CHAPTER 4

Documentation

The SparkFun qwiic I2C module documentation is hosted at ReadTheDocs

CHAPTER 5

Installation

5.1 PyPi Installation

This repository is hosted on PyPi as the sparkfun-qwiic-i2c 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-i2c
```

For the current user:

```
pip install sparkfun-qwiic-i2c
```

CHAPTER 6

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_i2c-<version>.tar.gz
```

CHAPTER 7

Examples

This package is used extensively by the python modules for the SparkFun qwiic ecosystem. References to the modules can be found in the qwiic python package

General package use examples:

```
import qwiic_i2c
connectedDevices = i2cDriver.scan()
if myDeviceAddress in connectedDevices:
    with qwiic_i2c.getI2CDriver() as i2c:
        i2c.writeByte(myDeviceAddress, register, 0x3F)
```

```
import qwiic_i2c
>>> if qwiic_i2c.isDeviceConnected(myDeviceAddress):
        with qwiic_i2c.getI2CDriver() as i2c:
                i2c.writeByte(myDeviceAddress, register, 0x3F)
```

CHAPTER 8

Table of Contents

8.1 API Reference

8.1.1 qwiic_i2c

A package to abstract the interface to the platform specific I2C bus calls. This package is part of the python package for SparkFun qwiic ecosystem.

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

example

```
>>> import qwiic_i2c
>>> connectedDevices = i2cDriver.scan()
>>> if myDeviceAddress in connectedDevices:
        with qwiic_i2c.getI2CDriver() as i2c:
                i2c.writeByte(myDeviceAddress, register, 0x3F)
```

example

```
>>> import qwiic_i2c
>>> if qwiic_i2c.isDeviceConnected(myDeviceAddress):
        with qwiic_i2c.getI2CDriver() as i2c:
                i2c.writeByte(myDeviceAddress, register, 0x3F)
```

`qwiic_i2c.`**`getI2CDriver`**`()`

`qwiic_i2c.`**`getI2CDriver`**`()`

Returns the qwiic I2C driver object for current platform.

Returns A qwiic I2C driver object for the current platform.

Return type object

Example

```
>>> import qwiic_i2c
>>> i2cDriver = qwiic_i2c.getI2CDriver()
>>> myData = i2cDriver.readByte(0x73, 0x34)
```

`qwiic_i2c.`**`isDeviceConnected`**`(`*devAddress*`)`

`qwiic_i2c.`**`isDeviceConnected`**`()`

Function to determine if a particular device (at the provided address) is connected to the bus.

Parameters **`devAddress`** – The I2C address of the device to check

Returns True if the device is connected, otherwise False.

Return type bool

`class` `qwiic_i2c.`**`I2CDriver`**

Implements the interface for the I2C bus for the qwiic ecosystem.

Returns The I2C Driver interface for the qwiic system.

Return type Object

`classmethod isPlatform``()`

Called to determine if the specific driver supports the current platorm

Returns True if this platform is supported, otherwise False

Return type bool

`readBlock``(`*address*, *commandCode*, *nBytes*`)`

Called to read a block of bytesfrom a specific device.

Parameters

- **`address`** – The I2C address of the device to read from
- **`commandCode`** – The "command" or register to read from
- **`nBytes`** – The number of bytes to read from the device

Returns Returns the read data as a list of integers.

Return type list

`readByte``(`*address*, *commandCode*`)`

Called to read a byte (8 bits) from a specific device.

Parameters

- **`address`** – The I2C address of the device to read from
- **`commandCode`** – The "command" or register to read from

Returns Returns the read data

Return type integer - first 8 bits contain the read data.

`readWord``(`*address*, *commandCode*`)`

Called to read a word (16 bits) from a specific device.

Parameters

- **`address`** – The I2C address of the device to read from
- **`commandCode`** – The "command" or register to read from

Returns Returns the read data

Return type integer - first 16 bits contain the read data.

`classmethod scan``()`
Used to scan the I2C bus, returning a list of I2C address attached to the computer.

Returns A list of I2C addresses. If no devices are attached, an empty list is returned.

Return type list

`writeBlock``(`*address*, *commandCode*, *value*`)`
Called to write a block of bytes to a device.

Parameters

- **`address`** – The I2C address of the device to read from
- **`commandCode`** – The "command" or register to read from
- **`value`** – A list of bytes (ints) to write on the I2C bus.

Returns None

`writeByte``(`*address*, *commandCode*, *value*`)`
Called to write a byte (8 bits) to a device.

Parameters

- **`address`** – The I2C address of the device to read from
- **`commandCode`** – The "command" or register to read from
- **`value`** – The byte (8 bits) to write to the I2C bus

Returns None

`writeCommand``(`*address*, *commandCode*`)`
Called to write a command to a device. No actual data is written

Parameters

- **`address`** – The I2C address of the device to read from
- **`commandCode`** – The "command" or register to read from

Returns None

`writeWord``(`*address*, *commandCode*, *value*`)`
Called to write a word (16 bits) to a device.

Parameters

- **`address`** – The I2C address of the device to read from
- **`commandCode`** – The "command" or register to read from
- **`value`** – The word (16 bits) to write to the I2C bus

Returns None

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index