sparkfun_eeprom_py

Release 0.0.1

SparkFun Electronics

Jul 26, 2021

CONTENTS:

Python module for the SparkFun Qwiic EEPROM Breakout - 512Kbit

This python package is a port of the existing SparkFun External EEPROM 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

The Qwiic EEPROM 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

The SparkFun Qwiic EEPROM module documentation is hosted at ReadTheDocs

CHAPTER

FIVE

INSTALLATION

5.1 PyPi Installation

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

For the current user:

```
pip install sparkfun-qwiic-eeprom
```

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

CHAPTER

SIX

EXAMPLE USE

See the examples directory for more detailed use examples.

```
from __future__ import print_function
import qwiic_eeprom
import time
import sys

def run_example():

    print("\nSparkFun Qwiic EEPROM, Example 1\n")
    my_eeprom = qwiic_eeprom.QwiicEEPROM()

    if my_eeprom.begin() != True:
        print("\nThe Qwiic EEPROM isn't connected to the system. Please check your␣
↪connection", \
            file=sys.stderr)
        return

    print("\nEEPROM ready!")

    print("\nMem size in bytes: " + str(my_eeprom.length()))

    my_value1 = 200 # 0xC8
    my_eeprom.write_byte(300, my_value1)  # (location = 0x12C, data byte)
    my_read1 = my_eeprom.read_byte(300) # (location = 0x12C)
    print("\nI read: " + str(my_read1)) # Should be 0xC8 = 200

    my_value2 = -366
    my_eeprom.write_int(10, my_value2)
    my_read2 = my_eeprom.read_int(10)    # (location)
    print("\nI read: " + str(my_read2))

    my_value3 = -7.35
    my_eeprom.write_float(9, my_value3)
    my_read3 = my_eeprom.read_float(9)    # (location)
    # Round to 2-decimal point precision
    my_read3 = round(my_read3, 2)
    print("\nI read: " + str(my_read3))

if __name__ == '__main__':
```

(continues on next page)

(continued from previous page)

```
    try:
        run_example()
    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_eeprom

Python module for the SparkFun Qwiic EEPROM Breakout - 512Kbit.

This package is a port of the exisiting [SparkFun External EEPROM Arduino Library](https://github.com/sparkfun/SparkFun_External_EEPROM_Arduino_Library).

This package can be used in conjuction 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_eeprom.**QwiicEEPROM**(*address=None*, *i2c_driver=None*)

Qwiic EEPROM

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 a driver object is created.

return The GPIo device object.

rtype Object

begin()

Initialize the operation of the Qwiic EEPROM. Run is_connected().

Returns Returns true if the initialization was successful, false otherwise.

Return type bool

disable_poll_for_write_complete()

Disable polling of when a write has completed

Returns Nothing

Return type Void

enable_poll_for_write_complete()

Enable I2C polling of when a write has completed

Returns Nothing

Return type Void

erase(*to_write=0*)

Erase entire EEPROM.

Parameters **to_write** – byte to write into each spot of EEPROM

Returns Nothing

Return type void

get_I2C_buffer_size()
Return the size of the TX buffer

Returns I2C_BUFFER_LENGTH_TX

Return type int

get_memory_size()
Return the size of EEPROM

Returns memory_size_bytes

Return type int

get_page_size()
Get the page size

Returns Current page size off EEPROM in bytes

Return type int

get_page_write_time()
Get the current time required per page write

Returns Time required per page write

Return type int

is_busy(*i2c_address=255*)
Returns true if the device is not answering (currently writing).

Parameters **i2c_address** – I2C address of EEPROM. Larger EEPROMs have two addresses.

Returns True if the IC is busy, false otherwise

Return type bool

is_connected(*i2c_address=255*)
Determine if a Qwiic EEPROM device is connected to the system

Parameters **i2c_address** – I2C address of EEPROM. Larger EEPROMs have two addresses.

Returns True if the device is connected, false otherwise.

Return type bool

length()
Returns the memory size of the EEPROM

Returns memory_size_bytes

Return type int

read(*eeprom_location*, *num_bytes*)
Bulk read from EEPROM. Handles breaking up read amt into 32 byte chunks (can be overidden with set_I2C_buffer_size() Handles a read that straddles the 512kbit barrier

Parameters

- **eeprom_location** – address of EEPROM to start reading from
- **num_bytes** – number of bytes to be read from external EEPROM

Returns a list of bytes read from EEPROM

Return type list

`read_byte`(*eeprom_location*)
Read exactly one byte from EEPROM at a given address location

Parameters `eeprom_location` – location in EEPROM to read byte from

Returns byte read from EEPROM

Return type byte

`read_float`(*eeprom_location*)
Read a 32-bit float from a given EEPROM location

Parameters `eeprom_location` – location in EEPROM to read float from

Returns float read from EEPROM

Return type float

`read_int`(*eeprom_location*)
Read a 32-bit signed int from a given EEPROM location

Parameters `eeprom_location` – location in EEPROM to read int from

Returns int read from EEPROM

Return type int

`read_string`(*eeprom_location*, *string_length*)
Read a stromg of given length from any address of EEPROM

Param eeprom_location: location in EEPROM to read string from

Parameters `string_length` – number of chars to read from EEPROM

Returns string read from EEPROM

Return type string

`set_I2C_buffer_size`(*buff_size*)
Set the size of the TX buffer

Parameters `buff_size` – the size of the I2C buffer

Returns nothing

Return type Void

`set_memory_size`(*mem_size*)
Set the size of memory in bytes

Parameters `mem_size` – memory size in bytes

Returns Nothing

Return type void

`set_page_size`(*page_size*)
Set the size of the page we can write at a time

Parameters `page_size` – new page size in bytes

Returns Nothing

Return type void

`set_page_write_time`(*write_time_ms*)
Set the number of ms required per page write

Parameters **write_time_ms** – write time in ms

Returns Nothing

Return type Void

`write`(*eeprom_location*, *data_list*)
Write large bulk amounts to EEPROM. Limits write to the I2C buffer size (default is 32 bytes).

Parameters

- **eeprom_location** – 2-byte EEPROM address to write to
- **data_list** – list of data bytes to be written to EEPROM sequentially, starting at the EEPROM address

Return type Void

Returns nothing

`write_byte`(*eeprom_location*, *byte_to_write*)
Write a single byte to given EEPROM location

Parameters

- **eeprom_location** – location in EEPROM to byte to
- **byte_to_write** – byte to write to EEPROM

Returns Nothing

Return type Void

`write_float`(*eeprom_location*, *float_to_write*)
Write a 32-bit float to a given EEPROM location

Parameters

- **eeprom_location** – location in EEPROM to write float to
- **float_to_write** – float to write to EEPROM

Returns Nothing

Return type Void

`write_int`(*eeprom_location*, *int_to_write*)
Write a signed 32-bit int to a given EEPROM location

Parameters

- **eeprom_location** – location in EEPROM to write int to
- **int_to_write** – int to write to EEPROM

Returns Nothing

Return type Void

`write_string`(*eeprom_location*, *string_to_write*)
Write a stirng to a given EEPROM location

Parameters

- **eeprom_location** – location in EEPROM to write string to

- **string_to_write** – string to write to EEPROM

Returns Nothing

Return type Void

7.2 Example One - Basic Read Write

Listing 1: examples/qwiic_eeprom_ex1_basic_read_write.py

```
# !/usr/bin/env python
# ------------------------------------------------------------------------
# qwiic_eeprom_ex1_basic_read_write.py
#
# This example demonstrates how to read and write various variables to memory.
# ------------------------------------------------------------------------
#
# Written by Priyanka Makin @ SparkFun Electronics, July 2021
#
# This python library supports the SparkFun Electronics 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 by buying 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
```

(continues on next page)

(continued from previous page)

```
import qwiic_eeprom
import time
import sys

def run_example():

    print("\nSparkFun Qwiic EEPROM, Example 1\n")
    my_eeprom = qwiic_eeprom.QwiicEEPROM()

    if my_eeprom.begin() != True:
        print("\nThe Qwiic EEPROM isn't connected to the system. Please check your␣
↪connection", \
            file=sys.stderr)
        return

    print("\nEEPROM ready!")

    print("\nMem size in bytes: " + str(my_eeprom.length()))

    my_value1 = 200 # 0xC8
    my_eeprom.write_byte(300, my_value1)  # (location = 0x12C, data byte)
    my_read1 = my_eeprom.read_byte(300) # (location = 0x12C)
    print("\nI read: " + str(my_read1)) # Should be 0xC8 = 200

    my_value2 = -366
    my_eeprom.write_int(10, my_value2)
    my_read2 = my_eeprom.read_int(10)    # (location)
    print("\nI read: " + str(my_read2))

    my_value3 = -7.35
    my_eeprom.write_float(9, my_value3)
    my_read3 = my_eeprom.read_float(9)    # (location)
    # Round to 2-decimal point precision
    my_read3 = round(my_read3, 2)
    print("\nI read: " + str(my_read3))

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

```

7.3 Example Two - Settings

Listing 2: examples/qwiic_eeprom_ex2_settings.py

```
# !/usr/bin/env python
# ------------------------------------------------------------------------
# qwiic_eeprom_ex2_settings.py
#
# This example demonstrates how to set the various settings for a given EEPROM.
# Read the datasheet! Each EEPROM will have specific values for:
# Overall EEPROM size in bytes (512kbit = 64000, 256kbit = 32000)
# Bytes per page write (64 and 128 are common)
# Whether write polling is supported
# ------------------------------------------------------------------------
#
# Written by Priyanka Makin @ SparkFun Electronics, July 2021
#
# This python library supports the SparkFun Electronics 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 by buying 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_eeprom
import time
```

(continues on next page)

(continued from previous page)

```
import sys

def run_example():

    print("\nSparkFun Qwiic EEPROM, Example 2\n")
    my_eeprom = qwiic_eeprom.QwiicEEPROM()

    if my_eeprom.begin() != True:
        print("\nThe Qwiic EEPROM isn't connected to the system. Please check your␣
↪connection", \
            file=sys.stderr)
        return

    print("\nEEPROM ready!")

    # Set settings for this EEPROM
    my_eeprom.set_memory_size(512000/8) # In bytes. 512kbit = 64kbyte
    my_eeprom.set_page_size(128)    # in bytes. Has 128 byte page size.
    my_eeprom.disable_poll_for_write_complete()  # Supports I2C polling of write␣
↪completion
    my_eeprom.set_page_write_time(3)    # 3 ms max write time

    print("\nMem size in bytes: " + str(my_eeprom.get_memory_size()))
    print("\nPage size in bytes: " + str(my_eeprom.get_page_size()))

    my_value = -7.35
    my_eeprom.write_float(20, my_value) # (location, data)
    my_read = my_eeprom.read_float(20)  # (location)
    # Round to 2-decimal point precision
    my_read = round(my_read, 2)
    print("\nI read: " + str(my_read))

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

7.4 Example Three - Advanced I2C

Listing 3: examples/qwiic_eeprom_ex3_advanced_i2c.py

```
# !/usr/bin/env python
# ------------------------------------------------------------------------
# qwiic_eeprom_ex3_advanced_i2c.py
#
# This example demonstrates how to pass a custom EEPROM address.
# ------------------------------------------------------------------------
#
```

(continues on next page)

(continued from previous page)

```
# Written by Priyanka Makin @ SparkFun Electronics, July 2021
#
# This python library supports the SparkFun Electronics 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 by buying 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 3

from __future__ import print_function
import qwiic_eeprom
import time
import sys

def run_example():

    EEPROM_ADDRESS = 0b1010001 # 0b1010(A2 A1 A0): A standard I2C EEPROM with the ADR0␣
↪bit set to VCC

    print("\nSparkFun Qwiic EEPROM, Example 3\n")
    my_eeprom = qwiic_eeprom.QwiicEEPROM(0x51)

    if my_eeprom.begin() != True:
        print("\nThe Qwiic EEPROM isn't connected to the system. Please check your␣
↪connection", \
            file=sys.stderr)
        return
```

(continues on next page)

(continued from previous page)

```
    print("\nEEPROM ready!")

    my_value = -7.35
    my_eeprom.write_float(20, my_value)   # (location, data)
    my_read = my_eeprom.read_float(20) # (location to read)
    # Round to 2-decimal point precision
    my_read = round(my_read, 2)
    print("\nI read: " + str(my_read))

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

7.5 Example Four - User Options

Listing 4: examples/qwiic_eeprom_ex4_user_options.py

```
# !/usr/bin/env python
# ------------------------------------------------------------------------
# qwiic_eeprom_ex4_user_options.py
#
# This example demonstrates how to record various user settings easily
# to EEPROM.
# ------------------------------------------------------------------------
#
# Written by Priyanka Makin @ SparkFun Electronics, July 2021
#
# This python library supports the SparkFun Electronics 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 by buying 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:
#
```

(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 4

from __future__ import print_function
import qwiic_eeprom
import time
import sys
# TODO: remove this?
from collections import namedtuple
from dataclasses import *

LOCATION_SETTINGS = 0
SETTINGS_SIZE = 10 # 2 ints + 2 bools = 10 bytes total

# Load the current settings from EEPROM into the settings struct
def load_user_settings(my_mem, settings):
    # Uncomment these lines to forcibly erase the EEPROM and see how the defaults are set
    # print("\nErasing EEPROM, be patient please")
    # my_mem.erase()

    # Check to see if EEPROM is blank. If the first four spots are zeros then we can␣
↪assume the EEPROM is blank.
    num_bytes = 4
    if (my_mem.read_int(LOCATION_SETTINGS) == 0):     # (EEPROM address to read, thing␣
↪to read to)
        # At power on, settings are set to defaults within the tuple
        # So go record the tuple as it currently exists so that defaults are set
        record_user_settings(my_mem, settings)

        print("\nDefault settings applied")
    else:

        settings.baud_rate == my_mem.read_int(0)
        settings.log_date == my_mem.read_byte(4)
        settings.enable_IMU == my_mem.read_byte(5)
        cal_val_temp = my_mem.read_float(6)
        # Round to 2-decimal point precision
        settings.cal_value == round(cal_val_temp, 2)

# Record the current settings into EEPROM
def record_user_settings(my_mem, settings):
    # Use our individual write functions to make writing this class easier
```

(continues on next page)

(continued from previous page)

```
    my_mem.write_int(0, settings.baud_rate)
    time.sleep(0.1)
    my_mem.write_byte(4, settings.log_date)
    time.sleep(0.1)
    my_mem.write_byte(5, settings.enable_IMU)
    time.sleep(0.1)
    my_mem.write_float(6, settings.cal_value)
    time.sleep(0.1)

def run_example():

    print("\nSparkFun Qwiic EEPROM, Example 4\n")
    my_eeprom = qwiic_eeprom.QwiicEEPROM()

    if my_eeprom.begin() != True:
        print("\nThe Qwiic EEPROM isn't connected to the syste. Please check your␣
↪connection", \
            file=sys.stderr)
        return

    print("\nEEPROM ready!")

    @dataclass
    class Settings:
        baud_rate: int
        log_date: bool
        enable_IMU: bool
        cal_value: float

    default_settings = Settings(115200, False, True, -5.17)

    load_user_settings(my_eeprom, default_settings)

    print("\nBaud rate: " + str(default_settings.baud_rate))

    log_str = ""
    if default_settings.log_date == True:
        log_str = "True"
    else:
        log_str = "False"
    print("\nlog_date: " + log_str)

    print("\ncal_value: " + str(default_settings.cal_value))

    # Now we can change something
    print("\nEnter a new baud rate (1200 to 115200): ")
    new_baud = input()
    if int(new_baud) < 1200 or int(new_baud) > 115200:
        print("\nError: baud rate out of range!")
    else:
        default_settings = replace(default_settings, baud_rate = int(new_baud))

```

(continues on next page)

(continued from previous page)

```
        print("\nThis is the new baud rate: ")
        print(default_settings.baud_rate)

        record_user_settings(my_eeprom, default_settings)

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

```

7.6 Example Five - Interface Test

Listing 5: examples/qwiic_eeprom_ex5_interface_test.py

```
# !/usr/bin/env python
# ------------------------------------------------------------------------
# qwiic_eeprom_ex5_interface_test.py
#
# This example demonstrates how to read and write various variables to memory
# ------------------------------------------------------------------------
#
# Written by Priyanka Makin @ SparkFun Electronics, July 2021
#
# This python library supports the SparkFun Electronics 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 by buying 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
```

(continues on next page)

(continued from previous page)

```
# 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_eeprom
import time
import sys
import random

def run_example():

    print("\nSparkFun Qwiic EEPROM, Example 5\n")
    my_eeprom = qwiic_eeprom.QwiicEEPROM()

    if my_eeprom.begin() != True:
        print("\nThe Qwiic EEPROM isn't connected to the system. Please check your␣
↪connection", \
            file=sys.stderr)
        return

    print("\nEEPROM ready!")

    my_eeprom.set_memory_size(51200 / 8)    # Qwiic EEPROM is 24512C (512kbit)
    # my_eeprom.set_page_size(128)
    # my_eeprom.disable_poll_for_write_complete()

    all_tests_passed = True

    print("\nMem size in bytes: " + str(my_eeprom.length()))

    # Erase test
    # -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
    print("\nErasing EEPROM, please be patient!")
    start_time = time.time()
    my_eeprom.erase()
    end_time = time.time()
    print("\nTime to erase all EEPROM: " + str(end_time  - start_time) + "s")

    # Byte sequential test
    # -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
    print("\n")
    print("\n8 bit tests")
    my_value1 = 200
    my_value2 = 23
    random_location = random.randrange(0, my_eeprom.length())

    start_time = time.time()
```

(continues on next page)

(continued from previous page)

```
    my_eeprom.write_byte(random_location, my_value1)   # (location, data)
    end_time = time.time()
    my_eeprom.write_byte(random_location + 1, my_value2)
    print("\nTime to record byte: " + str(end_time - start_time) + " s")

    start_time = time.time()
    my_eeprom.write_byte(random_location, my_value1)    # (location, data)
    end_time = time.time()
    print("\nTime to write identical byte to same location (should be ~0s): " + str(end_time - start_time) + " s")

    start_time = time.time()
    response1 = my_eeprom.read_byte(random_location)
    end_time = time.time()
    print("\nTime to read byte: " + str(end_time - start_time) + " s")

    response2 = my_eeprom.read_byte(random_location + 1)
    print("\nLocation " + str(random_location) + " should be " + str(my_value1) + ": " + str(response1))
    print("\nLocation " + str(random_location + 1) + " should be " + str(my_value2) + ": " + str(response2))

    if my_value1 != response1:
        all_tests_passed = False
    if my_value2 != response2:
        all_tests_passed = False

    # 32 bit test
    # -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
    print("\n")
    print("\n32 bit tests")

    my_value3 = -245000
    my_value4 = 400123
    random_location = random.randrange(0, my_eeprom.length())

    start_time = time.time()
    my_eeprom.write_int(random_location, my_value3)
    end_time = time.time()
    print("\nTime to record int32: " + str(end_time - start_time) + " s")
    my_eeprom.write_int(random_location + 4, my_value4)

    start_time = time.time()
    response3 = my_eeprom.read_int(random_location)
    end_time = time.time()
    print("\nTime to read 32 bits: " + str(end_time - start_time) + " s")

    response4 = my_eeprom.read_int(random_location + 4)
    print("\nLocation " + str(random_location) + " should be " + str(my_value3) + ": " + str(response3))
    print("\nLocation " + str(random_location + 4) + " should be " + str(my_value4) + ": " + str(response4))
```

(continues on next page)

(continued from previous page)

```python

    if my_value3 != response3:
        all_tests_passed = False
    if my_value4 != response4:
        all_tests_passed = False

    # 32 bit sequential test
    # -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
    my_value5 = -341002
    my_value6 = 241544
    random_location = random.randrange(0, my_eeprom.length())

    my_eeprom.write_int(random_location, my_value5)
    my_eeprom.write_int(random_location + 4, my_value6)

    start_time = time.time()
    response5 = my_eeprom.read_int(random_location)
    end_time = time.time()
    print("\nTime to read 32 bits: " + str(end_time - start_time) + " s")

    response6 = my_eeprom.read_int(random_location + 4)
    print("\nLocation " + str(random_location) + " should be " + str(my_value5) + ": " + str(response5))
    print("\nLocation " + str(random_location + 4) + " should be " + str(my_value6) + ": " + str(response6))

    if my_value5 != response5:
        all_tests_passed = False
    if my_value6 != response6:
        all_tests_passed = False

    # Float sequential test
    # -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
    my_value7 = -7.35
    my_value8 = 5.22
    random_location = random.randrange(0, my_eeprom.length())

    my_eeprom.write_float(random_location, my_value7)
    my_eeprom.write_float(random_location + 4, my_value8)

    start_time = time.time()
    response7 = my_eeprom.read_float(random_location)
    end_time = time.time()
    print("\nTime to read float: " + str(end_time - start_time) + " s")

    response8 = my_eeprom.read_float(random_location + 4)

    # Round floats read to 2-decimal point precision
    response7 = round(response7, 2)
    response8 = round(response8, 2)

    print("\nLocation " + str(random_location) + " should be " + str(my_value7) + ": " + str(response7))
```

(continues on next page)

(continued from previous page)

```
    print("\nLocation " + str(random_location + 4) + " should be " + str(my_value8) + ":
↪" + str(response8))

    if my_value7 != response7:
        all_tests_passed = False
    if my_value8 != response8:
        all_tests_passed = False

    # 64 bits sequential test
    # -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
    print("\n")
    print("\n64 bit tests")

    # Made up list of 64-bits
    my_value9 = [0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8]
    random_location = random.randrange(0, my_eeprom.length())

    start_time = time.time()
    my_eeprom.write(random_location, my_value9)
    end_time = time.time()
    print("\nTime to record 64 bits: " + str(end_time - start_time) + " s")

    start_time = time.time()
    response9 = my_eeprom.read(random_location, len(my_value9))
    end_time = time.time()
    print("\nTime to read 64 bits: " + str(end_time - start_time) + " s")

    print("\nLocation " + str(random_location) + " should be " + str(my_value9) + ": " +␣
↪str(response9))

    if my_value9 != response9:
        all_tests_passed = False

    # Buffer write test
    # -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
    print("\n")
    print("\nBuffer write test")

    my_chars = "Lorem ipsum dolor sit amet, has in verte rem accusamus. Nulla viderer␣
↪inciderint eum at."
    random_location = random.randrange(0, my_eeprom.length() - len(my_chars))

    est_time = len(my_chars) / my_eeprom.get_page_size() * my_eeprom.get_page_write_
↪time()
    print("\nCalculated time to record array of " + str(len(my_chars)) + " characters: ~
↪" + str(est_time) + " ms")

    start_time = time.time()
    my_eeprom.write_string(random_location, my_chars)
    end_time = time.time()
    print("\nTime to record string: " + str(end_time - start_time) + " s")

```

(continues on next page)

(continued from previous page)

```
    start_time = time.time()
    read_chars = my_eeprom.read_string(random_location, len(my_chars))
    end_time = time.time()
    print("\nTime to read string: " + str(end_time - start_time) + " s")

    print("\nLocation " + str(random_location) + " string should read:\n" + my_chars)
    print("\n" + read_chars)

    if my_chars != read_chars:
        print("\nString compare failed")
        all_tests_passed = False

    # # -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
    if all_tests_passed == True:
        print("\nAll tests PASSED!")
    else:
        print("\nOne or more tests failed. See output")

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

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

q

INDEX