Adafruit HT16K33 Library Documentation

Release 1.0

Radomir Dopieralski

Feb 17, 2020

Contents

This is a library for using the I^2C-based LED matrices with the HT16K33 chip. It supports both 16x8 and 8x8 matrices, as well as 7- and 14-segment displays.

- **Notes**

  1. This library is intended for Adafruit CircuitPython's API. For a library compatible with MicroPython machine API see this library.
  2. This library does not work with the Trellis 4x4 LED+Keypad board. For that product use: CircuitPython Trellis Library

CHAPTER 1

Dependencies

This driver depends on:

- Adafruit CircuitPython
- Bus Device

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the Adafruit library and driver bundle.

CHAPTER 2

Installing from PyPI

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally from PyPI. To install for current user:

```
pip3 install adafruit-circuitpython-ht16k33
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-ht16k33
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .env
source .env/bin/activate
pip3 install adafruit-circuitpython-ht16k33
```

CHAPTER 3

Usage Example

```
# Import all board pins and bus interface.
import board
import busio

# Import the HT16K33 LED matrix module.
from adafruit_ht16k33 import matrix

# Create the I2C interface.
i2c = busio.I2C(board.SCL, board.SDA)

# Create the matrix class.
# This creates a 16x8 matrix:
matrix = matrix.Matrix16x8(i2c)
# Or this creates a 8x8 matrix:
#matrix = matrix.Matrix8x8(i2c)
# Or this creates a 8x8 bicolor matrix:
#matrix = matrix.Matrix8x8x2
# Finally you can optionally specify a custom I2C address of the HT16k33 like:
#matrix = matrix.Matrix16x8(i2c, address=0x70)

# Clear the matrix.
matrix.fill(0)

# Set a pixel in the origin 0,0 position.
matrix[0, 0] = 1
# Set a pixel in the middle 8, 4 position.
matrix[8, 4] = 1
# Set a pixel in the opposite 15, 7 position.
matrix[15, 7] = 1
matrix.show()

# Change the brightness
matrix.brightness = 8
```

(continues on next page)

(continued from previous page)

```
# Set the blink rate
matrix.blink_rate = 2
```

CHAPTER 4

Contributing

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER 5

Documentation

For information on building library documentation, please check out this guide.

CHAPTER 6

Table of Contents

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/ht16k33_matrix_simpletest.py

```
# Basic example of clearing and drawing a pixel on a LED matrix display.
# This example and library is meant to work with Adafruit CircuitPython API.
# Author: Tony DiCola
# License: Public Domain

# Import all board pins.
import time
import board
import busio

# Import the HT16K33 LED matrix module.
from adafruit_ht16k33 import matrix


# Create the I2C interface.
i2c = busio.I2C(board.SCL, board.SDA)

# Create the matrix class.
# This creates a 16x8 matrix:
matrix = matrix.Matrix16x8(i2c)
# Or this creates a 16x8 matrix backpack:
# matrix = matrix.MatrixBackpack16x8(i2c)
# Or this creates a 8x8 matrix:
#matrix = matrix.Matrix8x8(i2c)
# Or this creates a 8x8 bicolor matrix:
#matrix = matrix.Matrix8x8x2(i2c)
# Finally you can optionally specify a custom I2C address of the HT16k33 like:
```

(continues on next page)

(continued from previous page)

```
#matrix = matrix.Matrix16x8(i2c, address=0x70)

# Clear the matrix.
matrix.fill(0)

# Set a pixel in the origin 0, 0 position.
matrix[0, 0] = 1
# Set a pixel in the middle 8, 4 position.
matrix[8, 4] = 1
# Set a pixel in the opposite 15, 7 position.
matrix[15, 7] = 1

time.sleep(2)

# Draw a Smiley Face
matrix.fill(0)

for row in range(2, 6):
    matrix[row, 0] = 1
    matrix[row, 7] = 1

for column in range(2, 6):
    matrix[0, column] = 1
    matrix[7, column] = 1

matrix[1, 1] = 1
matrix[1, 6] = 1
matrix[6, 1] = 1
matrix[6, 6] = 1
matrix[2, 5] = 1
matrix[5, 5] = 1
matrix[2, 3] = 1
matrix[5, 3] = 1
matrix[3, 2] = 1
matrix[4, 2] = 1

# Move the Smiley Face Around
while True:
    for frame in range(0, 8):
        matrix.shift_right(True)
        time.sleep(0.05)
    for frame in range(0, 8):
        matrix.shift_down(True)
        time.sleep(0.05)
    for frame in range(0, 8):
        matrix.shift_left(True)
        time.sleep(0.05)
    for frame in range(0, 8):
        matrix.shift_up(True)
        time.sleep(0.05)
```

Listing 2: examples/ht16k33_segments_simpletest.py

```
# Basic example of setting digits on a LED segment display.
# This example and library is meant to work with Adafruit CircuitPython API.
# Author: Tony DiCola
```

(continues on next page)

(continued from previous page)

```
# License: Public Domain

import time

# Import all board pins.
import board
import busio

# Import the HT16K33 LED segment module.
from adafruit_ht16k33 import segments

# Create the I2C interface.
i2c = busio.I2C(board.SCL, board.SDA)

# Create the LED segment class.
# This creates a 7 segment 4 character display:
display = segments.Seg7x4(i2c)
# Or this creates a 14 segment alphanumeric 4 character display:
#display = segments.Seg14x4(i2c)
# Or this creates a big 7 segment 4 character display
#display = segments.BigSeg7x4(i2c)
# Finally you can optionally specify a custom I2C address of the HT16k33 like:
#display = segments.Seg7x4(i2c, address=0x70)

# Clear the display.
display.fill(0)

# Can just print a number
display.print(42)
time.sleep(2)

# Or, can print a hexadecimal value
display.print_hex(0xFF23)
time.sleep(2)

# Or, print the time
display.print("12:30")
time.sleep(2)

display.colon = False

# Or, can set indivdual digits / characters
# Set the first character to '1':
display[0] = '1'
# Set the second character to '2':
display[1] = '2'
# Set the third character to 'A':
display[2] = 'A'
# Set the forth character to 'B':
display[3] = 'B'
time.sleep(2)

# Or, can even set the segments to make up characters
if isinstance(display, segments.Seg7x4):
    # 7-segment raw digits
    display.set_digit_raw(0, 0xFF)
    display.set_digit_raw(1, 0b11111111)
```

(continues on next page)

(continued from previous page)

```
    display.set_digit_raw(2, 0x79)
    display.set_digit_raw(3, 0b01111001)
else:
    # 14-segment raw digits
    display.set_digit_raw(0, 0x2D3F)
    display.set_digit_raw(1, 0b0010110100111111)
    display.set_digit_raw(2, (0b00101101, 0b00111111))
    display.set_digit_raw(3, [0x2D, 0x3F])
time.sleep(2)

#Show a looping marquee
display.marquee('Deadbeef 192.168.100.102... ', 0.2)
```

Listing 3: examples/ht16k33_bicolor24_simpletest.py

```
# Basic example of using the Bi-color 24 segment bargraph display.
# This example and library is meant to work with Adafruit CircuitPython API.
# Author: Carter Nelson
# License: Public Domain

import time

# Import board related modules
import board
import busio

# Import the Bicolor24 driver from the HT16K33 module
from adafruit_ht16k33.bargraph import Bicolor24

# Create the I2C interface
i2c = busio.I2C(board.SCL, board.SDA)

# Create the LED bargraph class.
bc24 = Bicolor24(i2c)

# Set individual segments of bargraph
bc24[0] = bc24.LED_RED
bc24[1] = bc24.LED_GREEN
bc24[2] = bc24.LED_YELLOW

time.sleep(2)

# Turn them all off
bc24.fill(bc24.LED_OFF)

# Turn them on in a loop
for i in range(24):
    bc24[i] = bc24.LED_RED
    time.sleep(0.1)
    bc24[i] = bc24.LED_OFF

time.sleep(1)

# Fill the entrire bargraph
bc24.fill(bc24.LED_GREEN)
```

6.2 adafruit_ht16k33.ht16k33

- Authors: Radomir Dopieralski & Tony DiCola for Adafruit Industries

class adafruit_ht16k33.ht16k33.**HT16K33**(*i2c*, *address=112*, *auto_write=True*)
The base class for all displays. Contains common methods.

Parameters

- **address** (*int*) – The I2C addess of the HT16K33.
- **auto_write** (*bool*) – True if the display should immediately change when set. If False, *show* must be called explicitly.

auto_write
Auto write updates to the display.

blink_rate
The blink rate. Range 0-3.

brightness
The brightness. Range 0-15.

fill(*color*)
Fill the whole display with the given color.

show()
Refresh the display and show the changes.

6.3 Matrix Displays

class adafruit_ht16k33.matrix.**Matrix16x8**(*i2c*, *address=112*, *auto_write=True*)
The matrix wing.

pixel(*x*, *y*, *color=None*)
Get or set the color of a given pixel.

class adafruit_ht16k33.matrix.**Matrix8x8**(*i2c*, *address=112*, *auto_write=True*)
A single matrix.

columns
Read-only property for number of columns

image(*img*)
Set buffer to value of Python Imaging Library image. The image should be in 1 bit mode and a size equal to the display size.

pixel(*x*, *y*, *color=None*)
Get or set the color of a given pixel.

rows
Read-only property for number of rows

shift(*x*, *y*, *rotate=False*)
Shift pixels by x and y

Parameters **rotate** – (Optional) Rotate the shifted pixels to the left side (default=False)

shift_down(*rotate=False*)
Shift all pixels down

Parameters **rotate** – (Optional) Rotate the shifted pixels to top (default=False)

shift_left(*rotate=False*)
Shift all pixels left

Parameters **rotate** – (Optional) Rotate the shifted pixels to the right side (default=False)

shift_right(*rotate=False*)
Shift all pixels right

Parameters **rotate** – (Optional) Rotate the shifted pixels to the left side (default=False)

shift_up(*rotate=False*)
Shift all pixels up

Parameters **rotate** – (Optional) Rotate the shifted pixels to bottom (default=False)

class adafruit_ht16k33.matrix.**Matrix8x8x2**(*i2c*, *address=112*, *auto_write=True*)
A bi-color matrix.

fill(*color*)
Fill the whole display with the given color.

image(*img*)
Set buffer to value of Python Imaging Library image. The image should be a size equal to the display size.

pixel(*x*, *y*, *color=None*)
Get or set the color of a given pixel.

class adafruit_ht16k33.matrix.**MatrixBackpack16x8**(*i2c*, *address=112*, *auto_write=True*)
A double matrix backpack.

pixel(*x*, *y*, *color=None*)
Get or set the color of a given pixel.

6.4 Segment Displays

class adafruit_ht16k33.segments.**BigSeg7x4**(*i2c*, *address=112*, *auto_write=True*)
Numeric 7-segment display. It has the same methods as the alphanumeric display, but only supports displaying a limited set of characters.

ampm
The AM/PM indicator.

bottom_left_dot
The bottom-left dot indicator.

top_left_dot
The top-left dot indicator.

class adafruit_ht16k33.segments.**Colon**(*disp*, *num_of_colons=1*)
Helper class for controlling the colons. Not intended for direct use.

class adafruit_ht16k33.segments.**Seg14x4**(*i2c*, *address=112*, *auto_write=True*)
Alpha-numeric, 14-segment display.

marquee(*text*, *delay=0.25*, *loop=True*)
Automatically scroll the text at the specified delay between characters

Parameters

- **text** (*str*) – The text to display

- **delay** (*float*) – (optional) The delay in seconds to pause before scrolling to the next character (default=0.25)
- **loop** (*bool*) – (optional) Whether to endlessly loop the text (default=True)

print(*value*, *decimal=0*)
Print the value to the display.

print_hex(*value*)
Print the value as a hexidecimal string to the display.

scroll(*count=1*)
Scroll the display by specified number of places.

set_digit_raw(*index*, *bitmask*)
Set digit at position to raw bitmask value. Position should be a value of 0 to 3 with 0 being the left most character on the display.

bitmask should be 2 bytes such as: 0xFFFF If can be passed as an integer, list, or tuple

class adafruit_ht16k33.segments.**Seg7x4**(*i2c*, *address=112*, *auto_write=True*)
Numeric 7-segment display. It has the same methods as the alphanumeric display, but only supports displaying a limited set of characters.

colon
Simplified colon accessor

scroll(*count=1*)
Scroll the display by specified number of places.

set_digit_raw(*index*, *bitmask*)
Set digit at position to raw bitmask value. Position should be a value of 0 to 3 with 0 being the left most digit on the display.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

T