tm1640-rpi Documentation

Release 0.1

Michael Farrell

October 20, 2015

Contents

Source code is available in the GitHub repository.

Contents:

CHAPTER 1

Introduction

This is a library for interfacing with the Titan Microelectronics TM1640 16-digit 7-segment display controller.

This was based on the Arduino version written by Ricardo Batista. It was originally ported to Raspberry Pi by FuryFire. Forked off into a seperate project with lots of work done on it by Michael Farrell.

As this is based on code from the original Arduino library, it is licensed under the GPLv3.

1.1 Resources

- Original Arduino version of this library
- Low level TM1640 interface notes
- Datasheet (written in Chinese pages 1 - 10, and English from page 11)
- Assembled unit from DealExtreme.

CHAPTER 2

Building the software

This library requires you have a Raspberry Pi with Linux installed on it.

In order to build the software, you will need to install the following dependencies:

- gcc
- git (if not building from a tarball)
- Python 2.7 (if you wish to use the Python bindings for the library)
- scons
- WiringPi

If you're using Arch Linux ARM, these are all available via `pacman`:

```
# pacman -S scons wiringpi gcc git python2
```

These dependencies will take some time to download and install to your SD card.

Once installed, you can build and install the C library and command-line interface with:

```
# cd /usr/src
# git clone https://github.com/micolous/tm1640-rpi.git
# cd tm1640-rpi
# scons install
```

Once this has completed, you may then install the Python bindings with:

```
# setup.py install
```

CHAPTER 3

Wiring the display

The display has four connections required with the Raspberry Pi:

- VIN (+5v)
- GND
- DIN (data in)
- CLK (data clock)

The DIN and CLK require use of GPIO connections on the Pi. These are all exposed on the P1 connector on the Pi's board.

Connect:

- Pin 2 to VIN
- Pin 6 to GND
- Pin 11 (WiringPi GPIO 0 / BCM GPIO 17) to DIN
- Pin 12 (WiringPi GPIO 1 / BCM GPIO 18) to CLK

You can also wire this to other GPIO pins on the Raspberry Pi. There are a total of 12 GPIO lines exposed. However, the command-line interface will only work when the display is wired using this pin configuration. The libraries do not have this restriction.

When the Pi is turned on, nothing will appear on the display until you wake it up and send some data.

You can test this with the command-line interface:

```
# tm1640 on 7 ; tm1640 clear
# tm1640 write "Hello"
```

You should now see the text Hello on the display.

If it does not appear, you may not have the connections wired properly. Make sure you have used Pin 11 for DIN and Pin 12 for CLK.

CHAPTER 4

Using with shell scripts

The command-line program `tm1640` will let you drive a display from the command-line.

However, it has a number of limitations:

- You cannot write to the screen with offsets, so you have to update the entire screen at once.
- You can only drive one display, using GPIO 0 and 1.
- Because of the overheads involved in creating a new process, update speeds will be slower than using the library.

The purpose of supplying a command-line interface is to facilitate testing and rapid prototyping. It is strongly recommended you write any applications to use the library instead.

Because of restrictions on use of GPIOs for users, you must always run the `tm1640` program as the `root` user.

4.1 Initialise the display

In order to turn on the display, you must set it's brightness to a value between 1 (dimmest) and 7 (brightest).

For example, to turn on the display at maximum brightness:

```
# tm1640 on 7
```

4.2 Clear the display

If you turn on the display, it may have the contents of whatever was in the TM1640 IC's memory last time it was used. You can clear the display with:

```
# tm1640 clear
```

4.3 Turn off the display

In order to turn off the display, and keep the contents of the display in memory, you can issue the off command:

```
# tm1640 off
```

4.4 Write to the display

You can write up to 16 characters to the display at once. If you write less than 16 characters, it will leave the cells that you have not specified in the state that they currently are in.

For example:

```
# tm1640 write "1234567890123456"
```

To print the current date and time to the screen, you can insert the output of the `date` command:

```
# tm1640 write "`date +%Y-%m-%d  %H%M`"
```

Note: Unlike regular PCs, the Raspberry Pi lacks a backup battery for it's clock, so will reset back to 1970-01-01 00:00 UTC on losing power. In order to display accurate time, the Pi's clock will need to be synchronised with an external source, for example, an NTP server.

4.4.1 Vertically-inverted mode

If you are viewing the display in a mirror, you can invert the display vertically.

For example:

```
# tm1640 iwrite "1234567890123456"
```

CHAPTER 5

Using with Python

In order to interact with the display, you must have permissions to control the GPIO pins on the Raspberry Pi. This typically means you need to run as the `root` user.

5.1 Initialise the display

In order to connect to the display, you can use the defaults from the standalone program and use GPIO 0 for data, and GPIO 1 for clock. You can do this with:

```
>>> from tm1640 import *
>>> display = TM1640()
```

If the library is not installed correctly, this will throw an error:

```
Traceback (most recent call last):
  File "tm1640.py", line 37, in __init__
    raise ImportError, 'Could not find libtm1640 in PATH'
ImportError: Could not find libtm1640 in PATH
```

If this occurs, make sure that `libtm1640.so` is in your dynamic loader's path. This typically means it should be located at `/usr/lib/libtm1640.so`, and you need to refresh the dynamic loader's cache by executing **`ldconfig`**. For more information, see *`ld.so(8)`*.

If you wish to use a different set of display pins, you can set this in the `TM1640` constructor as follows:

```
>>> display = TM1640(clock_pin=3, data_pin=2)
```

You must then switch on the display. By default, the Python library will turn on the display at maximum brightness with the `TM1640.on()` method:

```
>>> display.on()
```

You can also set a specific brightness. To turn on the display at brightness level 3 (about half brightness):

```
>>> display.on(3)
```

5.2 Clear the display

If you turn on the display, it may have the contents of whatever was in the TM1640 IC's memory last time it was used. You can clear the display with:

```
>>> display.clear()
```

5.3 Turn off the display

In order to turn off the display, and keep the contents of the display in memory, you can issue the off command:

```
>>> display.off()
```

5.4 Write to the display

You can write up to 16 characters to the display at once. If you write less than 16 characters, it will leave the cells that you have not specified in the state that they currently are in.

For example:

```
>>> display.write('1234567890123456')
```

To print the current date and time to the screen, you could use `datetime.datetime.strftime()`:

```
>>> import datetime
>>> display.write(datetime.datetime.now().strftime('%Y-%m-%d  %H%M'))
```

Note: Unlike regular PCs, the Raspberry Pi lacks a backup battery for it's clock, so will reset back to 1970-01-01 00:00 UTC on losing power. In order to display accurate time, the Pi's clock will need to be synchronised with an external source, for example, an NTP server.

5.4.1 Writing to part of the display at an offset

You can update parts of the screen instead of the entire display at once. This allows you to push updates partial updates to the screen from different parts of your program, or improve the speed of updates.

In the previous example, you could update the time only in the previous example with:

```
>>> display.write(datetime.datetime.now().strftime('%H%M'), offset=12)
```

5.4.2 Vertically-inverted mode

If you are viewing the display in a mirror, you can invert the display vertically, using the *tm1640.INVERT_MODE_VERTICAL* parameter:

```
>>> display.write('1234567890123456', invert_mode=tm1640.INVERT_MODE_VERTICAL)
```

5.5 Class Reference

src/python/tm1640.py - Python interface shim for libtm1640. Copyright 2013 Michael Farrell <http://micolous.id.au/>

This program is free software: you can redistribute it and/or modify it under the terms of the version 3 GNU General Public License as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

class `tm1640.`**`TM1640`**(*clock_pin=1*, *data_pin=0*)

Bases: `object`

Interface class for connecting to a TM1640 display IC over the Raspberry Pi's GPIO pins.

Initialises a connection to the TM1640 display.

Parameters

- **`clock_pin`** (*int*) – WiringPi GPIO that has the clock pin plugged into.
- **`data_pin`** (*int*) – WiringPi GPIO that has the data pin plugged into.

Throws ImportError If `libtm1640.so` could not be found by your dynamic linker. Typically `libtm1640.so` should be installed in to `/usr/lib`.

Throws Exception If there is a problem accessing BCM GPIO, typically caused by lack of permissions.

`clear`()

Clears the contents of the display.

`off`()

Switches off the display, retaining the contents of the display in the controller's memory.

`on`(*brightness=7*)

Turns on the display.

Parameters **`brightness`** – The brightness level to set on the display, between 1 and 7. 1 is dimmest, 7 is brightest.

`write`(*string*, *offset=0*, *invert_mode=0*)

Writes a string to the display.

String must be less than 16 bytes, less any offset bytes.

Throws an exception on error.

Parameters

- **`string`** (*str*) – String to write to the display
- **`offset`** (*int*) – Where to start writing the string to on the display.
- **`invert_mode`** – How to invert the display segments. This must be set to one of *`INVERT_MODE_NONE`* or *`INVERT_MODE_VERTICAL`*.

`tm1640.`**`INVERT_MODE_NONE = 0`**

Used by *`TM1640.write()`* to indicate that the display output should be sent normally.

`tm1640.`**`INVERT_MODE_VERTICAL = 1`**

Used by *`TM1640.write()`* to indicate that the display output segments should be flipped vertically, in order to be displayed correctly through a mirror.

CHAPTER 6

Using with C

In order to use the library in your program, you will need to add the following include line:

```
#include <tm1640.h>
```

When you compile your program, you will need to tell the linker to use `libtm1640.so`. You can typically do this with the `-l` option to **gcc**:

```
$ gcc -o myprogram myprogram.c -ltm1640
```

6.1 Initialise the display

In order to connect to the display, always need to pass the GPIO pins you wish to use. This is normally GPIO 0 for data, and GPIO 1 for clock:

```
tm1640_display* display = tm1640_init(1, 0);
```

You should then verify that there was no error while initialising the display. `tm1640_init()` returns `NULL` on error:

```
if (display == NULL) {
        fprintf(stderr, "Error initialising display!\n");
        return EXIT_FAILURE;
}
```

You must then switch on the display. You must specify a brightness level between 1 and 7. To set the display to maximum brightness:

```
tm1640_displayOn(display, 7);
```

6.2 Clear the display

If you turn on the display, it may have the contents of whatever was in the TM1640 IC's memory last time it was used. You can clear the display with:

```
tm1640_displayClear(display);
```

6.3 Turn off the display

In order to turn off the display, and keep the contents of the display in memory, you can issue the off command:

```
tm1640_displayOff(display);
```

6.4 Write to the display

You can write up to 16 characters to the display at once. If you write less than 16 characters, it will leave the cells that you have not specified in the state that they currently are in.

For example:

```
char* s = "1234567890123456";
int result = tm1640_displayWrite(display, 0, s, strlen(s), INVERT_MODE_NONE);
```

You should verify that the display was written to correctly:

```
if (result != 0) {
        fprintf(stderr, "write error %d\n", result);
        return EXIT_FAILURE;
}
```

6.4.1 Writing to part of the display at an offset

You can update parts of the screen instead of the entire display at once. This allows you to push updates partial updates to the screen from different parts of your program, or improve the speed of updates.

For example, you could write the last 4 characters of the display (offset 12) with:

```
char* s = "test";
int result = tm1640_displayWrite(display, 12, s, strlen(s), INVERT_MODE_NONE);
```

6.4.2 Vertically-inverted mode

If you are viewing the display in a mirror, you can invert the display vertically, using the *INVERT_MODE_VERTICAL* parameter:

```
char* s = "1234567890123456";
int result = tm1640_displayWrite(display, 0, s, strlen(s), INVERT_MODE_VERTICAL);
```

6.5 Library Reference

Defines

DIN_PIN

Default data GPIO pin to use.

Used by the standalone tm1640 application in order to set which display to use.

SCLK_PIN
Default clock GPIO pin to use.

Used by the standalone tm1640 application in order to set which display to use.

INVERT_MODE_NONE
Used by tm1640_displayWrite

Sets an inversion mode of "none", that the segments will be output in the way they were input.

INVERT_MODE_VERTICAL
Used by tm1640_displayWrite

Sets an inversion mode of "vertical", that the segments will be flipped vertically when output on the display.

Functions

tm1640_display ***tm1640_init** (int *clockPin*, int *dataPin*)
Initialises the display.

Return

NULL if wiringPiSetup() fails (permission error)

pointer to *tm1640_display* on successful initialisation.

Parameters

- `clockPin` - WiringPi pin identifier to use for clock (SCLK)
- `dataPin` - WiringPi pin identifier to use for data (DIN)

void **tm1640_destroy** (*tm1640_display* **display*)
Destroys (frees) the structure associated with the connection to the TM1640.

Parameters

- `display` - TM1640 display connection to dispose of.

char **tm1640_invertVertical** (char *input*)
Flips 7-segment characters vertically, for display in a mirror.

Return

Bitmask of segments flipped vertically.

Parameters

- `input` - Bitmask of segments to flip.

int **tm1640_displayWrite** (*tm1640_display* **display*, int *offset*, **const** char **string*, char *length*, int *invertMode*)
displayWrite

Return

-EINVAL if invertMode is invalid

-EINVAL if offset + length > 16

0 on success.

Parameters

- `display` - TM1640 display to write to
- `offset` - offset on the display to start writing from
- `string` - string to write to the display
- `length` - length of the string to write to the display
- `invertMode` - invert mode to apply to text written to the display

char **tm1640_ascii_to_7segment** (char *ascii*)
Converts an ASCII character into 7 segment binary form for display.

Return

0 if there is no translation available.

bitmask of segments that represents the input character.

Parameters

- `ascii` - Input ASCII byte to translate.

void **tm1640_displayClear** (*tm1640_display* **display*)
Clears the display

Parameters

- `display` - TM1640 display to clear

void **tm1640_displayOn** (*tm1640_display* **display*, char *brightness*)
Turns on the display and sets the brightness level

Parameters

- `display` - TM1640 display to set brightness of
- `brightness` - Brightness to set (1 is lowest, 7 is highest)

void **tm1640_displayOff** (*tm1640_display* **display*)
Turns off the display preserving display data.

Parameters

- `display` - TM1640 display to turn off

void **tm1640_send** (*tm1640_display* **display*, char *cmd*, char **data*, int *len*)
Sends a cmd followed by len amount of data. Includes delay from wiringPi.

Bitbanging the output pins too fast creates unpredictable results.

Parameters

- `display` - TM1640 display structure to use for this operation.
- `cmd` - The command
- `data` - Pointer to data that should be appended, or NULL if no data is to be passed.
- `len` - Length of data.

void **tm1640_sendRaw**(*tm1640_display* **display*, char *out*)
Shifts out the byte on the port.

Implementing this with WiringPi directly is too fast for the IC.

Parameters

- `display` - TM1640 display structure to use this for this operation.
- `out` - Byte to send

void **tm1640_sendCmd**(*tm1640_display* **display*, char *cmd*)
Send a single byte command

Parameters

- `display` - TM1640 display structure to use for this operation.
- `cmd` - Command code to send

struct tm1640_display
#include <tm1640.h> Structure that defines a connection to a TM1640 IC.

You should not manipulate this structure directly, and always create new instances of this with tm1640_init

Public Members

int **clockPin**
WiringPi GPIO pin for display clock (SCLK).

int **dataPin**
WiringPi GPIO pin for display data (DIN).

CHAPTER 7

Customising fonts

Fonts are defined in *src/libtm1640/font.h*, which is embedded in the library.

It consists of a 1 dimensional table, defining the LED pattern for that character.

The display uses the `GFEDCBA` configuration of digits. Unsupported charecters are simply set to a space.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

t

C

D

I

O

S

T

W