# 1.8inch LCD Module USER MANUAL

## OVERVIEW

This product is 1.8inch resistive screen module with resolution 128x160. It has internal controller and uses SPI interface for communication. It has already basic functions: setting the point size, the line thickness, drawing circle, rectangle, and displaying English characters.

We provide Raspberry Pi, STM32 and Arduino routines for this product.

## FEATURES

Display type:          TFT

Interface:             SPI

Driver:                ST7735S

Colors:                256K

Resolution:            128 x 160 (Pixel)

Product size:          56.5 x 34(mm)

Display size:          35.4(W) x 28.03(H)(mm)

Pixel size:            0.219(W) x 0.219(H)(MM)

Operating temperature: -30°C ~ 85°C

## INTERFACE DESCRIPTION

| Marking | Description |
|---------|-------------|
| 3V3 | 3.3V power |
| GND | ground |
| DIN | SPI data input |
| CLK | SPI clock |
| CS | chip select |
| DC | data/command |
| RST | reset |
| BL | back light |

## PROGRAM ANALYSIS

1. Working principles:

ST7735S is 132 x 162 pixels LCD panel, but the product is 128 x 160 pixels LCD display. In the display there are two processes: the horizontal direction scanning – from the 2nd pixel, the vertical direction scanning – from the 1st pixel. So, you can see that positions of pixels in RAM correspond to their actual positions while displaying.

The LCD supports 12-bit, 16-bit and 18-bit per pixel input formats. They correspond to RGB444, RGB565 and RGB666 color formats. This routine uses the RGB565 color format, which is commonly used.

LCD uses 4-wired SPI communication interface, which can save a lot of GPIO ports and provides fast data transfer to LCD as well.

2. Communication protocol
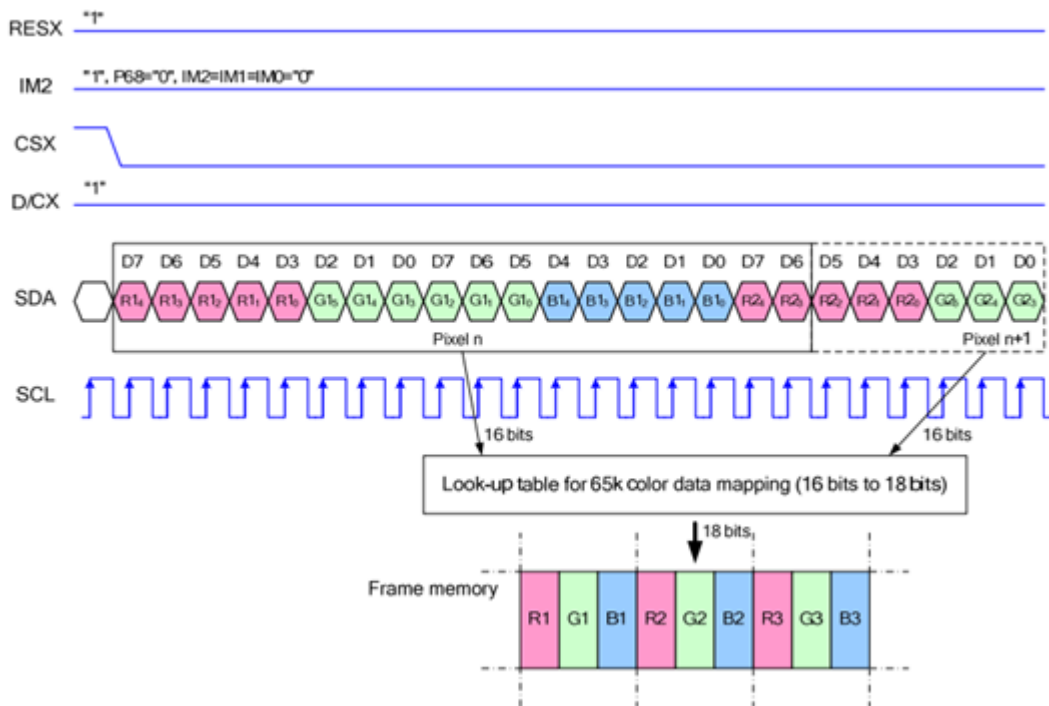


Note: there is a difference from traditional SPI. Here we only need display, so sine wires come from slave to host are hidden. The detailed information please refer to datasheet at page 58

**RESX:** Reset. Pull-down while powering on the module. General set as 1

**IM2:** data communication mode pin, which define usage of SPI

**CSX:** chip selection control pin. If CS=0 – the chip is selected

**D/CX:** data/command control pin, if DC=0 – command is written, otherwise – data are written

**SDA:** transmitted RGB data

**SCL:** SPI clock

The SPI communication protocol of the data transmission uses control bits: clock phase (CPHA) and clock polarity (CPOL):

The value of CPOL determines the level when the serial synchronous clock is in idle state. CPOL=0, that its idle level is 0.

The value of CPHA determines the timing of the data bits relative to the clock pulses. CPHA=0, data is sampled at the first clock pulse edge.

The combination of these two parameters provides 4 modes of SPI data transmission. The commonly used is SPI0 mode, it is that GPOL=0 and CPHA=0.

From the figure above, SCLK begins to transfer data at the first falling edge. 8 bits data are transferred at one clock period. Use SPI0 mode, High bits transfer first, and LOW bits following.

## DEMO CODE

Raspberry Pi, STM32 and Arduino programs are provided, wherein Raspberry Pi provides BCM2835, WiringPi and python programs. It implements common graphical functions as drawing dot, line, rectangle, circle, setting their sizes and line with; filling arias, and displaying English characters of 5 common fonts and other display's functions.

Following instructions are offered for you convenience

### RASPBERRY

1.  Hardware connection

| 1.8inch LCD module | Raspberry Pi |
|---|---|
| 3.3V | 3.3V |
| GND | GND |
| DIN | MOSI (PIN 19) |
| CLK | SCLK (PIN23) |
| CS | CE0 (PIN 24) |
| DC | GPIO.6 (PIN 22) |
| RST | GPIO.2 (PIN13) |
| BL | GPIO.5 (PIN18) |

2.   Enable SPI function of the Raspberry Pi

**sudo raspi-config**

Select: *Advanced Options -> SPI -> yes*

Activate SPI hardware driver.

3.   Installation of libraries

Fore detailed information about libraries installation, please refer to this page:

https://www.waveshare.com/wiki/Libraries_Installation_for_RPi

It is description of WiringPi, bcm2835 and python libraries installation.

4.   Usage

BCM2835 and WiringPi program should be only copied into directory of Raspberry Pi ()by samba or directly copy to the SD card). The following code are compied directly to the user directory of Pi.

4.1  Usage of BCM2835

Run **ls** command as you can see below:



**bin**: contains ".o" files. We don't need to change it generally

Fonts: contains 5 commonly used fonts

**Pic**: contains pictures used for displaying. The resolution of pictures must be 128x128, otherwise they cannot be displayed properly. And the format of pictures must be BMP.

**Obj**: contains object files, like main.c, LCD_Driver.c, DEV_Config.c, LCD_GUI.c and their header files.

*main.c:* The mian function. What need to note is that even though there are LCD_ScanDir used to control the direction of scanning, you need not to change it. Because this module is designed for Raspberry Pi, and for compatibility, we don't recommend you to change it.

*DEV_Config.c:* Definations of Raspberry Pi's pins and the communication mode.

*LCD_Driver.c:* Drive code of LCD. Need not change generally.

*LCD_BMP.c:* Reading and analyzing BMP files and display them

*Makefile:* This file contains compilation rules. If there are some changes in code, please run **make clean** to clean all the dependency file and executable files. Then execute **make** to compile the whole project and generate new executable files.

tftlcd_1in8: executable file, generated by command **make**

To run the program, you just need to run this command on terminal: **sudo ./tftlcd_1in8**

## 4.2  WiringPi

Input **ls** command, now you can see following:

```
pi@raspberrypi:~/wiringpi $ ls
bin  Fonts  Makefile  obj  pic  tftlcd_1in8
```

The folders is similar to BCM2835's. The only differences are that:

1.  WiringPi oprates by read/write the device files of Linux OS. and the bcm2835 is library function of Raspberry Pi's CPU, it operates registers directly. Thus, if you have used bcm2835 libraries firstly, the usage of WiringPi code will be failed. In this case, you just need to reboot the system and try again.

2.  Due to the first difference, they underlying configuration are different. In DEV_Config.c, use wiringpiPi and the corresponding wiringPiSPI to provide underlay interfaces.

The program executed by command **sudo ./tftlcd_1in8** as well

## 4.3  Python

Input **ls** command, you can see that:

```
pi@raspberrypi:~/python $ ls
LCD_1in8.py  LCD_Config.py  main.py  time.bmp
```

LCD_1in8.py: Driver code of LCD

LCD_Config.py: configuration of hardware underlaying interface.

Executing program: **sudo python LCD_1in8.py**

Note: Some of the OS don't have image libraries. In this case, you can run: **sudo apt-get install python-imaging** to install the image library.

Image is an image processing library of python, represents any image by an image object. Thus, we can create a blank image by **new**, its size must be same as the display size of LCD. Then draw picture by Draw library, finally, transfer the image to the LCD. Here using **Image.load()** too read RGB888 data of pixel, and convert to RGB565. Scanning every pixel then we could get the whole image for displaying. Its most important code is as below:

```python
def LCD_ShowImage(self,Image,Xstart,Ystart):
    if (Image == None):
        return

    self.LCD_SetWindows ( 0, 0, self.LCD_Dis_Column , self.LCD_Dis_Page )
    Pixels = Image.load()
    for j in range(0, self.LCD_Dis_Page ):
        for i in range(0, self.LCD_Dis_Column ):
            Pixels_Color = ((Pixels[i, j][0] >> 3) << 11)|((Pixels[i, j][1] >> 2) << 5)|(Pixels[i, j][2] >> 3)#RGB Data
            self.LCD_SetColor(Pixels_Color , 1, 1)
```

## 5.  Auto-run

Initialize autorun in Raspberry Pi by configuring code of /etc/rc.local file:

**sudo vim /etc/rc.local**

Before exit0 add:

**sudo python /home/pi/python/demo.py &**

Important: to place the program /home/pi/python/demo.py at the same director, you can input command **pwd** to get the path. And & character is necessary at the end of command line, otherwise probable need to reinstall the system (impossible terminate the process by pressing ctrl+c, impossible to login with pi user permission).

## STM32

This demo uses XNUCLEO-F103RB developing board and is based on HAT library.

1.  Hardware connection

| 1.8inch LCD | XNUCLEO-F103RB |
|:-----------:|:--------------:|
| VCC | 3V3 |
| GND | GND |
| DIN | PA7 |
| CLK | PA5 |
| CS | PB6 |
| DC | PA8 |
| RST | PA9 |
| BL | PC7 |

2.  Expected result

Program the demo into the development Board. Firstly the screen is refreshed completely, then a solid line, dashed line, open circle, solid circle, rectangle, solid torque are drawn and English characters are shown.

## ARDUINO

UNO PLUS Arduino development board is used here.

1.  Hardware connection

| 1.8inch LCD | Arduino |
|:-----------:|:-------:|
| 3.3V | 3V3 |

| | |
|---|---|
| GND | GND |
| DIN | D11 |
| CLK | D13 |
| CS | D10 |
| DC | D7 |
| RST | D8 |
| BL | D9 |

2. Due to small global memory 2Kb of UNO PLUS, the display can't work in graphical mode, but the calling method is the same. Just because there is no enough memory, this demo is not provided.

## COMPATIBLE CODE PORTING

Offered demo is the commonly used programs, which are able to be ported. They can be used with two screens and the difference is only in initialization of them and their sizes.

The usage method is defined by macros. In LCD_Driver.h or in LCD.h:

#define LCD_1IN44

#define LCD_1in8.

As the name of the macros, they are used for 1.44inch and 1.8inch LCD separately. To use for one LCD, just need to comment other one.

For example:

//#define LCD_1IN44

#define LCD_1IN8

Here we use it for 1.8inch LCD, so we comment the 1.44 macro. After saving, Run **make clean** to remove dependency files, and then run **make** to generate new executable files.

# X-ON Electronics

Largest Supplier of Electrical and Electronic Components

*Click to view similar products for* LCD Graphic Display Modules & Accessories *category:*

*Click to view products by* Waveshare *manufacturer:*

Other Similar products are found below :

HDM64GS12L-Y11S  MGLS-240128-Z05  15932  RED202A  16249  18079  17344  DEM 128032A1 FGH  DEM 128032A1 FGH-PW  DEM 128064K1 SBH-PW-N  20755  19192  19576  19579  19650  14972  15867  18628  19173  19653  19804  19888  19907  18231  18366  HDM64GS24L-2-Y14S  RG12864A-FHG-V  RG12864A-TIG-V  RG12864A-YHY-X  RG12864B-BIW-V  RG12864B-FHW-V  RG12864B-GHW-V  RG12864K-BIW-VBG  RG320240A1-BIW-V  13892  DEM 128064B SBH-PW-N  DEM 128064G FGH-PW  DEM 128064I FGH-PW  DEM 128064O FGH-PW  DEM 128064O SBH-PW-N  DEM 128064P SBH-PW-N  DEM 128064Q SBH-PW-N  DEM 128128D FGH-PW  DEM 240064B FGH-PW  DEM 240064B SBH-PW-N  DEM 320240B FGH-PW-N  EA W240-7KHLW  16239  RX12864D3-BIW  RX240128A-TIW