0.96inch LCD Module

用户手册

产品概述

本产品是 0.96 寸 IPS 显示屏模块，分辨率为 160*80，使用 SPI 接口通信，LCD 带有内部控制器，已封装好基本函数，可以实现图片旋转、画点、线、圆、矩形，显示英文字符和中文字符，显示图片。

提供完善的配套树莓派例程（BCM2835 库，WiringPi 库，以及 python 例程），STM32 例程，Arduino 例程。

产品参数

工作电压：	3.3V
通信接口：	SPI
屏幕类型：	TFT
控制芯片：	ST7735S
分辨率：	160 (V)RGB x 80(H)
显示尺寸：	21.7（V）x 10.8（H）mm
像素大小：	0.1356（V）x 0.135（H）mm
产品尺寸	32.5 x26.00(mm)

版本：V1.0.0，日期：2018 年 12 月 05 日

接口说明

功能引脚	描述
VCC	3.3V/5V 电源正
GND	电源地
DIN	SPI 数据输入
CLK	SPI 时钟输入
CS	片选，低电平有效
DC	数据/命令选择
RST	复位
BL	背光

目录

产品概述 1
产品参数 1
接口说明 2
硬件说明 4
LCD 及其控制器 4
通信协议 4
示例程序 6
下载例程 6
树莓派例程 6
复制例程到树莓派 6
函数库安装 7
硬件连接 9
运行示例程序 9
预期效果 10
STM32 程序 11
硬件连接 11
预期效果 11
Arduino 12
硬件连接 12
预期效果 12
常见问题 13

硬件说明

LCD 及其控制器

本款 LCD 使用的内置控制器为 ST7735S，是一款 162 x RGB x 132 像素的 LCD 控制器，而本 LCD 本身的像素为 160(H)RGB x 80(V)。

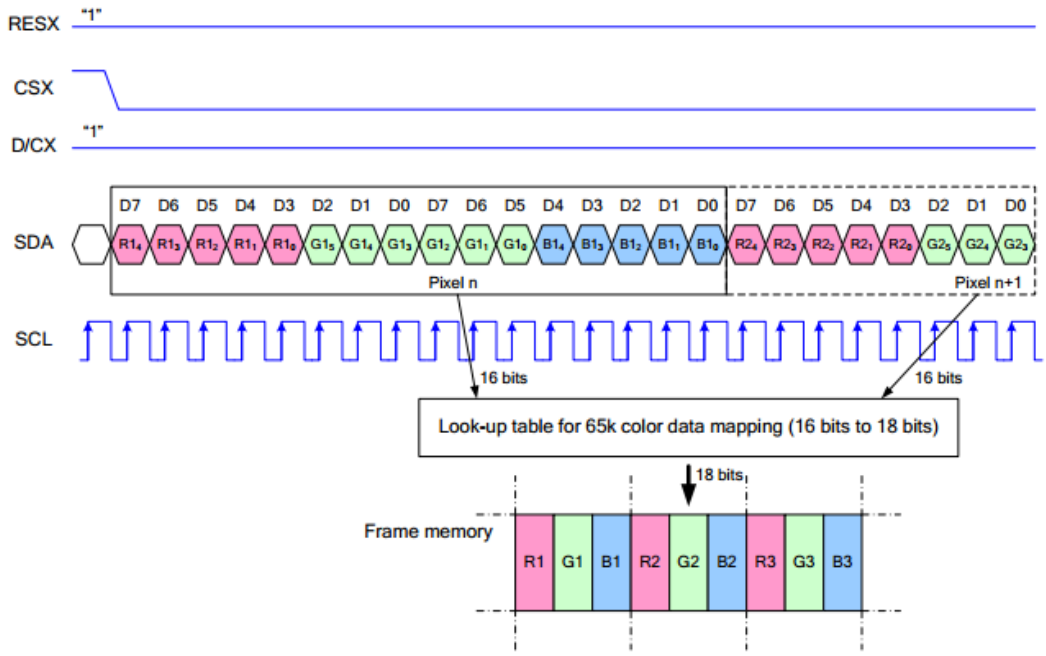
查看数据手册可以得知该控制器支持 12 位，16 位以及 18 位每像素的输入颜色格式，即 RGB444,RGB565,RGB666 三种颜色格式，本屏幕使用 RGB565 格式的颜色格式，这也是常用的 RGB 格式。

由于 LCD 的显示起始位置与控制器的原点不一致，因此在使用控制器初始化显示全屏显示区域时需要对做偏移处理：水平方向从第二个像素点开始显示，垂直方向从第 27 个像素点开始。这样就可以保证显示的 LCD 中 RAM 对应的位置与实际一致。

对于大部分的 LCD 控制器而言，都可以配置控制器的通信方式，通常都有 8080 并行接口、三线 SPI、四线 SPI 等通信方式。此 LCD 使用四线 SPI 通信接口，这样可以大大的节省 GPIO 口，同时通信速度也会比较快。

通信协议

从上的得知使用的是 4 线 SPI 通信，查阅数据手册可以得到如下的通信时序图，以传输 RGB556 为例：



注：与传统的 SPI 协议不同的地方是：由于是只需要显示，故而将从机发往主机的数据线进行了隐藏，该表格详见 Datasheet Page 66。

RESX 为复位，模块上电时拉低，通常情况下置 1；

CSX 为从机片选， 仅当 CS 为低电平时，芯片才会被使能。

D/CX 为芯片的数据/命令控制引脚，当 DC = 0 时写命令，当 DC = 1 时写数据

SDA 为传输的数据，即 RGB 数据；

SCL 为 SPI 通信时钟。

对于 SPI 通信而言，数据是有传输时序的，即时钟相位（CPHA）与时钟极性(CPOL)的组合：

CPHA 的高低决定串行同步时钟是在第一时钟跳变沿还是第二个时钟跳变沿数据被采集，当 CPHL = 0，在第一个跳变沿进行数据采集；

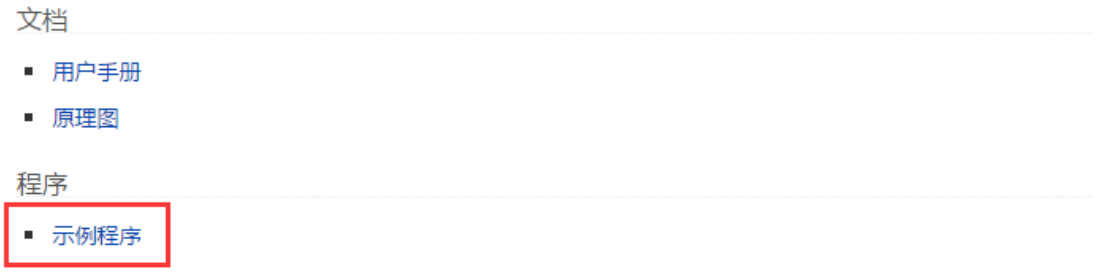
CPOL 的高低决定串行同步时钟的空闲状态电平，CPOL = 0，为低电平。

从图中可以看出，当 SCLK 第一个下降沿时开始传输数据，一个时钟周期传输 8bit 数据，使用 SPI0，按位传输,高位在前,低位在后。

示例程序

下载例程

在微雪官网找到对应的产品，找到产品资料下载路径并打开，下载示例程序：



解压得到如下文件：

其中：

Arduino:为 Arduino 例程，以 UNO 开发板为例；

RaspberryPi:为树莓派例程，包含 BCM2835、WiringPi、python 三种例程；

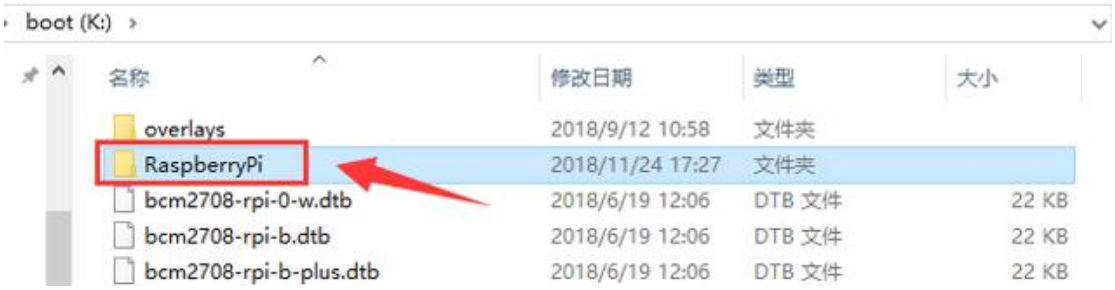
STM32:为 STM32 例程，以 XNUCLEO-F103RB 开发板为例，是基于 STM32F103RBT6 的；

树莓派例程

复制例程到树莓派

使用读卡器将 SD 卡插入电脑，将会显示一个 40M 左右的 U 盘，盘名叫：boot.

将 RaspberryPi 文件夹复制到 boot 根目录下。



然后弹出 U 盘，将 SD 卡插入到树莓派中，启动树莓派，查看/boot 目录，可以看到刚刚复制

进去的文件：ls /boot

```
pi@raspberrypi:~ $ ls /boot/
bcm2708-rpi-0-w.dtb     bcm2710-rpi-3-b.dtb       config.txt     fixup_x.dat   kernel.img        start_cd.elf
bcm2708-rpi-b.dtb       bcm2710-rpi-3-b-plus.dtb  COPYING.linux  FSCK0000.REC  LICENCE.broadcom  start_db.elf
bcm2708-rpi-b-plus.dtb  bcm2710-rpi-cm3.dtb       fixup_cd.dat   FSCK0001.REC  LICENSE.oracle    start.elf
bcm2708-rpi-cm.dtb      bootcode.bin              fixup.dat      issue.txt     overlays          start_x.elf
bcm2709-rpi-2-b.dtb     cmdline.txt               fixup_db.dat   kernel7.img   RaspberryPi       System Volume Information
```

将文件加复制到用户目录下，并修改权限：

sudo cp -r /boot/RaspberryPi/ ./

sudo chmod 777 -R RaspberryPi/

```
pi@raspberrypi:~ $ sudo cp -r /boot/RaspberryPi/ ./
pi@raspberrypi:~ $ ls
code  libcode  RaspberryPi  RPIlib  ubuntu  usbdisk
pi@raspberrypi:~ $ sudo chmod 777 -R RaspberryPi/
pi@raspberrypi:~ $ ls
code  libcode  RaspberryPi  RPIlib  ubuntu  usbdisk
```

函数库安装

在使用例程前，需要先安装一下函数库，否则无法正常运行

安装 BCM2835 库:

进入 BCM2835 官网下载最新的函数库安装包，并复制到树莓派上，运行指令进行安装

```
cd

sudo tar zxvf bcm2835-1.xx.tar.gz

cd bcm2835-1.xx

sudo ./configure
```

```
make

sudo make check

sudo make install

cd
```

其中 xx 代表的是下载的版本号，例如，下载的如果是 bcm2835-1.52，那么应该执行：sudo

tar zxvf bcm2835-1.52.tar.gz

安装 wiringPi 库:

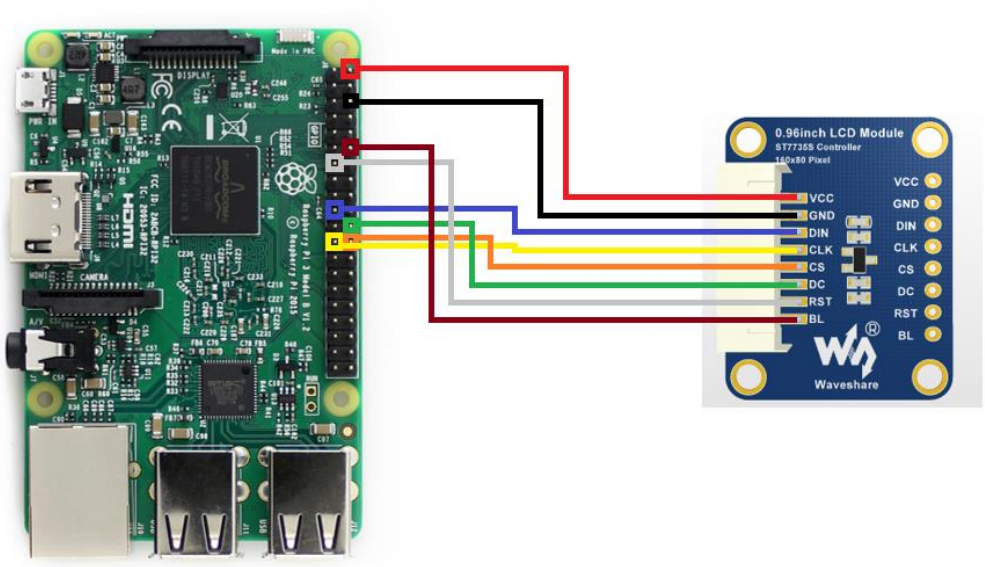
运行指令安装

```
cd

sudo apt-get install git

sudo git clone git://git.drogon.net/wiringPi

cd wiringPi

sudo ./build

cd
```

安装 python 库:

```
cd

sudo apt-get install python-pip

sudo pip install RPi.GPIO

sudo pip install spidev

sudo apt-get install python-imaging

cd
```

硬件连接



具体连接入下表所示：

0.96inch LCD	Raspberry Pi (Board)	Raspberry Pi (BCM)
VCC	5V	5V
GND	GND	GND
DIN	19	MOSI
CLK	23	SCLK
CS	24	CE0
DC	22	P25
RST	13	P27
BL	12	P18

运行示例程序

进入文件夹：cd RaspberryPi/

```
pi@raspberrypi:~ $ cd RaspberryPi/
pi@raspberrypi:~/RaspberryPi $ ls
bcm2835  python  wiringpi
```

运行 bcm2835 程序：

```
cd bcm2835

sudo ./motor
```

Ctrl+C 可以终止程序运行

运行 wiringpi 程序：

```
cd wiringpi

sudo ./motor
```

Ctrl+C 可以终止程序运行

运行 python 程序：

```
cd python

sudo python main.py
```

Ctrl+C 可以终止程序运行

预期效果

1. 屏幕全部刷白
2. 显示数字、英文和中文
3. 画矩形框
4. 画直线（对角线）
5. 显示一张 40X40 的 logo
6. 显示一张 160X80 全屏图片

STM32 程序

本例程使用的开发板为 XNUCLEO-F103RB，程序是基于 HAL 库

硬件连接

0.96inch LCD	XNUCLEO-F103RB
VCC	5V
GND	GND
DIN	PA7
CLK	PA5
CS	PB6
DC	PA8
RST	PA9
BL	PB0

预期效果

1. 屏幕全部刷白
2. 显示数字和英文
3. 显示中文
4. 画矩形框
5. 画直线（对角线）
6. 画 1 个空心圆
7. 显示一张 40X40 的 logo
8. 显示一张 160X80 全屏图片

ARDUINO

本例程使用的开发板为 Arduino UNO

硬件连接

0.96inch LCD	UNO
VCC	5V
GND	GND
CLK	D13
DIN	D11
CS	D10
DC	D7
RST	D8
BL	D9

预期效果

1. 屏幕全部刷白
2. 显示数字和英文
3. 显示中文
4. 画矩形框
5. 画直线（对角线）
6. 画 1 个空心圆
7. 显示一张 40X40 的 logo

常见问题

1. 背光的控制?

对于背光的控制，在 C 语言中全部封装成一个函数 LCD_SetBacklight();但是在不同的开发环境，里面的值代表的不同的亮度，需要用户自己去找到函数定义，上面有说明。

2. 使用树莓派控制，LCD 黑屏?

a) 检查打开了 SPI;

b) 检查背光引脚是否有输出，无输出可尝试悬空 BL 控制线;

3. 不正确的使用树莓派控制可能会导致?

如果运行 wiringPi 例程正常，再运行 python 或者 BCM2835 可能会屏幕无法正常刷新，因为 bcm2835 库是树莓派 cpu 芯片的库函数，底层是直接操作寄存器，而 wiringPi 库和 python 的底层都是通过读写 linux 系统的设备文件操作设备，可能导致 GPIO 口异常，重启树莓派可完美解决。

4. 如何翻转图像?

C 语言控制可以使用函数 Paint_SetRotate(Rotate);当时在 C 语言中翻转的角度只能为 0、90、180、270 度;

Python 可以调用 rotate(Rotate)来翻转任意角度。

5. Python Image 库

对于有些树莓派系统可能并没有 image 这个库，运行：sudo apt-get install python-imaging 安装 python-imaging 库