1.54inch OLED SSD1309 SPI&IIC Module
MSP154W&MSP154B
用户手册

OLED 简介

OLED 即有机发光二级管（Organic Light-Emitting Diode, OLED)。OLED 显示技术具有自发光、广视角、几乎无穷高的对比度、较低耗电、极高反应速度、可用于挠曲性面板、使用温度范围广、构造及制程较简单等优点，被认为是下一代的平面显示器新兴应用技术。

OLED 显示和传统的 LCD 显示不同,其可以自发光,所以不需要背光灯,这使得 OLED 显示屏相对于 LCD 显示屏尺寸更薄，同时显示效果更优。

产品概述

该款 OLED 模块显示尺寸为 1.54 寸，拥有 128x64 分辨率。可以选择 3 线制、4 线制 SPI 以及 IIC 三种通信方式，驱动 IC 为 SSD1309。包含显示黑色、蓝色或者黄蓝双色三款模块。

产品特点

- 1.54 寸 OLED 屏，支持黑白、黑蓝显示
- 128x64 分辨率，显示效果清晰，对比度高
- 超大可视角度:大于 160°(显示屏中可视角度最大的一种屏幕)
- 宽电压供电（3V~5V），兼容 3.3V 和 5V 逻辑电平，无需电平转换芯片
- 默认 4 线制 SPI 总线，可以选择 IIC 总线
- 超低功耗：正常显示仅为 0.06W（远低于 TFT 显示屏）
- 军工级工艺标准,长期稳定工作
- 提供丰富的 STM32、C51、Arduino、Raspberry Pi 以及 MSP430 平台示例程序
- 提供底层驱动技术支持

产品参数

名称	描述
显示颜色	白色、蓝色
SKU	MSP154W MSP154B
尺寸	1.54(inch)
类型	OLED
驱动芯片	SSD1309
分辨率	128*64(Pixel)
模块接口	4-line SPI、IIC interface
有效显示区域	35.052x17.516(mm)
触摸屏类型	无触摸屏
触摸 IC	无触摸 IC
模块尺寸	42.40x38.00(mm)
视角	>160°
工作温度	-20℃~60℃
存储温度	-30℃~70℃
工作电压	3.3V/5V
功耗	待定
产品重量（含包装）	12(g)

接口说明

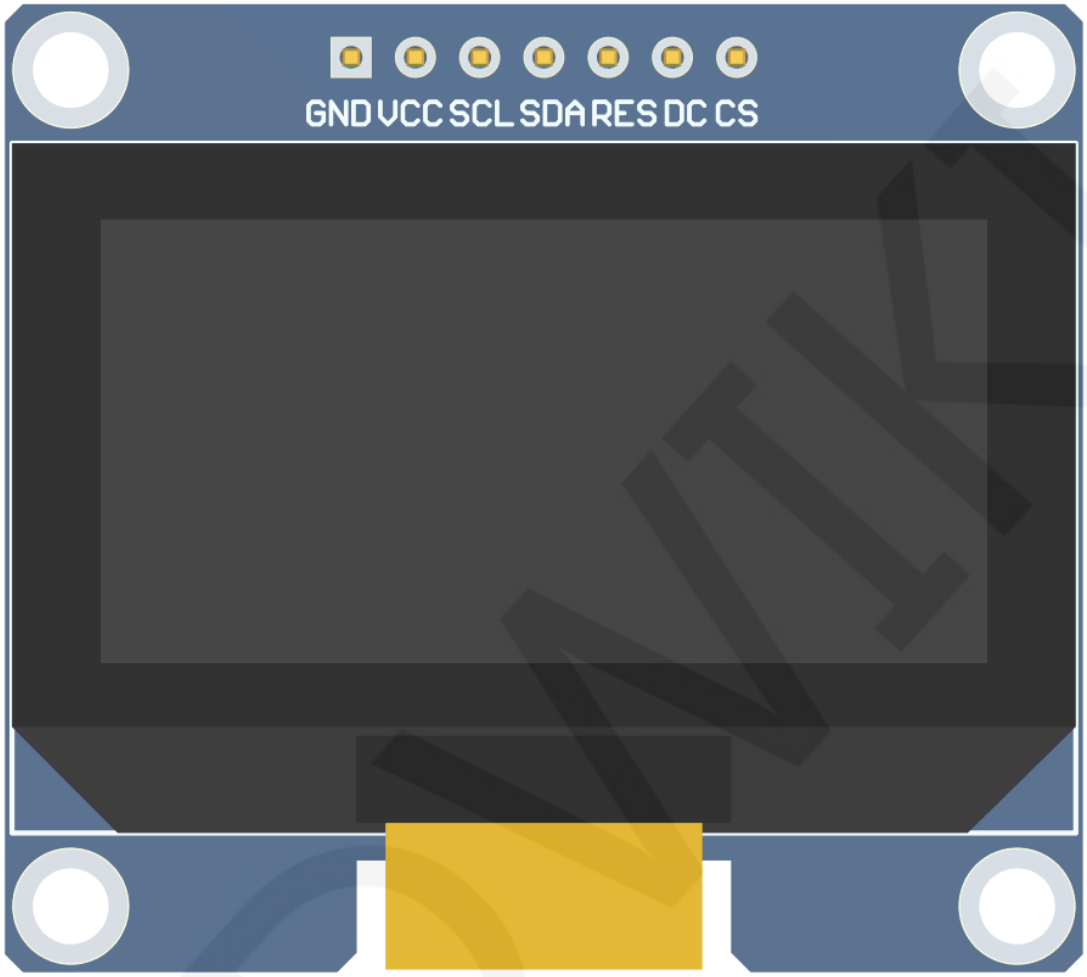


图1. 模块引脚丝印图

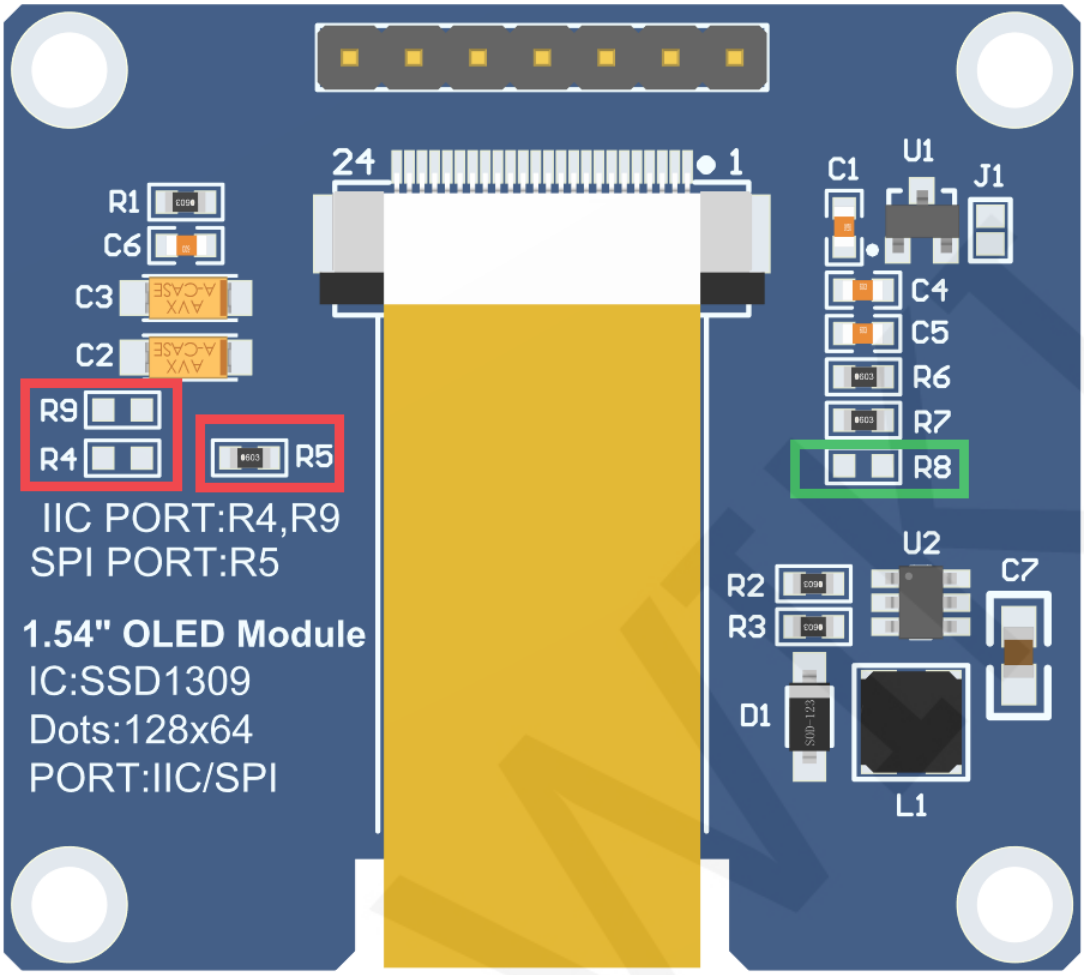


图2. 模块背面图

注意：

1、 本模块支持IIC和4线制SPI接口总线模式切换（如图2红框内所示），具体说明如下：

A、 使用4.7K电阻只焊接R5电阻，则选择4线制SPI总线接口（默认）；

B、使用4.7K电阻只焊接R4、R9电阻，则选择IIC总线接口；

2、接口总线模式切换后，需要选择相应配套的软件和相应的接线引脚（如图1所示），模块才能正常运行。相应的接线引脚说明如下：

A、选择4线制SPI总线接口，所有的引脚都需要使用；

B、选择IIC总线接口，只需要使用GND、VCC、SCL、SDA、RES这五个引脚，将CS引脚接电源地，DC引脚可以用来选择IIC从设备地址，接高电平选择0x7A，接低电平选择0x78；

3、 焊接R8电阻（如图2绿框内所示），则CS引脚不需要接。

重要说明:

1. 以下引脚序号1~7是指我司带PCB底板的模块引脚编号，如果您购买的是裸屏，请参考裸屏规格书的引脚定义，按照信号类型来参考接线而不是直接根据下面的模块引脚编号来接线，举例：CS在我们模块上是7脚，可能在不同尺寸裸屏上是x脚，以下接线说明是告诉您，CS这个信号是接到的单片机的A5引脚上的。
2. 关于VCC供电电压：该OLED显示模块可以接3.3V或者5V。

标号	模块引脚	引脚说明
1	GND	OLED电源地
2	VCC	OLED电源正(3.3V~5V)
3	SCL	OLED SPI和IIC总线时钟信号
4	SDA	OLED SPI和IIC总线数据信号
5	RES	OLED复位信号，低电平复位（选择IIC总线时，该引脚需要接高电平（可以接VCC））
6	DC	选择SPI总线，作为命令/数据输入选择信号，高电平：数据，低电平：命令； 选择IIC总线时，该引脚可以用来选择IIC从设备地址，接高电平选择0x7A，接低电平选择0x78；
7	CS	OLED片选信号，低电平使能； 焊接R8电阻（如图2绿框内所示），则该引脚不需要接。 不焊接R8电阻，选择IIC接口，则该引脚接地

硬件配置

该模块硬件电路由 4 部分组成：OLED 显示控制电路、OLED 升压电路、排针接口、电源稳压电路。

OLED 显示控制电路主要用于控制 OLED 显示，包括片选、复位以及数据、命令传输控制，接口选择。

OLED 升压电路用于将输入电压升压为 OLED 发光电压。

排针接口用于外接主控开发板。

电源稳压电路用于 3.3V 稳压供电。

该 OLED 模块默认采用 4 线制 SPI 通信方式，另外还可以选择 IIC 通信方式，硬件配置 7 个引脚，不同的通信方式，选择的引脚不一样（具体见接口说明部分）。

工作原理

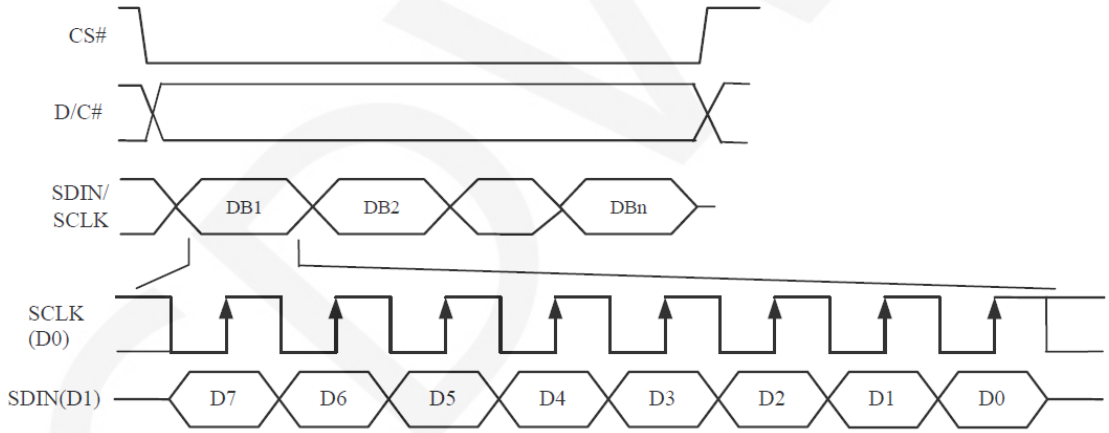
1、SSD1309 控制器简介

SSD1309 为一款 OLED/PLED 控制器,支持的最大分辨率为 128*64,拥有一个 1024 字节大小的 GRAM。支持 8 位 6800 和 8 位 8080 并口数据总线，还支持 3 线制和 4 线制 SPI 串口总线以及 I2C 总线。由于并行控制需要大量的 IO 口，所以最常用的还是 SPI 串口总线和 I2C 总线。其支持垂直滚动显示，可用于小型便携式设备，如手机、MP3 播放器等。

SSD1309 控制器使用 1bit 来控制一个像素点显示,所以每个像素点只能显示黑白双色。其显示的 RAM 总共分为 8 页，每页有 8 行，每行 128 个像素点。设置像素点数据时，需要先指定页地址，再分别指定列低地址和列高地址，所以每次同时设置垂直方向的 8 个像素点。为了能够灵活控制任意位置的像素点,软件上先设置一个和显示 RAM 一样大小的全局一维数组，先将像素点数据设置到全局数组中，此过程采用或、与操作保证之前写入全局数组的数据不受破坏,然后将全局数组的数据写入到显示 RAM 中,这样就可以通过 OLED 显示出来了。

2、SPI 通信协议简介

4 线制 SPI 总线写模式时序如下图所示：



4 线制的 D/C#信号直接由 D/C#输入。

CS#为从机片选信号， 仅当 CS#为低电平时，芯片才会被使能。

D/C#为芯片的数据/命令控制信号，当 DC#为低电平时写命令，为高电平时写数据。

SCLK 为 SPI 总线时钟信号，每个上升沿传输 1bit 数据；

SDIN 为 SPI 总线写数据信号，按照高位在前，先传输的方式，一次传输 8bit 数据，数据对于 SPI 通信而言，数据有传输时序，即时钟相位（CPHA）与时钟极性(CPOL)的组合：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。CPOL 对传输协

议没有很多的影响；

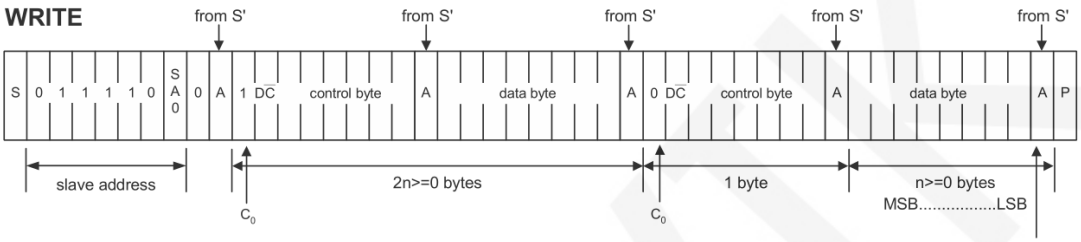
CPHA 的高低决定串行同步时钟是在第一时钟跳变沿还是第二个时钟跳变沿数据被采集，

当 CPHL = 0，在第一个跳变沿进行数据采集；

这两者组合就成为四种 SPI 通信方式，国内通常使用 SPI0,即 CPHL = 0，CPOL = 0

3、IIC 通信协议简介

IIC 总线写数据过程如下图所示：



IIC 总线开始工作后，首先会发送从设备地址，待收到从设备应答后，然后发送一个控制字节，用于通知从设备，接下来要发送的数据是写入 IC 寄存器的命令还是写入 RAM 的数据，待收到从设备应答后，然后发送多个字节的值，直到发送完成，IIC 总线停止工作。

其中：

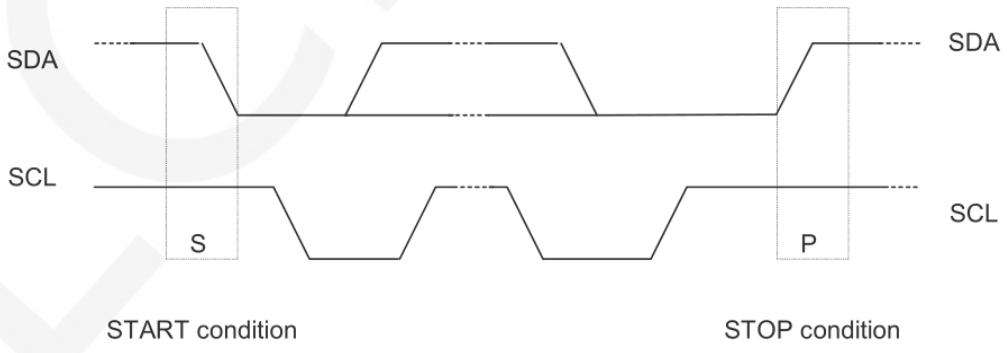
C_0=0：此为最后一个控制字节，接下里发送的都是数据字节

C_0=1：接下来两份要发送的两个字节分别为数据字节和另外一个控制字节

$D/\overline{C}$=0：为寄存器命令操作字节

$D/\overline{C}$=1：为 RAM 数据操作字节

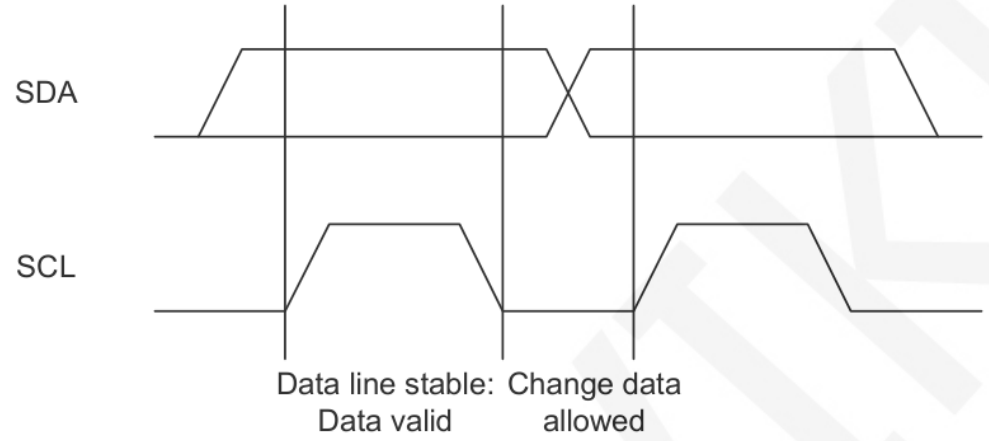
IIC 开始和停止时序图如下：



当 IIC 的数据线和时钟线都保持高电平时，IIC 为空闲状态，此时数据线由高电平变为

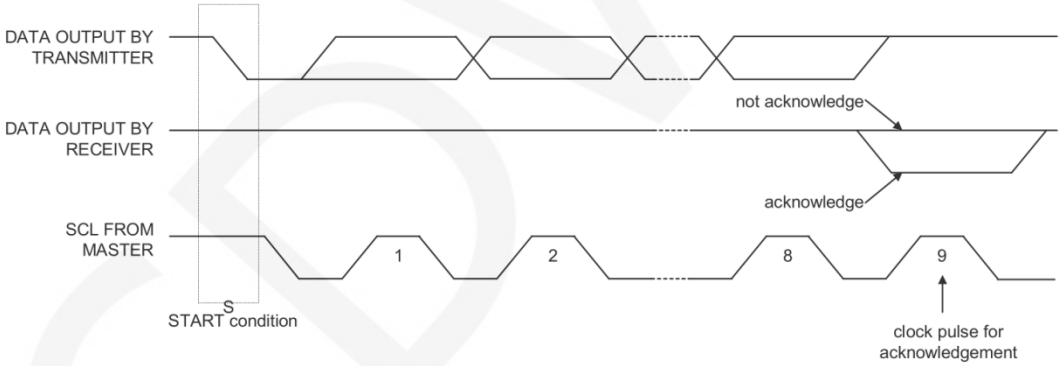
低电平，时钟线继续保持高电平，IIC 总线就启动数据传输。当时钟线保持高电平时，数据线由低电平变为高电平，IIC 总线停止数据传输。

IIC 发送一个 bit 数据的时序图如下：



每一个时钟脉冲（拉高拉低的过程），发送 1bit 数据。当时钟线为高电平时，数据线必须保持稳定，当时钟线为低电平时，才允许数据线改变。

ACK 发送时序图如下：



主设备等待从设备的 ACK 时，需要保持时钟线为高电平，从设备发送 ACK 时，要将数据线保持为低电平。

使用说明

1、Arduino 使用说明

接线说明：

引脚标注见接口说明。

Arduino UNO单片机测试程序接线说明			
序号	模块引脚	对应UNO开发板接线引脚	
		SPI	IIC
1	GND	GND	
2	VCC	5V/3.3V	
3	SCL	13	A5
4	SDA	11	A4
5	RES	3.3V	
6	DC	9	VCC或者GND
7	CS	10	GND

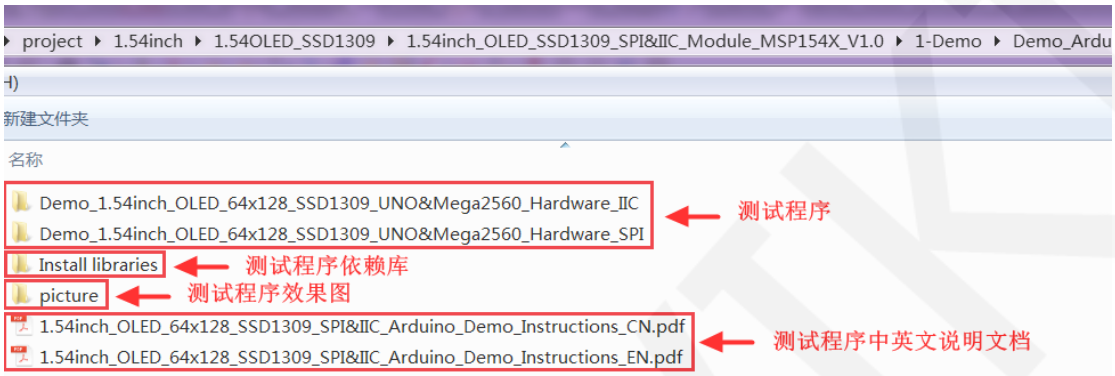
Arduino MEGA2560单片机测试程序接线说明			
序号	模块引脚	对应MEGA2560开发板接线引脚	
		SPI	IIC
1	GND	GND	
2	VCC	5V/3.3V	
3	SCL	53	21
4	SDA	51	20
5	RES	3.3V	
6	DC	9	VCC或者GND
7	CS	10	GND

操作步骤：

A、按照上述接线说明将 OLED 模块和 Arduino 单片机连接起来，并上电；

B、选择需要测试的示例，如下图所示：

（测试程序说明请查阅测试程序说明文档）



C、打开所选的示例工程，进行编译和下载。

关于 Arduino 测试程序编译和下载的具体操作方法见如下文档：

http://www.lcdwiki.com/res/PublicFile/Arduino_IDE_Use_Illustration_CN.pdf

D、OLED 模块如果正常显示字符和图形，则说明程序运行成功；

2、RaspberryPi 使用说明

接线说明：

引脚标注见接口说明

注意：

物理引脚是指 RaspBerry Pi 开发板的 GPIO 引脚编码

BCM 编码是指使用 BCM2835 GPIO 库时的 GPIO 引脚编码

wiringPi 编码是指使用 wiringPi GPIO 库时的 GPIO 引脚编码

在代码里面使用哪个 GPIO 库，引脚定义就需要使用相应的 GPIO 库编码，详情见 GPIO map 表

wiringPi 编码	BCM 编码	功能名	物理引脚 BOARD编码		功能名	BCM 编码	wiringPi 编码
		3.3V	1	2	5V		
8	2	SDA.1	3	4	5V		
9	3	SCL.1	5	6	GND		
7	4	GPIO.7	7	8	TXD	14	15
		GND	9	10	RXD	15	16
0	17	GPIO.0	11	12	GPIO.1	18	1
2	27	GPIO.2	13	14	GND		
3	22	GPIO.3	15	16	GPIO.4	23	4
		3.3V	17	18	GPIO.5	24	5
12	10	MOSI	19	20	GND		
13	9	MISO	21	22	GPIO.6	25	6
14	11	SCLK	23	24	CE0	8	10
		GND	25	26	CE1	7	11
30	0	SDA.0	27	28	SCL.0	1	31
21	5	GPIO.21	29	30	GND		
22	6	GPIO.22	31	32	GPIO.26	12	26
23	13	GPIO.23	33	34	GND		
24	19	GPIO.24	35	36	GPIO.27	16	27
25	26	GPIO.25	37	38	GPIO.28	20	28
		GND	39	40	GPIO.29	21	29

图 3.GPIO Map

Raspberry Pi测试程序接线说明			
序号	引脚丝印	对应开发板接线引脚	
		SPI	IIC
1	GND	GND (物理引脚：6,9,14,20,25,30,34,39)	
2	VCC	5V/3.3V (物理引脚：1,2,4)	
3	SCL	物理引脚：23 BCM编码：11 wiringPi编码：14	物理引脚：5 BCM编码：3 wiringPi编码：9

4	SDA	物理引脚：19 BCM编码：10 wiringPi编码：12	物理引脚：3 BCM编码：2 wiringPi编码：8
5	RES	物理引脚：5 BCM编码：3 wiringPi编码：9	物理引脚：23 BCM编码：11 wiringPi编码：14
6	DC	物理引脚：3 BCM编码：2 wiringPi编码：8	VCC或者GND
7	CS	物理引脚：24 BCM编码：8 wiringPi编码：10	GND

操作步骤：

A、开启 RaspberryPi 的 SPI 功能

使用串口终端工具（如 putty）登录 RaspberryPi，输入如下命令：

```
sudo raspi-config
```

选择 Interfacing Options->SPI->YES

启动 RaspberryPi 的 SPI 内核驱动

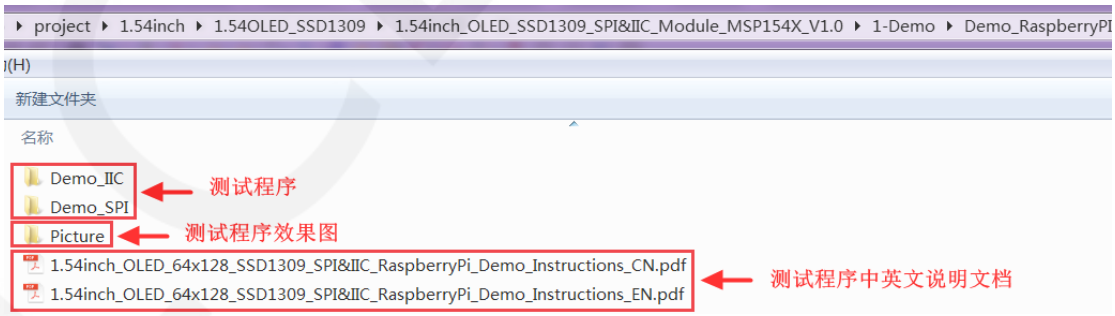
B、安装函数库

关于 RaspberryPi 的 bcm2835、wiringPi、python 函数库的详细安装方法见如下文档：

http://www.lcdwiki.com/res/PublicFile/Raspberrypi_Use_Illustration_CN.pdf

C、选择需要测试的示例，如下图所示

（测试程序说明请查阅测试程序说明文档）



D、bcm2835 使用说明（以 4 线制硬件 SPI 测试程序为例）

a） 先将 OLED 模块按照上述接线和 RaspberryPi 开发板连接起来

b） 将测试程序目

Demo_1.54inch_OLED_64x128_SSD1309_bcm2835_Hardware_4-wire_SPI 拷贝到 RaspberryPi 里(可以通过 SD 卡拷贝,也可以通过 FTP 工具(如 FileZilla)传输)。

c)执行如下命令运行 bcm2835 测试程序:

```
cd Demo_1.54inch_OLED_64x128_SSD1309_bcm2835_Hardware_4-wire_SPI
make
sudo ./1.54_SPI_OLED
```

如下图所示:

```
pi@raspberrypi:~/0821 $ cd 0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI/
pi@raspberrypi:~/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI $ make
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI/source/src/test.
/pi/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI/source/include
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI/source/src/spi.c
i/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI/source/include
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI/source/src/gui.c
i/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI/source/include
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI/source/src/main.
/pi/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI/source/include
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI/source/src/delay
me/pi/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI/source/include
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI/source/src/oled.
/pi/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI/source/include
gcc -g -O0 /home/pi/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI/output/test.o /home/
emo_bcm2835_Hardware_4-wire_SPI/output/gui.o /home/pi/0821/0.96inch_OLED_Demo_bcm2835_Hardwa
elay.o /home/pi/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI/output/oled.o -o 0.96_SP
pi@raspberrypi:~/0821/0.96inch_OLED_Demo_bcm2835_Hardware_4-wire_SPI $ sudo ./0.96_SPI_OLED
```

E、 wiringPi 使用说明(以 4 线制硬件 SPI 测试程序为例)

a) 先将 OLED 模块按照上述接线和 RaspberryPi 开发板连接起来

b)将测试程序目录

Demo_1.54inch_OLED_64x128_SSD1309_wiringPi_Hardware_4-wire_SPI 拷贝到 RaspberryPi 里(可以通过 SD 卡拷贝,也可以通过 FTP 工具(如 FileZilla)传输)。

c)执行如下命令运行 wiringPi 测试程序:

```
cd Demo_1.54inch_OLED_64x128_SSD1309_wiringPi_Hardware_4-wire_SPI
make
sudo ./1.54_SPI_OLED
```

如下图所示:

```
pi@raspberrypi:~/0821 $ cd 0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/
pi@raspberrypi:~/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI $ make
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/src/test.
ome/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/include
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/src/spi.c
e/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/include
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/src/gui.c
e/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/include
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/src/main.
ome/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/include
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/src/delay
/home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/include
gcc -g -O0 -c  /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/src/oled.
ome/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/source/include
gcc -g -O0 /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/output/test.o /home/
_Demo_wiringPi_Hardware_4-wire_SPI/output/gui.o /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Har
put/delay.o /home/pi/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI/output/oled.o -o 0.
pi@raspberrypi:~/0821/0.96inch_OLED_Demo_wiringPi_Hardware_4-wire_SPI $ sudo ./0.96_SPI_OLED
```

F、 python 使用说明（以 4 线制硬件 SPI 测试程序为例）

a）运行python测试程序之前还需要安装图像处理库PIL，具体安装方法见如下文档：

http://www.lcdwiki.com/res/PublicFile/Python_Image_Library_Install_Illustration_CN.pdf

b）将 OLED 模块按照上述接线和 RaspberryPi 开发板连接起来

c）将测试程序目录

Demo_1.54inch_OLED_64x128_SSD1309_python_Hardware_4-wire_SPI 拷贝到 RaspberryPi 里（可以通过 SD 卡拷贝，也可以通过 FTP 工具（如 FileZilla）传输）。

d）执行如下命令分别运行3个python测试程序：

```
cd Demo_1.54inch_OLED_64x128_SSD1309_python_Hardware_4-wire_SPI/source

sudo python show_graph.py

sudo python show_char.py

sudo python show_bmp.py
```

如下图所示：

```
pi@raspberrypi:~/0821 $ cd 0.96inch_OLED_Demo_python_Hardware_4-wire_SPI/source/
pi@raspberrypi:~/0821/0.96inch_OLED_Demo_python_Hardware_4-wire_SPI/source $ sudo python show_graph.py
pi@raspberrypi:~/0821/0.96inch_OLED_Demo_python_Hardware_4-wire_SPI/source $ sudo python show_char.py
pi@raspberrypi:~/0821/0.96inch_OLED_Demo_python_Hardware_4-wire_SPI/source $ sudo python show_bmp.py
```

3、STM32 使用说明

接线说明：

引脚标注见接口说明。

STM32F103C8T6单片机测试程序接线说明			
序号	模块引脚	对应STM32F103C8T6开发板接线引脚	
		SPI	IIC
1	GND	GND	
2	VCC	3.3V/5V	
3	SCL	PA5	
4	SDA	PA7	
5	RES	PB8	
6	DC	PB7	VCC或者GND
7	CS	PB9	GND

STM32F103RCT6单片机测试程序接线说明			
序号	模块引脚	对应MiniSTM32开发板接线引脚	
		SPI	IIC
1	GND	GND	
2	VCC	3.3V/5V	
3	SCL	PB13	
4	SDA	PB15	
5	RES	PB12	
6	DC	PB10	VCC或者GND
7	CS	PB11	GND

STM32F103ZET6单片机测试程序接线说明		
序号	引脚丝印	对应Elite STM32开发板接线引脚

		SPI	IIC
1	GND	GND	
2	VCC	3.3V/5V	
3	SCL	PB13	
4	SDA	PB15	
5	RES	PB12	
6	DC	PB10	VCC或者GND
7	CS	PB11	GND

STM32F407ZGT6单片机测试程序接线说明			
序号	引脚丝印	对应Explorer STM32F4开发板接线引脚	
		SPI	IIC
1	GND	GND	
2	VCC	3.3V/5V	
3	SCL	PB3	
4	SDA	PB5	
5	RES	PB12	
6	DC	PB14	VCC或者GND
7	CS	PB15	GND

STM32F429IGT6单片机测试程序接线说明			
序号	引脚丝印	对应Apollo STM32F4/F7开发板接线	
		SPI	IIC
1	GND	GND	
2	VCC	3.3V/5V	

3	SCL	PF7	
4	SDA	PF9	
5	RES	PD12	
6	DC	PD5	VCC或者GND
7	CS	PD11	GND

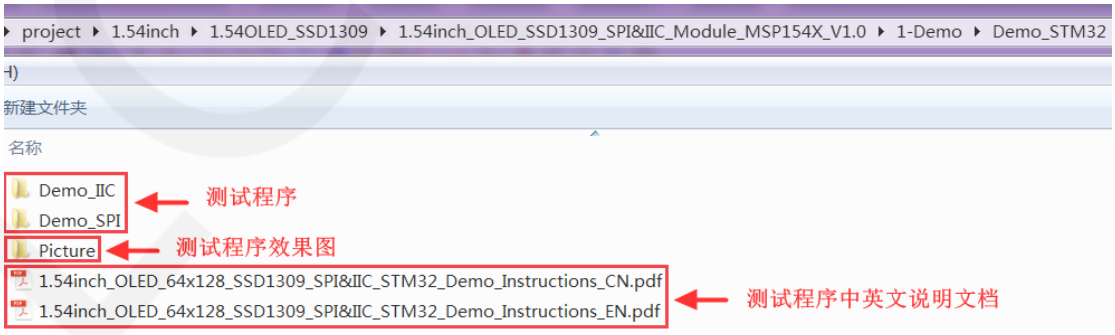
STM32F767IGT6和STM32H743IIT6单片机测试程序接线说明		
序号	模块引脚	对应Apollo STM32F4/F7开发板接线
		SPI
1	GND	GND
2	VCC	3.3V/5V
3	SCL	PB13
4	SDA	PB15
5	RES	PD12
6	DC	PD5
7	CS	PD11

操作步骤：

A、 按照上述接线说明将 IPS 模块和 STM32 单片机连接起来，并上电；

B、 根据单片机型号选择测试示例，如下图所示：

（测试程序说明请查阅测试程序包中测试程序说明文档）



C、打开所选的测试程序工程，进行编译和下载；

关于 STM32 测试程序编译和下载的详细说明见如下文档：

http://www.lcdwiki.com/res/PublicFile/STM32_Keil_Use_Illustration_CN.pdf

D、OLED 模块如果正常显示字符和图形，则说明程序运行成功；

4、C51 使用说明

接线说明：

引脚标注见接口说明。

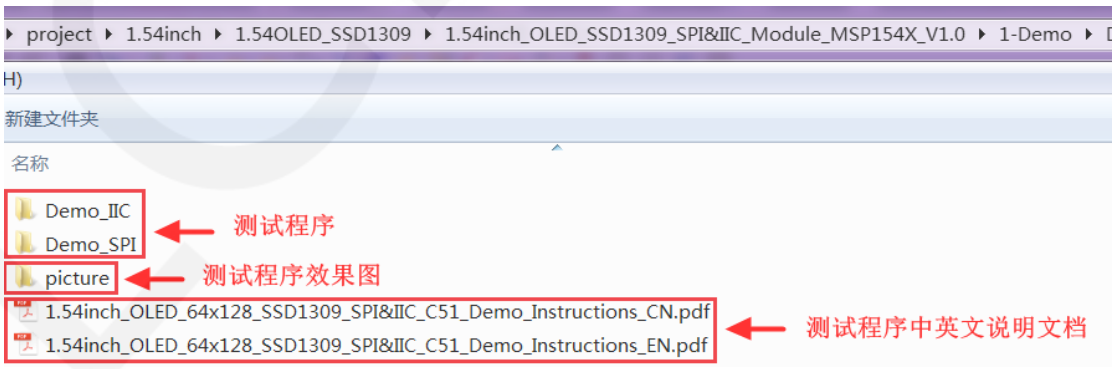
STC89C52RC和STC12C5A60S2单片机测试程序接线说明			
序号	模块引脚	对应STC89/STC12开发板接线引脚	
		SPI	IIC
1	GND	GND	
2	VCC	3.3V/5V	
3	SCL	P17	
4	SDA	P15	
5	RES	P33	
6	DC	P12	VCC或者GND
7	CS	P13	GND

操作步骤：

A、 按照上述接线说明将 IPS 模块和 C51 单片机连接起来，并上电；

B、 选择需要测试的 C51 测试程序，如下图所示：

（测试程序说明请查阅测试程序包中测试程序说明文档）



C、打开所选的测试程序工程，进行编译和下载；

关于 C51 测试程序编译和下载的详细说明见如下文档：

http://www.lcdwiki.com/res/PublicFile/C51_Keil%26stc-isp_Use_Illustration_CN.pdf

D、OLED 模块如果正常显示字符和图形，则说明程序运行成功；

5、MSP430 使用说明

接线说明：

引脚标注见接口说明。

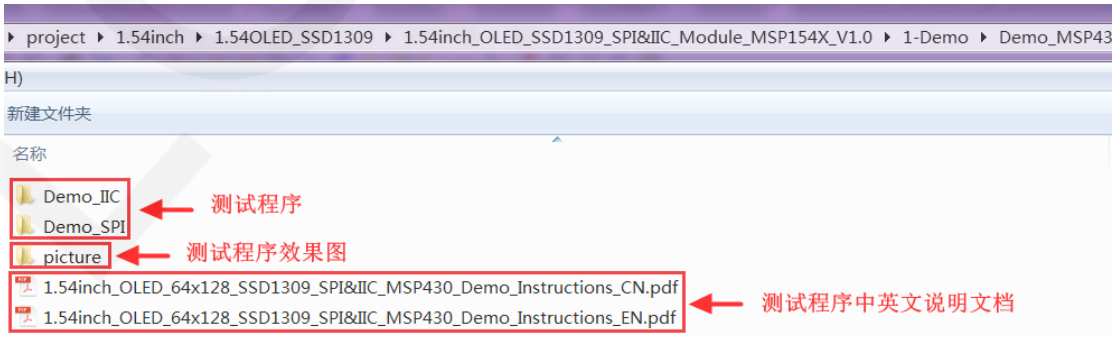
MSP430F149单片机测试程序接线说明			
序号	模块引脚	对应MSP430开发板接线引脚	
		SPI	IIC
1	GND	GND	
2	VCC	3.3V/5V	
3	SCL	P33	
4	SDA	P31	
5	RES	P22	
6	DC	P21	VCC或者GND
7	CS	P20	GND

操作步骤：

A、 按照上述接线说明将 IPS 模块和 MSP430 单片机连接起来，并上电；

B、 选择需要测试的 MSP430 测试程序，如下图所示：

（测试程序说明请查阅测试程序包中测试程序说明文档）



C、打开所选的测试程序工程，进行编译和下载；

关于 MSP430 测试程序编译和下载的详细说明见如下文档：

http://www.lcdwiki.com/res/PublicFile/IAR_IDE%26MspFet_Use_Illustration_CN.pdf

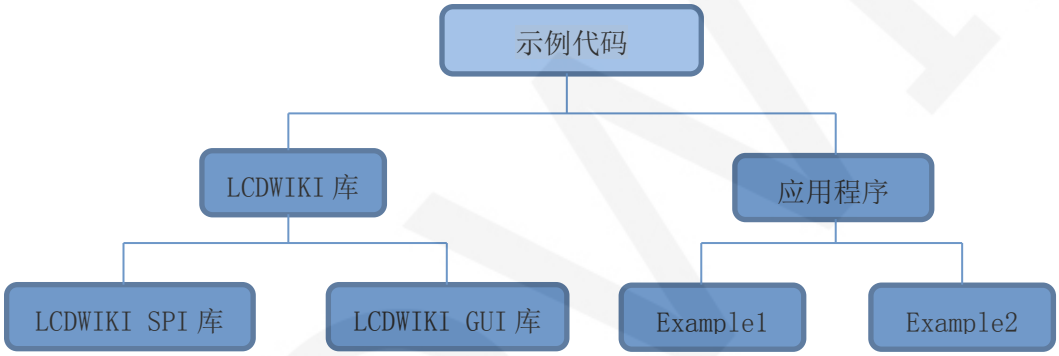
C、 OLED 模块如果正常显示字符和图形，则说明程序运行成功；

软件说明

1、代码架构

A、 Arduino 代码架构说明

代码架构如下图所示：



Arduino 的测试程序代码由两部分组成：LCDWIKI 库和应用代码。

LCDWIKI 库包含两部分内容：LCDWIKI_SPI 库和 LCD_GUI 库。

应用程序包含几个测试示例，每个测试示例包含不同的测试内容

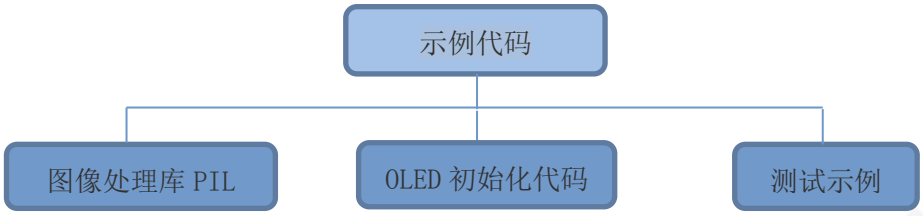
LCDWIKI_SPI 为底层库，和硬件有关联，主要负责操作寄存器，包括硬件模块初始化，数据和命令传输，像素点坐标和颜色设置，显示方式配置等。

LCDWIKI_GUI 为中间层库，主要负责使用底层库提供的 API 实现图形的绘制，字符显示等操作。

应用程序是利用 LCDWIKI 库提供的 API，编写一些测试示例，实现某方面的测试功能。

B、 RaspberryPi 代码架构说明：

python 测试程序代码架构如下图所示：



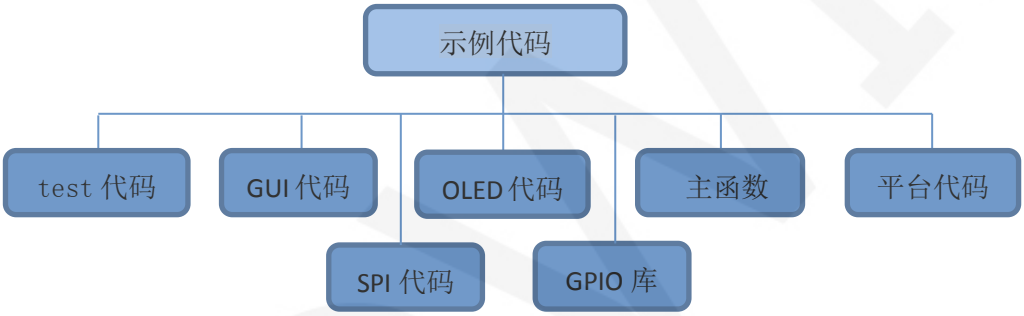
python 测试程序由但部分组成：PIL 图像处理库，OLED 初始化代码，测试示例代码

PIL 图像处理库负责图像绘制，字符和文字显示等操作

OLDE 初始化代码负责操作寄存器，包括硬件模块初始化，数据和命令传输，像素点坐标和颜色设置，显示方式配置等

测试示例则是利用上述两部分代码提供的 API，实现一些测试功能

Bcm2835 和 wiringPi 测试程序代码架构如下：



主程序运行时的 Demo API 代码包含在 test 代码中；

OLED 初始化以及相关的操作都包含在 OLED 代码中；

画点、线、图形以及中英文字符显示相关的操作都包含在 GUI 代码中；

GPIO 库提供 GPIO 操作；

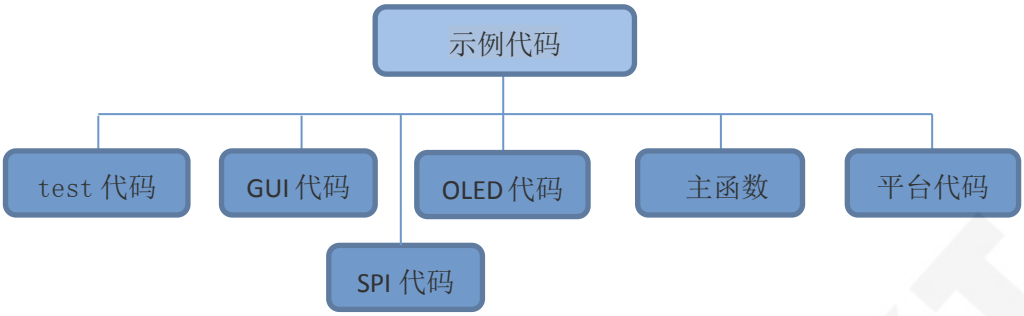
主函数实现应用程序运行；

平台代码因平台而异；

SPI 初始化及配置相关的操作包含在 SPI 代码中；

C、 STM32、C51 以及 MSP430 代码架构说明

STM32 和 C51 测试程序代码架构如下图所示：



主程序运行时的 Demo API 代码包含在 test 代码中；

OLED 初始化以及相关的操作都包含在 OLED 代码中；

画点、线、图形以及中英文字符显示相关的操作都包含在 GUI 代码中；

主函数实现应用程序运行；

平台代码因平台而异；

SPI 初始化及配置相关的操作包含在 SPI 代码中；

2、软件 SPI 和硬件 SPI 说明

该 IPS 模块分别提供了软件 SPI 和硬件 SPI 示例代码（STC89C52RC 只有软件 SPI 功能），两种示例代码在显示内容上没任何区别，但是如下方面有区别：

A、显示速度

硬件 SPI 明显比软件 SPI 要快，这是由硬件决定的。

B、 GPIO 定义

软件 SPI 全部控制引脚都要定义，可以使用任何空闲引脚，硬件 SPI 的数据和时钟信号引脚是固定的（因平台而异），其他控制引脚要自己定义，也可使用任何空闲引脚。

C、初始化

软件 SPI 初始化时，只需要对用于引脚定义的 GPIO 进行初始化，硬件 SPI 初始化时，需要对相关的控制寄存器以及数据寄存器进行初始化。

3、模块 GPIO 定义说明

A、Arduino 测试程序 GPIO 定义说明

Arduino 测试程序 GPIO 定义放在应用示例里，每个应用示例都可以定义 GPIO。如下图所示（以 UNO 单片机 4 线制软件 SPI 测试程序为例）：

```
//paramters define
#define MODEL SSD1306
#define CS    A5
#define DC    A3
#define D1    11
#define D0    13
#define RES   A4
#define LED   -1   //if you don't need to control
```

如果使用软件 SPI，所有引脚定义都可以修改成其他任何空闲的 GPIO。

如果使用硬件 SPI，则 D0 和 D1 不可修改，也不需要定义，其他 GPIO 都可修改。

如果使用 3 线制 SPI，则 DC 不需要定义。

B、RaspberryPi 测试程序 GPIO 定义说明

RaspberryPi 测试程序都是使用硬件 SPI，所以只需定义 3 个 GPIO 口。bcm2835 和 WiringPi 测试程序将 GPIO 定义放在 oled.h 文件里面，如下图所示（以 4 线制 SPI 测试程序为例）：

```
//-----------------OLED module pin definition----------------
#define OLED_CS   8   //chip selection control signal  bcm:8
#define OLED_DC   2   //data or command selection control signal bcm:2
#define OLED_RST  3   //reset control signal  bcm:3
```

Python 测试程序将 GPIO 定义放在每个测试示例里，如下图所示（以 4 线制 SPI 测试程序为例）：

```
#·RaspberryPi·pin·configuration:
DC·=·2
RES·=·3
CS·=·8
```

此三个 GPIO 可以根据相应的 GPIO 库编码来修改。

如果使用 3 线制 SPI，则 OLED_DC 或者 DC 不需要定义。

C、STM32 测试程序 GPIO 定义说明

STM32 测试程序 GPIO 定义分为两部分：控制 GPIO 定义和 SPI GPIO 定义

控制 GPIO 定义放在 oled.h 里，SPI GPIO 定义放在 spi.h 里，分别如下图所示（以 STM32F103RCT6 软件 4 线制 SPI 测试程序为例）：

```
//-----------------OLED端口定义----------------
#define OLED_CS    GPIO_Pin_11    //片选信号          PB11
#define OLED_DC    GPIO_Pin_10    //数据/命令控制信号  PB10
#define OLED_RST   GPIO_Pin_12    //复位信号          PB12
```

```
//--------------SPI总线引脚定义-----------------------
#define OLED_MOSI       GPIO_Pin_15  //OLED屏SPI写数据信号
#define OLED_CLK        GPIO_Pin_13  //OLED屏SPI时钟信号
```

如果使用软件 SPI，所有引脚定义都可以修改成其他任何空闲的 GPIO。

如果使用硬件 SPI，则 OLED_MOSI 和 OLED_CLK 不可修改，也不需要定义，其他 GPIO 都可修改。

如果使用 3 线制 SPI，则 OLED_DC 不需要定义

修改完 GPIO 定义后，需要到 oled.c 文件的 OLED_Init_GPIO 函数里将 GPIO 初始化做相应的修改。

D、C51 测试程序 GPIO 定义说明

C51 测试程序 GPIO 定义分为两部分：控制 GPIO 定义和 SPI GPIO 定义

控制 GPIO 定义放在 oled.h 里，SPI GPIO 定义放在 spi.h 里，分别如下图所示（以 STC12C5A60S2 软件 4 线制 SPI 测试程序为例）：

```
//------------------OLED端口定义----------------
sbit OLED_CS  = P1^3;     //片选信号              P13
sbit OLED_DC  = P1^2;    //数据/命令控制信号   P12
sbit OLED_RST = P3^3;     //复位信号              P33
//SPI的数据引脚定义和的钟引脚定义都可以任意修改
sbit OLED_MOSI = P1^5;    //OLED屏SPI写数据引脚 P15
sbit OLED_CLK = P1^7;     //OLED屏SPI时钟引脚   P17
```

如果使用软件 SPI，所有引脚定义都可以修改成其他任何空闲的 GPIO。

如果使用硬件 SPI，则 OLED_MOSI 和 OLED_CLK 不可修改，也不需要定义，其他 GPIO 都可修改。（只有 STC12C5A60S2 单片机才有硬件 SPI 功能）

如果使用 3 线制 SPI，则 OLED_DC 不需要定义

E、MSP430 测试程序 GPIO 定义说明

MSP430 的液晶屏非 SPI 的 GPIO 定义放在 lcd.h 里面，如下图所示(以 MSP430F149 软件 4 线制 SPI 测试程序为例)：

```
//-----------------OLED端口定义----------------
#define OLED_CS   BIT0    //片选信号          P20
#define OLED_DC   BIT1    //数据/命令控制信号  P21
#define OLED_RST  BIT2    //复位信号          P22
```

所有引脚定义都可以修改，可以定义成其他任何空闲的 GPIO。

如果使用 3 线制 SPI，则 OLED_DC 不需要定义

MSP430 的液晶屏 SPI 的 GPIO 定义放在 spi.h 里面，如下图所示（以 MSP430F149 软件 4 线制 SPI 测试程序为例）：

```
//本测试程序使用的是软件SPI接口驱动
//SPI时钟信号以及SPI读、写信号引脚都可以更改

#define SPI_SCLK BIT3    //P33
#define SPI_MOSI BIT1    //P31
```

如果使用软件 SPI，所有引脚定义都可以修改，可以定义成其他任何空闲的 GPIO。

如果使用硬件 SPI，这些引脚都不需要定义

4、SPI 通信代码实现

A、 Arduino 测试程序 SPI 通信代码实现

SPI 通信代码都在 LCDWIKI_SPI 库里实现。

4 线制软件和硬件 SPI 代码实现如下图所示：

```
//spi write for hardware and software
void LCDWIKI_SPI::Spi_Write(uint8_t data)
{
    if(hw_spi)
    {
        SPI.transfer(data);
    }
    else
    {
        uint8_t val = 0x80;
        while(val)
        {
            if(data&val)
            {
                MOSI_HIGH;
            }
            else
            {
                MOSI_LOW;
            }
            CLK_LOW;
            CLK_HIGH;
            val >>= 1;
        }
    }
} « end Spi_Write »
```

3 线制软件和硬件 SPI 代码实现如下图所示：

```
void LCDWIKI_SPI::Spi_3_wire_Write(uint8_t data,uint8_t cmd)
{
    uint16_t txdata = 0;
    txdata = ((cmd<<15)|(data<<7));
    if(hw_spi)
    {
        SPI.transfer16(txdata);
    }
    else
    {
        uint16_t val = 0x8000;
        while(val>(1<<6))
        {
            if(txdata&val)
            {
                MOSI_HIGH;
            }
            else
            {
                MOSI_LOW;
            }
            CLK_LOW;
            CLK_HIGH;
            val >>= 1;
        }
    }
} « end Spi_3_wire_Write »
```

都是通过标志位来决定使用软件 SPI 还是硬件 SPI

B、 RaspberryPi 测试程序 SPI 通信代码实现

bcm2835 和 wiringPi 测试程序的 SPI 通信代码在 spi.c 中实现。

python 测试程序的 SPI 通信代码在 oled.py 中实现。

bcm2835 测试程序 4 线制硬件 SPI 代码实现如下图所示：

```
/*********************************************************************************
 * @name       :void SPI_WriteByte(uint8_t byte)
 * @date       :2018-08-27
 * @function   :Write a byte of data using RaspberryPi hardware SPI
 * @parameters :Byte:Data to be written
 * @retvalue   :Data received by the bus
*********************************************************************************/
void SPI_WriteByte(uint8_t byte)
{
    bcm2835_spi_transfer(byte);
}
```

bcm2835 测试程序 3 线制硬件 SPI 代码实现如下图所示：

```
/*********************************************************************************
 * @name        :void SPI_WriteByte(uint8_t byte, uint8_t cmd)
 * @date        :2018-08-27
 * @function    :Write a byte of data using RaspberryPi hardware SPI
 * @parameters  :Byte:Data to be written
                 cmd:0-command
                     1-data
 * @retvalue    :Data received by the bus
*********************************************************************************/
void SPI_WriteByte(uint8_t byte, uint8_t cmd)
{
    uint16_t data=0;
    char txbuf[2]={0};
    data=((cmd<<15)|(byte<<7));
    txbuf[0]=(char)(data>>8);
    txbuf[1]=(char)(data&0xFF);
        bcm2835_spi_transfern(txbuf,2);
}
```

wiringPi 测试程序 4 线制硬件 SPI 代码实现如下图所示：

```
/*********************************************************************************
 * @name        :void SPI_WriteByte(uint8_t byte)
 * @date        :2018-08-27
 * @function    :Write a byte of data using RaspberryPi hardware SPI
 * @parameters  :Byte:Data to be written
 * @retvalue    :Data received by the bus
*********************************************************************************/
void SPI_WriteByte(uint8_t byte)
{
    wiringPiSPIDataRW(CHANNEL,&byte,1);
}
```

wiringPi 测试程序 3 线制硬件 SPI 代码实现如下图所示：

```
/*********************************************************************************
 * @name        :void SPI_WriteByte(uint8_t byte, uint8_t cmd)
 * @date        :2018-08-27
 * @function    :Write a byte of data using RaspberryPi hardware SPI
 * @parameters  :Byte:Data to be written
                 cmd:0-command
                     1-data
 * @retvalue    :Data received by the bus
*********************************************************************************/
void SPI_WriteByte(uint8_t byte, uint8_t cmd)
{
    uint16_t data;
    unsigned char txbuf[2]={0};
    data=((cmd<<15)|(byte<<7));
    txbuf[0]=(unsigned char)(data>>8);
    txbuf[1]=(unsigned char)(data&0xFF);
    wiringPiSPIDataRW(CHANNEL,txbuf,2);
}
```

python 测试程序 4 线制硬件 SPI 代码实现如下图所示：

```
    def writebyte(self,val,flag):
        """send one byte data to oled module"""
        if flag == OLED_COMMAND:
            GPIO.output(self.oleddc,GPIO.LOW)
        else:
            GPIO.output(self.oleddc,GPIO.HIGH)
        GPIO.output(self.oledcs,GPIO.LOW)
        self.oledspi.writebytes([val])
        self.oledspi.xfer([val],8000000)
        GPIO.output(self.oledcs,GPIO.HIGH)
```

python 测试程序 3 线制硬件 SPI 代码实现如下图所示：

```
    def writebyte(self,val,flag):
        """send two byte data to oled module"""
        data=((flag<<15)|(val<<7))
        txbuf=[(data>>8)&0xFF,data&0xFF]
        GPIO.output(self.oledcs,GPIO.LOW)
        self.oledspi.writebytes(txbuf)
        self.oledspi.xfer(txbuf,8000000)
        GPIO.output(self.oledcs,GPIO.HIGH)
```

C、STM32 测试程序 SPI 通信代码实现

SPI 通信代码都在 spi.c 中实现。（以 STM32F103RCT6 测试程序为例）

4 线制软、硬件 SPI 通信代码实现如下所示：

软件 SPI：

```
/*****************************************************************
 * @name        :void SPI_WriteByte(u8 Data)
 * @date        :2018-08-27
 * @function    :Write a byte of data using STM32's Software SPI
 * @parameters  :Data:Data to be written
 * @retvalue    :None
*****************************************************************/
void SPI_WriteByte(u8 Data)
{
  unsigned char i=0;
  for(i=8;i>0;i--)
  {
    if(Data&0x80)
    {
      OLED_MOSI_SET(); //写数据1
    }
    else
    {
      OLED_MOSI_CLR(); //写数据0
    }
    OLED_CLK_CLR();     //将时钟拉低拉高
    OLED_CLK_SET();     //发送1bit数据
    Data<<=1;
  }
}
```

硬件 SPI：

```
/*****************************************************************************
 * @name        :u8 SPI_WriteByte(SPI_TypeDef* SPIx,u8 Byte)
 * @date        :2018-08-27
 * @function    :Write a byte of data using STM32's hardware SPI
 * @parameters  :SPIx: SPI type,x for 1,2,3
                 Byte:Data to be written
 * @retvalue    :Data received by the bus
*****************************************************************************/
u8 SPI_WriteByte(SPI_TypeDef* SPIx,u8 Byte)
{
  while((SPIx->SR&SPI_I2S_FLAG_TXE)==RESET);     //等待发送区空
  SPIx->DR=Byte;     //发送一个byte
  while((SPIx->SR&SPI_I2S_FLAG_RXNE)==RESET);//等待接收完一个byte
  return SPIx->DR;                  //返回收到的数据
}
```

3 线制软、硬件 SPI 通信代码实现如下所示：

软件 SPI：

```
/*****************************************************************************
 * @name        :void SPI_WriteByte(u8 data,u8 Cmd)
 * @date        :2018-08-27
 * @function    :Write a byte of data using STM32's Software SPI
 * @parameters  :Data:Data to be written
        Cmd:0-command
            1-data
 * @retvalue    :None
*****************************************************************************/
void SPI_WriteByte(u8 data,u8 Cmd)
{
  unsigned char i=0;
  u16 Data;
  Data = ((Cmd<<15)|(data<<7));
  for(i=9;i>0;i--)
  {
    if(Data&0x8000)
    {
      OLED_MOSI_SET(); //写数据1
    }
    else
    {
      OLED_MOSI_CLR(); //写数据0
    }
    OLED_CLK_CLR();     //将时钟拉低拉高
    OLED_CLK_SET();     //发送1bit数据
    Data<<=1;
  }
}
```

硬件 SPI：

```
/*****************************************************************************
 * @name       :u8 SPI_WriteByte(SPI_TypeDef* SPIx,u8 Byte,u8 cmd)
 * @date       :2018-08-27
 * @function   :Write a byte of data using STM32's hardware SPI
 * @parameters :SPIx: SPI type,x for 1,2,3
                Byte:Data to be written
                cmd:0-write command
                    1-write data
 * @retvalue   :Data received by the bus
*****************************************************************************/
u8 SPI_WriteByte(SPI_TypeDef* SPIx,u8 Byte,u8 cmd)
{
  while((SPIx->SR&SPI_I2S_FLAG_TXE)==RESET);     //等待发送区空
  SPIx->DR=((cmd<<15)|(Byte<<7));    //发送两个byte
  while((SPIx->SR&SPI_I2S_FLAG_RXNE)==RESET);//等待接收完两个byte
  return SPIx->DR;                   //返回收到的数据
}
```

D、C51 测试程序 SPI 通信代码实现

SPI 通信代码都在 spi.c 中实现。（以 STC12C5A60S2 测试程序为例）

4 线制软、硬件 SPI 通信代码实现如下所示：

软件 SPI：

```
/*****************************************************************************
 * @name       :void SPI_WriteByte(u8 byte)
 * @date       :2018-08-09
 * @function   :Write a byte of data using C
 * @parameters :byte:Data to be written
 * @retvalue   :None
*****************************************************************************/
void SPI_WriteByte(u8 byte)
{
  u8 i;
  for(i=0;i<8;i++)
  {
    if(byte&0x80)
    {
      OLED_MOSI_Set();
    }
    else
    {
      OLED_MOSI_Clr();
    }
    OLED_CLK_Clr();
    OLED_CLK_Set();
    byte<<=1;
  }
}
```

硬件 SPI：

```
/*************************************************************
 * @name        :void SPI_WriteByte(u8 byte)
 * @date        :2018-08-09
 * @function    :Write a byte of data using C51's Hardware SPI
 * @parameters  :byte:Data to be written
 * @retvalue    :None
*************************************************************/
void SPI_WriteByte(u8 byte)
{
      SPDAT = byte;    //发送一个字节
      while((SPSTAT & SPIF)==0) ; //等待发送完成
      SPSTAT = SPIF+WCOL;      //清0 SPIF和WCOL标志
}
```

3 线制软、硬件 SPI 通信代码实现如下所示：

软件 SPI：

```
/*************************************************
 * @name        :void SPI_WriteByte(u8 byte, u8 cmd)
 * @date        :2018-08-09
 * @function    :Write a byte of data using C51's so
 * @parameters  :byte:Data to be written
                 cmd:0-command
                     1-data
 * @retvalue    :None
*************************************************/
void SPI_WriteByte(u8 byte, u8 cmd)
{
  u8 i;
  u16 Data=0;
  Data=((cmd<<15)|(byte<<7));
  for(i=0;i<9;i++)
  {
    if(Data&0x8000)
    {
      OLED_MOSI_Set();
    }
    else
    {
      OLED_MOSI_Clr();
    }
    OLED_CLK_Clr();
    OLED_CLK_Set();
    Data<<=1;
  }
}
```

硬件 SPI：

```
/*****************************************************************
 * @name        :void SPI_WriteByte(u8 byte, u8 cmd)
 * @date        :2018-08-09
 * @function    :Write a byte of data using C51's Hardware SPI
 * @parameters  :byte:Data to be written
                 cmd:0-command
                     1-data
 * @retvalue    :None
*****************************************************************/
void SPI_WriteByte(u8 byte, u8 cmd)
{
    u8 i=0;
    u16 Data=0;
    Data=((cmd<<15)|(byte<<7));
    for(i=2;i>0;i--)
    {
      SPDAT = (Data>>((i-1)*8));     //发送一个字节
      while((SPSTAT & SPIF)==0) ; //等待发送完成
      SPSTAT = SPIF+WCOL;       //清0 SPIF和WCOL标志
    }

}
```

E、MSP430 测试程序 SPI 通信代码实现

软件 SPI 通信代码在 spi.c 中实现。

4 线制软、硬件 SPI 通信代码实现如下所示：

软件 SPI：

```
/*****************************************************************
 * @name        :void  SPI_WriteByte(u8 Data)
 * @date        :2018-08-09
 * @function    :Write a byte of data using STM32's hardware SPI
 * @parameters  :SPIx: SPI type,x for 1,2,3
                 Byte:Data to be written
 * @retvalue    :Data received by the bus
*****************************************************************/
void SPI_WriteByte(u8 Data)
{
        unsigned char i=0;
        for(i=8;i>0;i--)
        {
          if(Data&0x80)
          SPI_MOSI_SET; //输出数据
      else SPI_MOSI_CLR;

      SPI_SCLK_CLR;
      SPI_SCLK_SET;
      Data<<=1;
        }
}
```

硬件 SPI：

```
/*****************************************************************************
 * @name       :u8 SPI_WriteByte(SPI_TypeDef* SPIx,u8 Byte)
 * @date       :2018-08-09
 * @function   :Write a byte of data using STM32's hardware SPI
 * @parameters :SPIx: SPI type,x for 1,2,3
               Byte:Data to be written
 * @retvalue   :Data received by the bus
*****************************************************************************/
u8 SPI_WriteByte(u8 Byte)
{
  while ((IFG1&UTXIFG0) ==0);   // wait while not ready / for RX
  U0TXBUF = Byte;
  while ((IFG1&URXIFG0)==0);   // wait for RX buffer (full)
  return (U0RXBUF);
}
```

3 线制软、硬件 SPI 通信代码实现如下所示：

软件 SPI：

```
/*****************************************************************
 * @name       :void  SPI_WriteByte(u8 Data)
 * @date       :2018-08-09
 * @function   :Write a byte of data using STM32's hardware SPI
 * @parameters :SPIx: SPI type,x for 1,2,3
               Byte:Data to be written
 * @retvalue   :Data received by the bus
*****************************************************************/
void SPI_WriteByte(u8 val, u8 cmd)
{
        unsigned char i=0;
        u16 Data=0;
        Data = ((cmd<<15)|(val<<7));
        for(i=9;i>0;i--)
        {
          if(Data&0x8000)
          SPI_MOSI_SET; //输出数据
          else SPI_MOSI_CLR;

          SPI_SCLK_CLR;
          SPI_SCLK_SET;
          Data<<=1;
        }
}
```

硬件 SPI：

```
/*****************************************************************
 * @name       :void OLED_WR_Byte(unsigned dat,unsigned cmd)
 * @date       :2018-08-27
 * @function   :Write a byte of content to the OLED screen
 * @parameters :dat:Content to be written
               cmd:0-write command
                                                                1-
 * @retvalue   :None
*****************************************************************/
void OLED_WR_Byte(unsigned dat,unsigned cmd)
{
        u16 data=0;
        data=((cmd<<15)|(dat<<7));
        OLED_CS_Clr;
        SPI_WriteByte((data>>8)&0xFF);
        SPI_WriteByte(data&0xFF);
        OLED_CS_Set;
}
```

常用软件

本套测试示例需要显示中英文、符号以及图片，所以要用到 PCtoLCD2002 取模软件。这里只针对该套测试程序说明一下取模软件的设置。

本套测试程序 PCtoLCD2002 取模软件设置如下：

点阵格式选择**阴码**

取模方式选择**逐行式（C51 和 MSP430 测试程序需要选择行列式）**

取模走向选择**顺向（高位在前）（C51 和 MSP430 测试程序需要选择逆向（低位在前））**

输出数制选择**十六进制数**

自定义格式选择 **C51 格式**

具体设置方法见如下网页：

http://www.lcdwiki.com/zh/%E3%80%90%E6%95%99%E7%A8%8B%E3%80%91%E4%B8%AD%E8%8B%B1%E6%96%87%E6%98%BE%E7%A4%BA%E5%8F%96%E6%A8%A1%E8%AE%BE%E7%BD%AE