1.54inch OLED SSD1309 IIC Module MC154GW&MC154GB 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,可显示黑白或者黑蓝双色。其采用 IIC 通信方式，内部驱动 IC 为 SSD1309。

产品特点

- 1.54 寸 OLED 屏，支持黑白或黑蓝双色显示
- 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
- 采用 IIC 总线，只需几个 IO 即可点亮显示
- 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 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	黑白色/黑蓝色
SKU	MC154GW MC154GB
尺寸	1.54(inch)
类型	OLED
驱动芯片	SSD1309
分辨率	128*64(Pixel)
模块接口	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

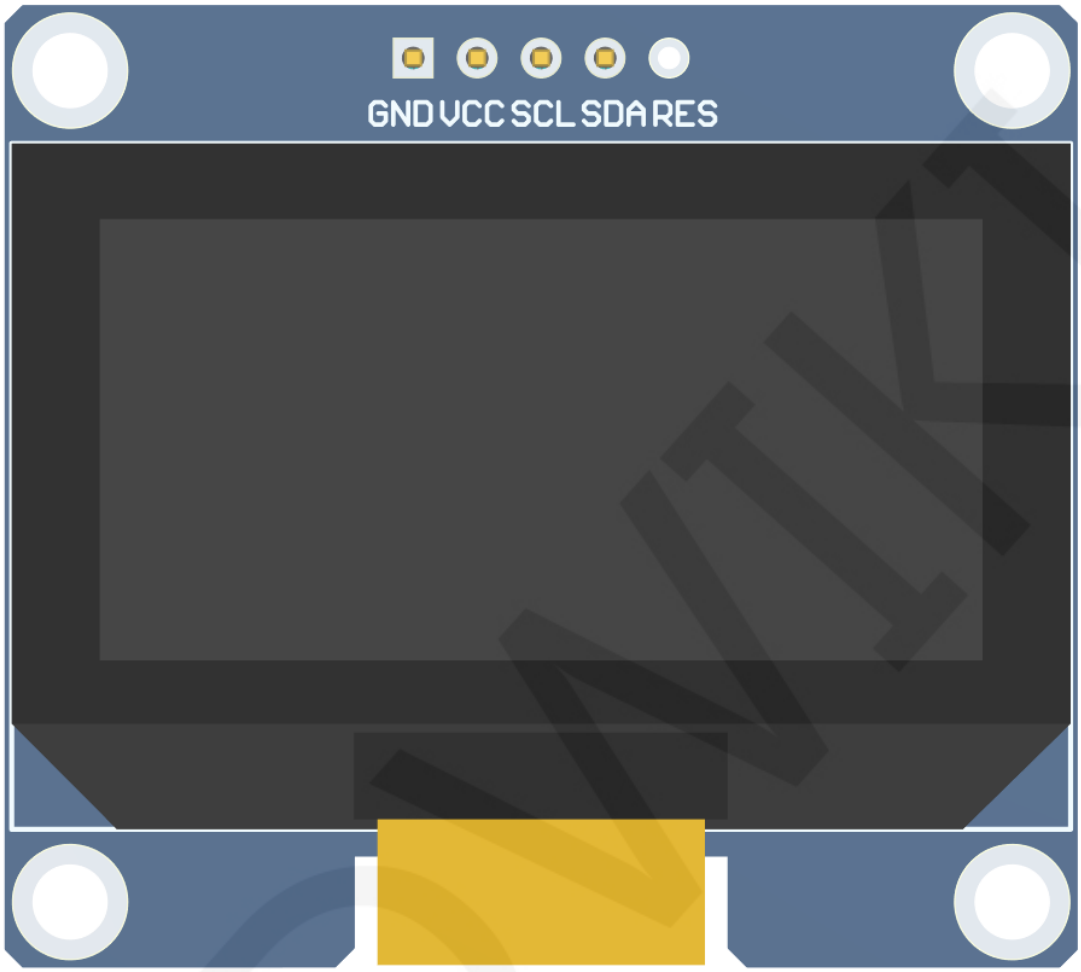


图1. 模块引脚丝印图

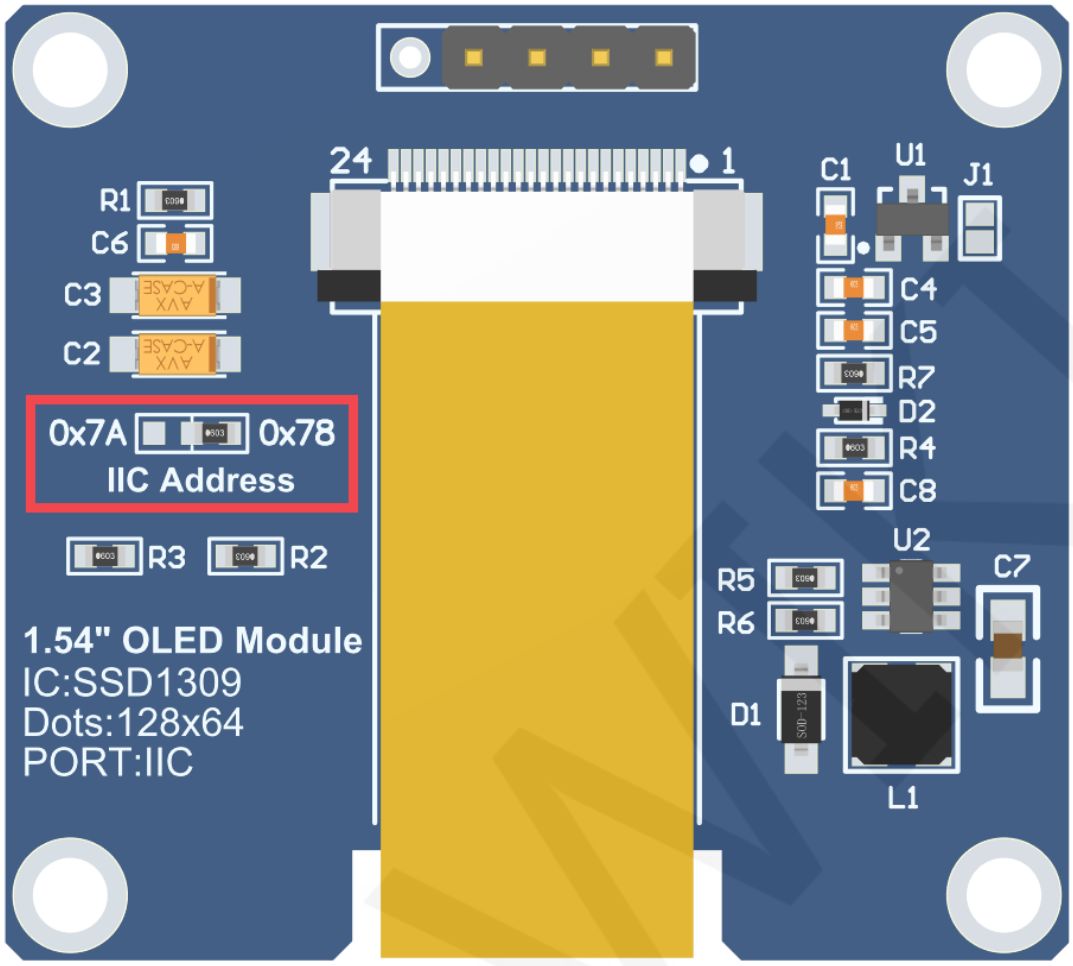


图2. 模块背面图

注意：

1、 本模块支持IIC从设备地址切换（如图2红框内所示），具体说明如下：

A、焊接0x78一侧电阻，断开0x7A一侧，则选择0x78从设备地址（默认）；

B、焊接0x7A一侧电阻，断开0x78一侧，则选择0x7A从设备地址；

2、 硬件切换了IIC从设置地址，软件上也要做相应的修改，具体修改方法见以下IIC从设备地址修改说明；

标号	PIN	引脚说明
1	GND	OLED 显示模块电源地
2	VCC	OLED 显示模块电源正（3.3V/5V）
3	SCL	OLED 显示模块 IIC 总线时钟信号

4	SDA	OLED 显示模块 IIC 总线数据信号
5	RES	OLED 复位信号，低电平复位（模块有复位电路，可以上电复位）

硬件配置

该模块硬件电路由 5 部分组成：OLED 显示控制电路、OLED 升压电路、IIC 从设备地址选择电路、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。

IIC 从设备地址选择控制电路用于选择不同的从设备地址。

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 模块采用 IIC 通信方式，硬件配置 2 个引脚：SCL（IIC 数据引脚）、SDA（IIC 时钟引脚）。按照 IIC 工作时序来控制此 2 个引脚就可以完成 IIC 数据传输。

工作原理

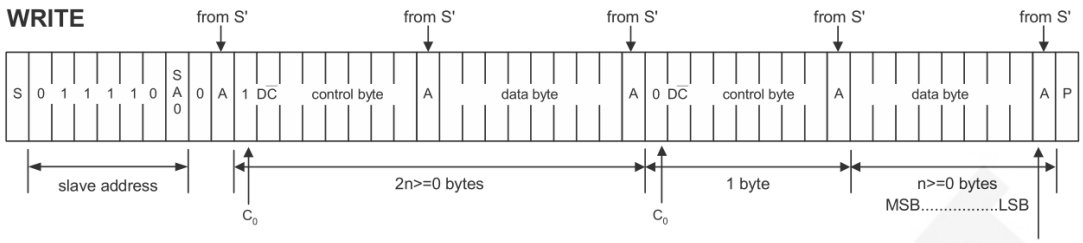
1、SSD1309 控制器简介

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、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：

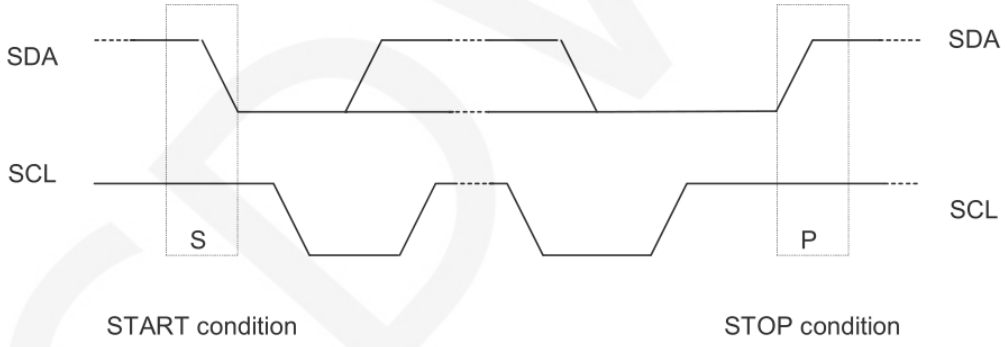
C_0=0：此为最后一个控制字节，接下里发送的都是数据字节

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

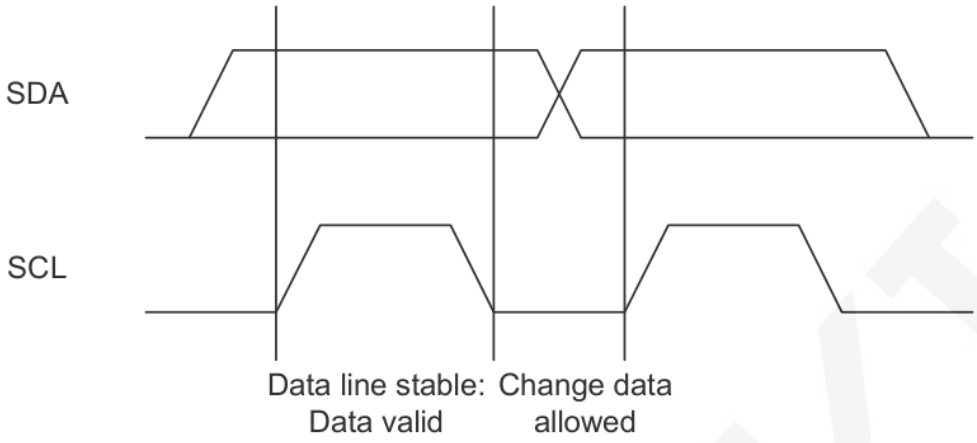
D/$\overline{C}$=1：为 RAM 数据操作字节

IIC 开始和停止时序图如下：



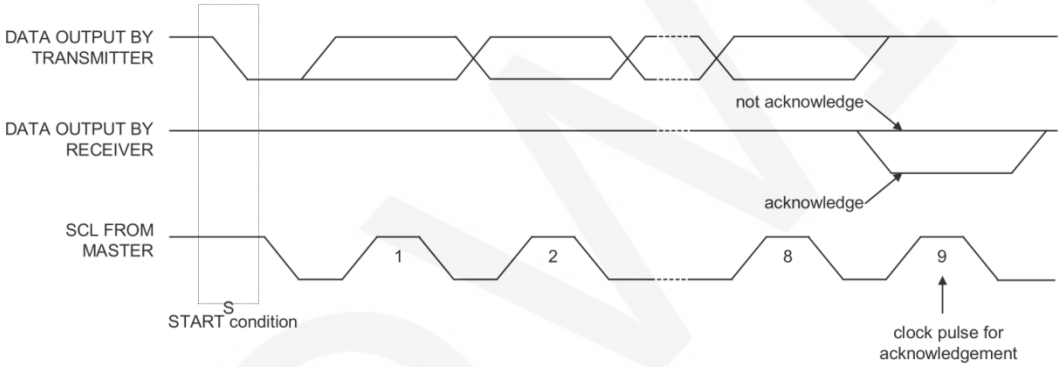
当 IIC 的数据线和时钟线都保持高电平时，IIC 为空闲状态，此时数据线由高电平变为低电平，时钟线继续保持高电平，IIC 总线就启动数据传输。当时钟线保持高电平时，数据线由低电平变为高电平，IIC 总线停止数据传输。

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**
1	GND	GND
2	VCC	5V/3.3V

3	SCL	A5
4	SDA	A4
5	RES	不需要接

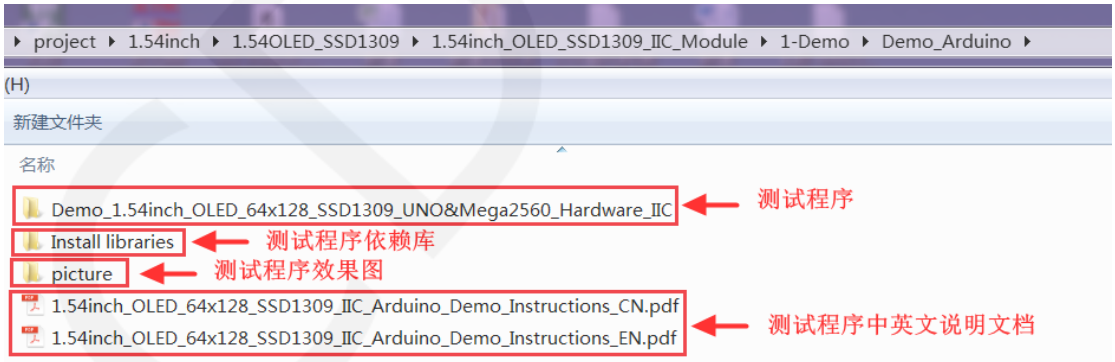
Arduino MEGA2560单片机测试程序接线说明		
序号	**模块引脚**	**对应MEGA2560开发板接线引脚**
1	GND	GND
2	VCC	5V/3.3V
3	SCL	21
4	SDA	20
5	RES	不需要接

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、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	**引脚丝印**	**对应STM32F103C8T6开发板接线引脚**
1	GND	GND
2	VCC	5V/3.3V
3	SCL	PA5
4	SDA	PA7
5	RES	不需要接

STM32F103RCT6单片机测试程序接线说明		
序号	**引脚丝印**	**对应MiniSTM32开发板接线引脚**
1	GND	GND
2	VCC	5V/3.3V
3	SCL	PB13
4	SDA	PB15
5	RES	不需要接

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**
1	GND	GND
2	VCC	5V/3.3V
3	SCL	PB13
4	SDA	PB15

5	RES	不需要接

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**
1	GND	GND
2	VCC	5V/3.3V
3	SCL	PB3
4	SDA	PB5
5	RES	不需要接

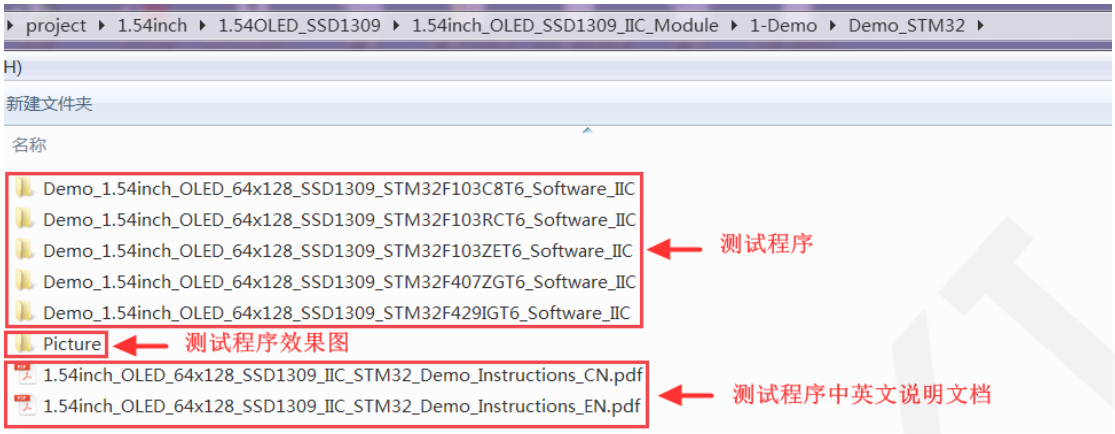
STM32F429IGT6单片机测试程序接线说明		
序号	**引脚丝印**	**对应Apollo STM32F4/F7开发板接线引脚**
1	GND	GND
2	VCC	5V/3.3V
3	SCL	PF7
4	SDA	PF9
5	RES	不需要接

操作步骤：

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

B、 选择需要测试的 STM32 测试程序，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；

3、C51 使用说明

接线说明：

引脚标注见接口说明

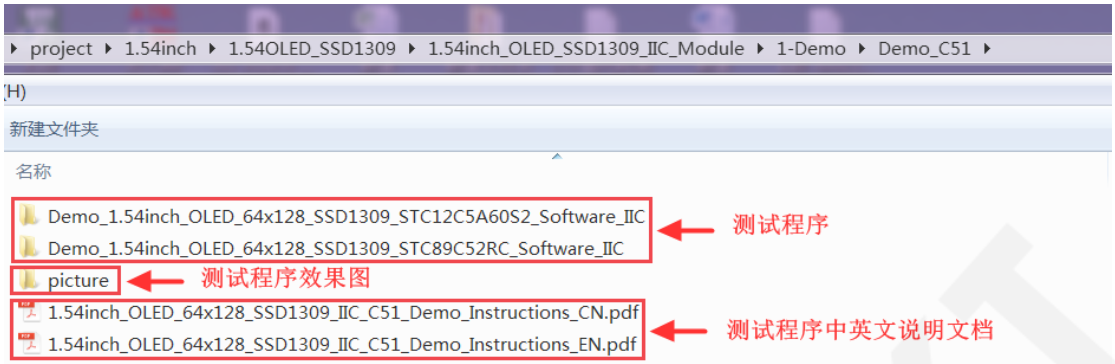
STC89C52RC和STC12C5A60S2单片机测试程序接线说明		
序号	**引脚丝印**	**对应STC89/STC12开发板接线引脚**
1	GND	GND
2	VCC	5V/3.3V
3	SCL	P17
4	SDA	P15
5	RES	不需要接

操作步骤：

A、按照上述接线说明将 OLED 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；

4、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 4. 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**
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：5 BCM编码：3 wiringPi编码：9
4	SDA	物理引脚：3 BCM编码：2 wiringPi编码：8

操作步骤：

A、开启 RaspberryPi 的 IIC 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->I2C->YES

启动 RaspberryPi 的 I2C 内核驱动

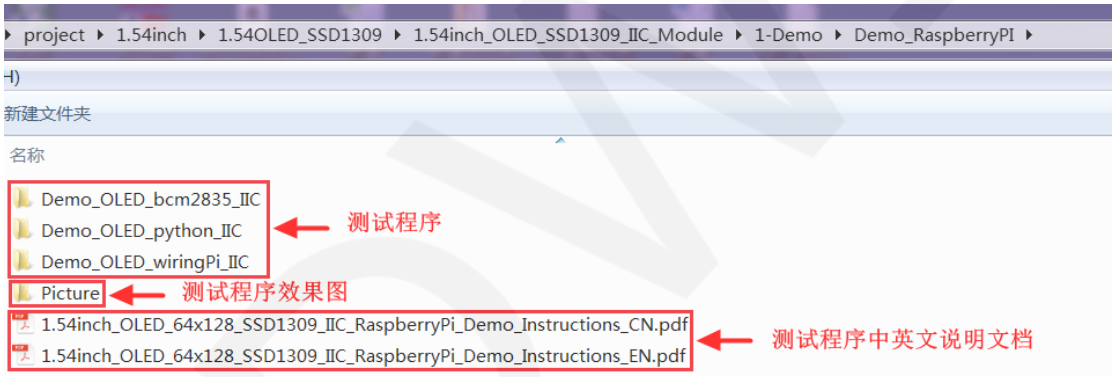
B、安装函数库

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

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_OLED_bcm2835_IIC 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_OLED_bcm2835_IIC

make

sudo ./1.54_IIC_OLED
```

如下图所示：

```
pi@raspberrypi:~/csl $
pi@raspberrypi:~/csl $ cd Demo_OLED_bcm2835_IIC/
pi@raspberrypi:~/csl/Demo_OLED_bcm2835_IIC $ make
gcc -g -O0 -c  /home/pi/csl/Demo_OLED_bcm2835_IIC/source/src/test.c -o /home/pi/csl/
gcc -g -O0 -c  /home/pi/csl/Demo_OLED_bcm2835_IIC/source/src/gui.c -o /home/pi/csl/I
gcc -g -O0 -c  /home/pi/csl/Demo_OLED_bcm2835_IIC/source/src/main.c -o /home/pi/csl/
gcc -g -O0 -c  /home/pi/csl/Demo_OLED_bcm2835_IIC/source/src/delay.c -o /home/pi/csl
gcc -g -O0 -c  /home/pi/csl/Demo_OLED_bcm2835_IIC/source/src/oled.c -o /home/pi/csl/
gcc -g -O0 -c  /home/pi/csl/Demo_OLED_bcm2835_IIC/source/src/iic.c -o /home/pi/csl/I
gcc -g -O0 /home/pi/csl/Demo_OLED_bcm2835_IIC/output/test.o /home/pi/csl/Demo_OLED_I
IC/output/delay.o /home/pi/csl/Demo_OLED_bcm2835_IIC/output/oled.o /home/pi/csl/Demo
pi@raspberrypi:~/csl/Demo_OLED_bcm2835_IIC $ sudo ./1.3_IIC_OLED
```

E、 wiringPi 使用说明

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_OLED_wiringPi_IIC 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_OLED_wiringPi_IIC

make

sudo ./1.54_IIC_OLED
```

如下图所示：

```
pi@raspberrypi:~/csl $
pi@raspberrypi:~/csl $ cd Demo_OLED_wiringPi_IIC/
pi@raspberrypi:~/csl/Demo_OLED_wiringPi_IIC $ make
gcc -g -O0 -c  /home/pi/csl/Demo_OLED_wiringPi_IIC/source/src/test.c -o /home/pi/csl
gcc -g -O0 -c  /home/pi/csl/Demo_OLED_wiringPi_IIC/source/src/gui.c -o /home/pi/csl/
gcc -g -O0 -c  /home/pi/csl/Demo_OLED_wiringPi_IIC/source/src/main.c -o /home/pi/csl
gcc -g -O0 -c  /home/pi/csl/Demo_OLED_wiringPi_IIC/source/src/delay.c -o /home/pi/csl
gcc -g -O0 -c  /home/pi/csl/Demo_OLED_wiringPi_IIC/source/src/oled.c -o /home/pi/csl
gcc -g -O0 -c  /home/pi/csl/Demo_OLED_wiringPi_IIC/source/src/iic.c -o /home/pi/csl/
gcc -g -O0 /home/pi/csl/Demo_OLED_wiringPi_IIC/output/test.o /home/pi/csl/Demo_OLED_
Pi_IIC/output/delay.o /home/pi/csl/Demo_OLED_wiringPi_IIC/output/oled.o /home/pi/csl
pi@raspberrypi:~/csl/Demo_OLED_wiringPi_IIC $ sudo ./1.3_IIC_OLED
```

如果想修改 IIC 传输速率，需要在/boot/config.txt 文件里添加如下内容，然后重启 raspberryPi

,i2c_arm_baudrate=2000000（注意逗号也需要）

如下图所示（红色框内为添加的内容，数字 2000000 为设置的速率，可更改）：

```
# Uncomment some or all of these to enable the optional hardware interfaces
dtparam=i2c_arm=on,i2c_arm_baudrate=2000000
```

F、 python 使用说明

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_OLED_python_IIC 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_OLED_python_IIC/source

sudo python show_graph.py

sudo python show_char.py

sudo python show_bmp.py

如下图所示：

```
pi@raspberrypi:~ $ cd Demo_OLED_python_IIC/source/
pi@raspberrypi:~/Demo_OLED_python_IIC/source $ sudo python show_graph.py
pi@raspberrypi:~/Demo_OLED_python_IIC/source $ sudo python show_char.py
pi@raspberrypi:~/Demo_OLED_python_IIC/source $ sudo python show_bmp.py
```

5、MSP430 使用说明

接线说明：

引脚标注见接口说明

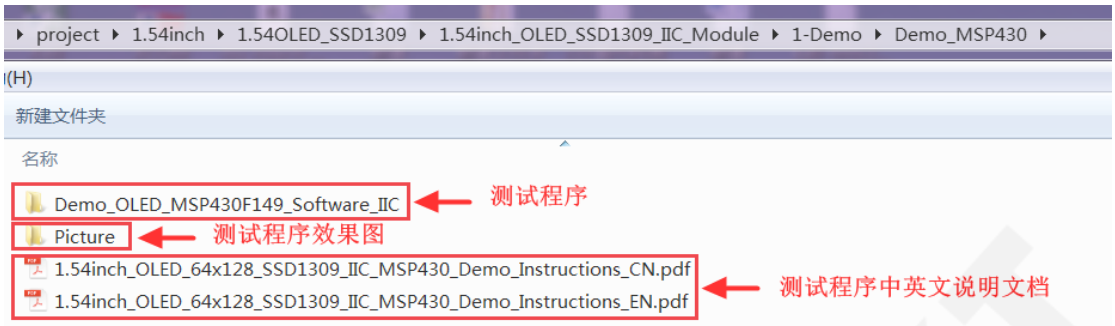
MSP430F149单片机测试程序接线说明		
序号	**引脚丝印**	**对应MSP430开发板接线引脚**
1	GND	GND
2	VCC	5V/3.3V
3	SCL	P54
4	SDA	P53

操作步骤：

A、按照上述接线说明将 OLED 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说明文档）



A、 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

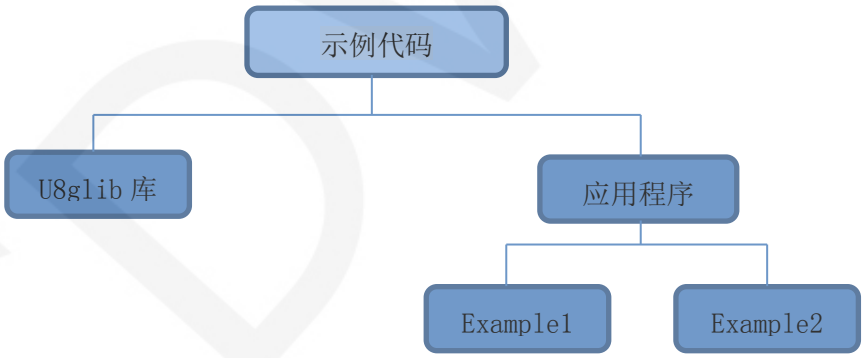
OLED 模块如果正常显示字符和图形，则说明程序运行成功；

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：



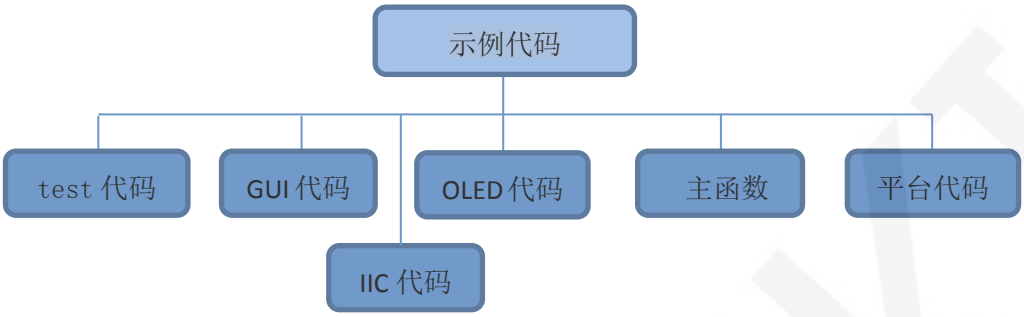
Arduino 的测试程序代码由两部分组成：U8glib 库和应用代码。

U8glib 库包含很多种控制 IC 配置，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。

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，是利用 U8glib 库提供的 API，编写一些测试示例，实现某方面的测试功能。

B、 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；

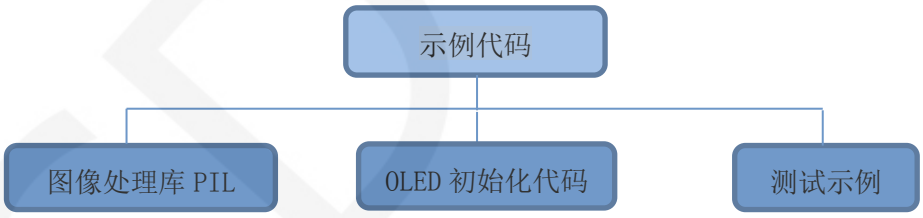
主函数实现应用程序运行；

平台代码因平台而异；

IIC 初始化及配置相关的操作包含在 IIC 代码中；

A、 RaspberryPi 代码架构说明：

python 测试程序代码架构如下图所示：



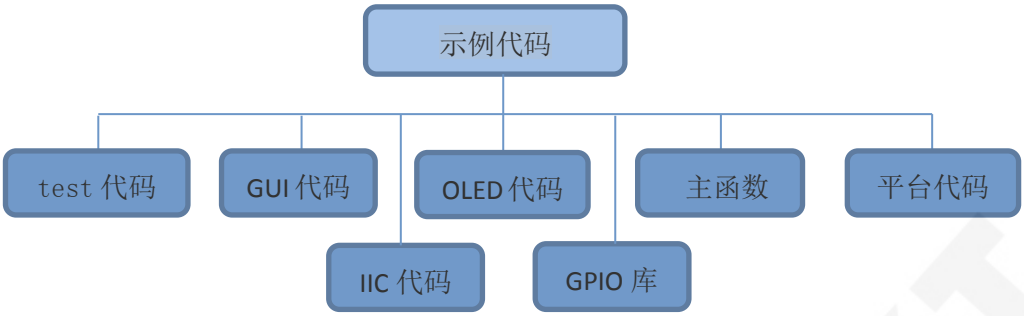
python 测试程序由但部分组成：PIL 图像处理库，OLED 初始化代码，测试示例代码

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；

IIC 初始化及配置相关的操作包含在 IIC 代码中；

2、GPIO 定义说明

A、Arduino 测试程序 GPIO 定义说明

Arduino 测试程序使用的是硬件 IIC 功能， GPIO 是固定的。

B、STM32 测试程序 GPIO 定义说明

STM32 测试程序使用的是软件模拟 IIC 功能,GPIO 定义放在 iic.h 文件里,如下图所示：

```
//---------------IIC总线引脚定义------------------------
#define OLED_SDA        GPIO_Pin_15   //OLED屏IIC数据信号
#define OLED_SCL        GPIO_Pin_13   //OLED屏IIC时钟信号
```

OLED_SDA 和 OLED_SCL 可以定义成任何空闲的 GPIO。

C、C51 测试程序 GPIO 定义说明

C51 测试程序使用的是软件模拟 IIC 功能，GPIO 定义放在 iic.h 文件里，如下图所示：

```
//---------------IIC总线引脚定义------------------------
sbit OLED_SDA = P1^5;       //OLED屏IIC数据信号 P15
sbit OLED_SCL = P1^7;       //OLED屏IIC时钟信号 P17
```

OLED_SDA 和 OLED_SCL 可以定义成任何空闲的 GPIO。

D、RaspberryPi 测试程序 GPIO 定义说明

RaspberryPi 测试程序使用的是硬件 IIC 功能，GPIO 是固定的。

E、 MSP430 测试程序 GPIO 定义说明

MSP430 测试程序使用的是软件模拟 IIC 功能，GPIO 定义放在 iic.h 文件里，如下图所示：

```
//--------------IIC总线引脚定义-----------------------
#define OLED_SDA        BIT3   //OLED屏IIC数据信号
#define OLED_SCL        BIT4   //OLED屏IIC时钟信号
```

OLED_SDA 和 OLED_SCL 可以定义成任何空闲的 GPIO。

3、 IIC 从设备地址修改

A、 Arduino 测试程序 IIC 从设备地址修改

IIC 的从设备地址定义在 u8g_com_arduino_ssd_i2c.c 文件里面，如下图所示：

```
#define I2C_SLA      (0x3c*2)
```

直接修改 I2C_SLA 即可（默认为 0x3c*2）。例如改为 0x3d*2，那么 IIC 从设备地址就是 0x3d*2

B、 STM32 和 C51 测试程序 IIC 从设备地址修改

STM32 和 C51 测试程序 IIC 的从设备地址定义在 iic.h 文件里面，如下图所示：

```
//定义IIC从设备地址
#define IIC_SLAVE_ADDR 0x78
```

直接修改 IIC_SLAVE_ADDR 即可(默认为 0x78)。例如改为 0x7A，那么 IIC 从设备地址就是 0x7A

C、 RaspberryPi 测试程序 IIC 从设备地址修改

bcm2835 和 wiringPi 测试程序 IIC 的从设备地址定义在 iic.h 文件里面，如下图所示：

```
#define IIC_SLAVE_ADDR 0x3C
```

直接修改 IIC_SLAVE_ADDR 即可（默认为 0x3C(对应 0x78)）。例如改为 0x3D，那么 IIC 从设备地址就是 0x3D(对应 0x7A)

python 测试程序 IIC 的从设备地址定义在 oled.py 文件里面，如下图所示：

```
IIC_SLAVE_ADDR = 0x3C
```

直接修改 IIC_SLAVE_ADDR 即可（默认为 0x3C(对应 0x78)）。例如改为 0x3D，那么 IIC 从设备地址就是 0x3D(对应 0x7A)

D、 MSP430 测试程序 IIC 从设备地址修改

MSP430 测试程序 IIC 的从设备地址定义在 iic.h 文件里面，如下图所示：

```
//定义IIC从设备地址
#define IIC_SLAVE_ADDR 0x78
```

直接修改 IIC_SLAVE_ADDR 即可(默认为 0x78)。例如改为 0x7A，那么 IIC 从设备地址就是 0x7A

4、IIC 通信代码实现

A、Arduino 测试程序 IIC 通信代码实现

Arduino 测试程序 IIC 通信代码由 U8glib 实现,具体实现方法可以查阅 U8glib 代码。

B、 STM32 测试程序 IIC 通信代码实现

STM32 测试程序 IIC 通信代码在 iic.c 中实现（不同的 MCU 实现方式有细微区别），如下图所示：

```
/*******************************************************************************
 * @name       :void IIC_Start(void)
 * @date       :2018-09-13
 * @function   :start iic bus
 * @parameters :None
 * @retvalue   :None
*******************************************************************************/
void IIC_Start(void)
{
  OLED_SCL_SET();
  OLED_SDA_SET();
  OLED_SDA_CLR();
  OLED_SCL_CLR();
}

/*******************************************************************************
 * @name       :void IIC_Stop(void)
 * @date       :2018-09-13
 * @function   :stop iic bus
 * @parameters :None
 * @retvalue   :None
*******************************************************************************/
void IIC_Stop(void)
{
  OLED_SCL_SET();
  OLED_SDA_CLR();
  OLED_SDA_SET();
}

/*******************************************************************************
 * @name       :void IIC_Wait_Ack(void)
 * @date       :2018-09-13
 * @function   :wait iic ack
 * @parameters :None
 * @retvalue   :None
*******************************************************************************/
void IIC_Wait_Ack(void)
{
  OLED_SCL_SET();
  OLED_SCL_CLR();
}
```

```
/*****************************************************************************
 * @name       :void Write_IIC_Byte(u8 IIC_Byte)
 * @date       :2018-09-13
 * @function   :Write a byte of content with iic bus
 * @parameters :IIC_Byte
 * @retvalue   :None
*****************************************************************************/
void Write_IIC_Byte(u8 IIC_Byte)
{
  u8 i;
  u8 m,da;
  da=IIC_Byte;
  OLED_SCL_CLR();
  for(i=0;i<8;i++)
  {
    m=da;
    m=m&0x80;
    if(m==0x80)
    {
      OLED_SDA_SET();
    }
    else
    {
      OLED_SDA_CLR();
    }
    da=da<<1;
    OLED_SCL_SET();
    OLED_SCL_CLR();
    }
}
```

```
/*****************************************************************************
 * @name       :void Write_IIC_Command(u8 IIC_Command)
 * @date       :2018-09-13
 * @function   :Write a byte of command to oled screen
 * @parameters :IIC_Command:command to be written
 * @retvalue   :None
*****************************************************************************/
void Write_IIC_Command(u8 IIC_Command)
{
  IIC_Start();
  Write_IIC_Byte(IIC_SLAVE_ADDR);            //Slave address,SA0=0
  IIC_Wait_Ack();
  Write_IIC_Byte(0x00);      //write command
  IIC_Wait_Ack();
  Write_IIC_Byte(IIC_Command);
  IIC_Wait_Ack();
  IIC_Stop();
}

/*****************************************************************************
 * @name       :void Write_IIC_Data(u8 IIC_Data)
 * @date       :2018-09-13
 * @function   :Write a byte of data to oled screen
 * @parameters :IIC_Data:data to be written
 * @retvalue   :None
*****************************************************************************/
void Write_IIC_Data(u8 IIC_Data)
{
  IIC_Start();
  Write_IIC_Byte(IIC_SLAVE_ADDR);      //D/C#=0; R/W#=0
  IIC_Wait_Ack();
  Write_IIC_Byte(0x40);      //write data
  IIC_Wait_Ack();
  Write_IIC_Byte(IIC_Data);
  IIC_Wait_Ack();
  IIC_Stop();
}
```

C、C51 测试程序 IIC 通信代码实现

C51 测试程序 IIC 通信代码在 iic.c 中实现，如下图所示：

```
/*****************************************************************************
 * @name       :void IIC_Start(void)
 * @date       :2018-09-13
 * @function   :start iic bus
 * @parameters :None
 * @retvalue   :None
*****************************************************************************/
void IIC_Start(void)
{
  OLED_SCL_SET();
  OLED_SDA_SET();
  OLED_SDA_CLR();
  OLED_SCL_CLR();
}

/*****************************************************************************
 * @name       :void IIC_Stop(void)
 * @date       :2018-09-13
 * @function   :stop iic bus
 * @parameters :None
 * @retvalue   :None
*****************************************************************************/
void IIC_Stop(void)
{
//  OLED_SCL_SET();
  OLED_SDA_CLR();
  OLED_SDA_SET();
}

/*****************************************************************************
 * @name       :void IIC_Wait_Ack(void)
 * @date       :2018-09-13
 * @function   :wait iic ack
 * @parameters :None
 * @retvalue   :None
*****************************************************************************/
void IIC_Wait_Ack(void)
{
  OLED_SCL_SET();
  OLED_SCL_CLR();
}
```

```
/*****************************************************************************
 * @name       :void Write_IIC_Byte(u8 IIC_Byte)
 * @date       :2018-09-13
 * @function   :Write a byte of content with iic bus
 * @parameters :IIC_Byte
 * @retvalue   :None
*****************************************************************************/
void Write_IIC_Byte(u8 IIC_Byte)
{
  u8 i;
  u8 m,da;
  da=IIC_Byte;
  OLED_SCL_CLR();
  for(i=0;i<8;i++)
  {
    m=da;
    m=m&0x80;
    if(m==0x80)
    {
      OLED_SDA_SET();
    }
    else
    {
      OLED_SDA_CLR();
    }
    da=da<<1;
    OLED_SCL_SET();
    OLED_SCL_CLR();
    }
}
```

```
/*****************************************************************************
 * @name       :void Write_IIC_Command(u8 IIC_Command)
 * @date       :2018-09-13
 * @function   :Write a byte of command to oled screen
 * @parameters :IIC_Command:command to be written
 * @retvalue   :None
*****************************************************************************/
void Write_IIC_Command(u8 IIC_Command)
{
  IIC_Start();
  Write_IIC_Byte(IIC_SLAVE_ADDR);            //Slave address,SA0=0
  IIC_Wait_Ack();
  Write_IIC_Byte(0x00);     //write command
  IIC_Wait_Ack();
  Write_IIC_Byte(IIC_Command);
  IIC_Wait_Ack();
  IIC_Stop();
}

/*****************************************************************************
 * @name       :void Write_IIC_Data(u8 IIC_Data)
 * @date       :2018-09-13
 * @function   :Write a byte of data to oled screen
 * @parameters :IIC_Data:data to be written
 * @retvalue   :None
*****************************************************************************/
void Write_IIC_Data(u8 IIC_Data)
{
  IIC_Start();
  Write_IIC_Byte(IIC_SLAVE_ADDR);     //D/C#=0; R/W#=0
  IIC_Wait_Ack();
  Write_IIC_Byte(0x40);     //write data
  IIC_Wait_Ack();
  Write_IIC_Byte(IIC_Data);
  IIC_Wait_Ack();
  IIC_Stop();
}
```

A、RaspberryPi 测试程序 IIC 通信代码实现

wiringPi 测试程序 IIC 通信代码在 iic.c 中实现，如下图所示：

```
51: uint32_t iic_fd;
52:
53:
54: uint32_t IIC_init(void)
55: {
56:     uint32_t fd;
57:     fd = wiringPiI2CSetup (IIC_SLAVE_ADDR);
58:     return fd;
59: }
60:
61: /*********************************************************************************
62:  * @name       :void IIC_WriteCmd(uint8_t I2C_Command)
63:  * @date       :2018-09-14
64:  * @function   :write a byte of command with iic bus
65:  * @parameters :I2C_Command:command to be writen
66:  * @retvalue   :None
67: ********************************************************************************/
68: void IIC_WriteCmd(uint8_t I2C_Command)
69: {
70:     wiringPiI2CWriteReg8(iic_fd, IIC_COMMAND, I2C_Command);
71: }
72:
73: /*********************************************************************************
74:  * @name       :void IIC_WriteDat(uint8_t I2C_Data)
75:  * @date       :2018-09-14
76:  * @function   :write a byte of data with iic bus
77:  * @parameters :I2C_Data:data to be writen
78:  * @retvalue   :None
79: ********************************************************************************/
80: void IIC_WriteDat(uint8_t I2C_Data)
81: {
82:     wiringPiI2CWriteReg8(iic_fd, IIC_DATA, I2C_Data);
83: }
84:
```

首先调用 IIC_init 进行初始化，设置 IIC 从设备地址，获取 IIC 设备文件描述符，然后使用 IIC 设备文件描述符分别写入寄存器命令和内存数据。

bcm2835 测试程序 IIC 通信代码在 iic.c 中实现，如下图所示：

```
58: void IIC_WriteCmd(uint8_t I2C_Command)
59: {
60:     char buf[2] = {0};
61:     int ref;
62:     buf[0] = IIC_COMMAND;
63:     buf[1] = I2C_Command;
64:     ref = bcm2835_i2c_write(buf, 2);
65:     while(ref != 0)
66:     {
67:         ref = bcm2835_i2c_write(buf, 2);
68:         if(ref == 0)
69:         {
70:             break;
71:         }
72:     }
73: }
74: /*********************************************************************************
75:  * @name       :void IIC_WriteDat(uint8_t I2C_Data)
76:  * @date       :2018-09-14
77:  * @function   :write a byte of data with iic bus
78:  * @parameters :I2C_Data:data to be writen
79:  * @retvalue   :None
80: ********************************************************************************/
81: void IIC_WriteDat(uint8_t I2C_Data)
82: {
83:     char buf[2] = {0};
84:     int ref;
85:     buf[0] = IIC_DATA;
86:     buf[1] = I2C_Data;
87:     ref = bcm2835_i2c_write(buf, 2);
88:     while(ref != 0)
89:     {
90:         ref = bcm2835_i2c_write(buf, 2);
91:         if(ref == 0)
92:         {
93:             break;
94:         }
95:     }
96: }
97: /*********************************************************************************
98:  * @name       :void IIC_init(void)
99:  * @date       :2018-09-14
100:  * @function   :initialise iic bus
101:  * @parameters :None
102:  * @retvalue   :None
103: ********************************************************************************/
104: void IIC_init(void)
105: {
106:     bcm2835_i2c_begin();
107:     bcm2835_i2c_setSlaveAddress(IIC_SLAVE_ADDR);    //7 bits i2c address
108:     bcm2835_i2c_set_baudrate(2000000);    //1M I2C rate
109: }
```

首先调用 IIC_init 进行初始化，设置 IIC 从设备地址，设置 IIC 传输速率，然后调用函数分别写入寄存器命令和内存数据。

python 的测试程序 IIC 通信代码在 oled.py 中实现，如下图所示：

```
        self.oledsmbus = smbus
    def iic_command(self,val):
        self.oledsmbus.write_byte_data(IIC_SLAVE_ADDR,COMMAND_MODE,val)
    def iic_data(self,val):
        self.oledsmbus.write_byte_data(IIC_SLAVE_ADDR,DATA_MODE,val)
```

首先调用 SMBus 进行初始化，然后调用 write_byte_data 函数分别写入寄存器命令和内存数据。

B、MSP430 测试程序 IIC 通信代码实现

MSP430 测试程序 IIC 通信代码在 iic.c 中实现，如下图所示：

```
/*****************************************************************************
 * @name       :void IIC_Start(void)
 * @date       :2018-09-13
 * @function   :start iic bus
 * @parameters :None
 * @retvalue   :None
*****************************************************************************/
void IIC_Start(void)
{
        OLED_SCL_SET();
        OLED_SDA_SET();
        OLED_SDA_CLR();
        OLED_SCL_CLR();
}

/*****************************************************************************
 * @name       :void IIC_Stop(void)
 * @date       :2018-09-13
 * @function   :stop iic bus
 * @parameters :None
 * @retvalue   :None
*****************************************************************************/
void IIC_Stop(void)
{
        OLED_SCL_SET();
        OLED_SDA_CLR();
        OLED_SDA_SET();
}

/*****************************************************************************
 * @name       :void IIC_Wait_Ack(void)
 * @date       :2018-09-13
 * @function   :wait iic ack
 * @parameters :None
 * @retvalue   :None
*****************************************************************************/
void IIC_Wait_Ack(void)
{
        OLED_SCL_SET();
        OLED_SCL_CLR();
}
```

```
/**********************************************************************************
 * @name       :void Write_IIC_Byte(u8 IIC_Byte)
 * @date       :2018-09-13
 * @function   :Write a byte of content with iic bus
 * @parameters :IIC_Byte
 * @retvalue   :None
**********************************************************************************/
void Write_IIC_Byte(u8 IIC_Byte)
{
        u8 i;
        u8 m,da;
        da=IIC_Byte;
        OLED_SCL_CLR();
        for(i=0;i<8;i++)
        {
                m=da;
                m=m&0x80;
                if(m==0x80)
                {
                        OLED_SDA_SET();
                }
                else
                {
                        OLED_SDA_CLR();
                }
                da=da<<1;
                OLED_SCL_SET();
                OLED_SCL_CLR();
                }
}
```

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