1.5inch IIC OLED Module
MC01506
User Manual

Introduction to OLED

OLED is an Organic Light-Emitting Diode (OLED). OLED display technology has the advantages of self-illumination, wide viewing angle, almost infinite contrast, low power consumption, high reaction speed, flexible panel, wide temperature range, simple structure and process, etc. A generation of flat panel display emerging application technology.

OLED display is different from traditional LCD display, it can self-illuminate, so no backlight is needed, which makes OLED display

The display is thinner than the LCD display and has a better display.

Product Description

The OLED module has a display size of 1.5" and has a 128x128 resolution for black and white or black and blue. It adopts IIC communication mode and the internal driver IC is SH1107.

Product Features

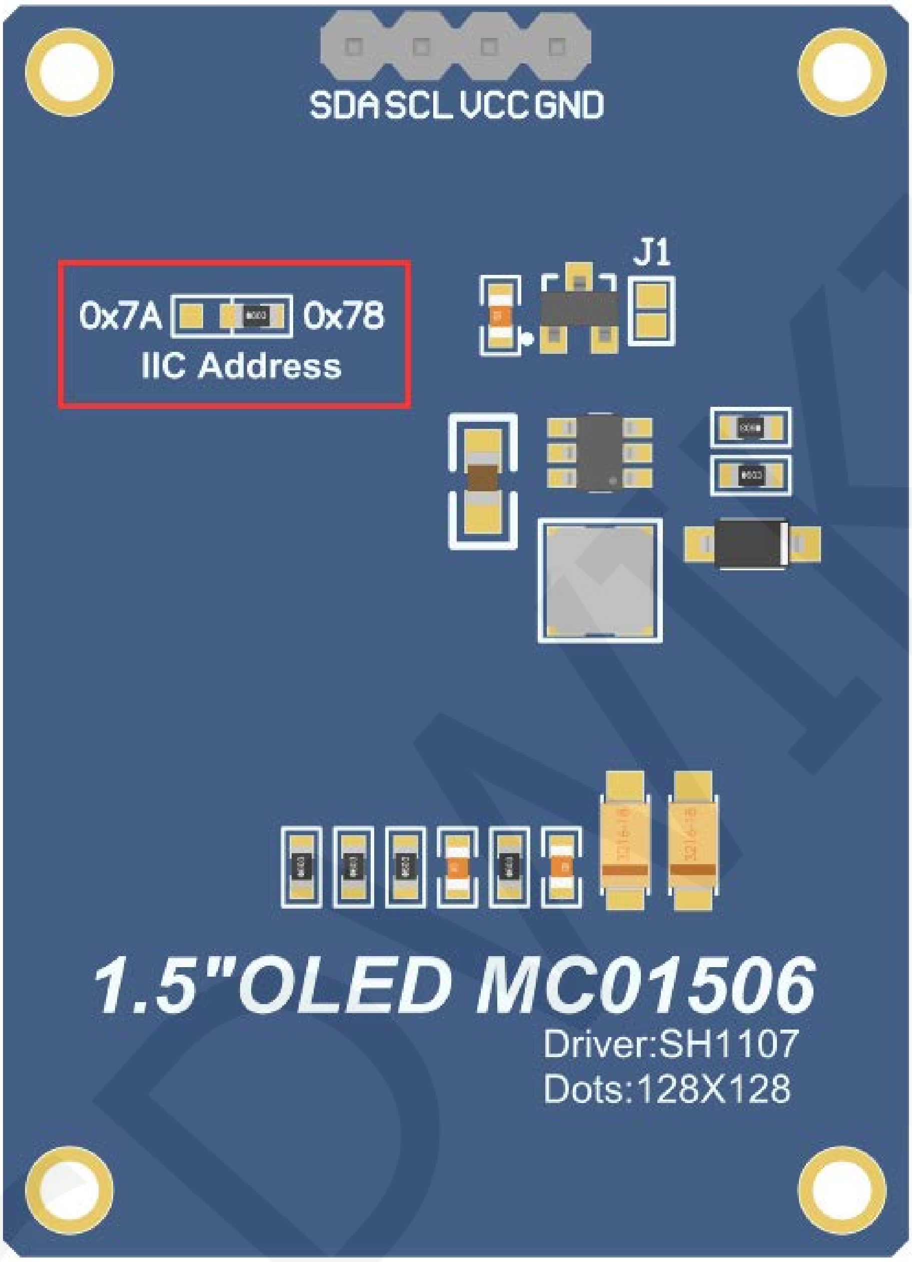
- 1.5 inch OLED screen with black and white or black and blue color display
- 128x128 resolution for clear display and high contrast
- Large viewing angle: greater than 160° (one screen with the largest viewing angle in the display)
- Wide voltage supply (3V~5V), compatible with 3.3V and 5V logic levels, no level shifting chip required
- With IIC bus, only a few IOs can be used to light up the display
- Ultra-low power consumption: normal display is only 0.06W (far below the TFT display)
- Military-grade process standards, long-term stable work
- Provides a rich sample program for STM32, C51, Arduino, Raspberry Pi platforms
- Provide underlying driver technical support

Product Parameters

Name	Description
Display Color	Black white / black blue
SKU	MC01506
Screen Size	1.5(inch)
Type	OLED
Driver IC	SH1107
Resolution	128*128(Pixel)
Module Interface	IIC interface
Active Area	26.86x26.86(mm)
Touch Screen type	No touch screen
Touch IC	No touch IC
Module PCB Size	45.50x34.30(mm)
Visual angle	>160°
Operating Temperature	-10℃~60℃
Storage Temperature	-10℃~70℃
Operating Voltage	3.3V / 5V
Power Consumption	TDB
Product Weight(With packaging)	15(g)

Interface Description

Picture 1. Module pin silk screen



Picture 2. Rear view of the module

NOTE:

1. **This module supports IIC slave device address switching (shown in red box in Picture 4), as follows:**
   A. **Solder the 0x78 side resistance, disconnect the 0x7A side, then select the 0x78 slave address (default);**
   B. **Solder the 0x7A side resistance, disconnect the 0x78 side, then select the**

0x7A slave address;

2. **The hardware switches the IIC from the set address, and the software also needs to be modified accordingly. For the specific modification method, see the following IIC slave device address modification instructions.**

Number	Module Pin	Pin description
1	**GND**	OLED power ground
2	**VCC**	OLED power positive (3.3V~5V)
3	**SCL**	OLED IIC bus clock signal
4	**SDA**	OLED IIC bus data signal

Hardware Configuration

Since the OLED can self-illuminate, the OLED module has no backlight control circuit and only the OLED display control circuit and the IIC slave device address selection control circuit (as shown in the red box of Figure 3).

The OLED display control circuit is mainly used to control OLED display, including chip selection, reset, and data and command transmission control.

The IIC slave device address selection control circuit is used to select different slave device addresses.

DC-DC boost circuit is used to provide stable power supply.

The OLED module adopts IIC communication mode, and the hardware is configured with two pins: SCL (IIC data pin) and SDA (IIC clock pin). The IIC data transfer can be completed by controlling the two pins according to the IIC working timing.

working principle

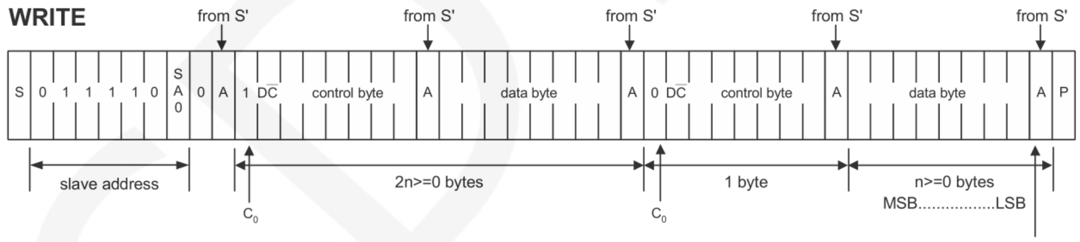
1. Introduction to SH1107 Controller

The SH1107 is an OLED/PLED controller that supports a maximum resolution of 128*128 and a 2048-byte GRAM. Support 8-bit 6800 and 8-bit 8080 parallel port data bus, also supports 3-wire and 4-wire SPI serial bus and I2C bus. Since parallel control requires a large number of IO ports, the most commonly used are the SPI serial bus and the I2C bus. It supports vertical scrolling and can be used in small portable devices such as mobile phones, MP3 players and more.

The SH1107 controller uses 1 bit to control a pixel display, so each pixel can only display black and white or black and blue. The displayed RAM is divided into 16 pages, with 8 lines per page and 128 pixels per line. When setting pixel data, you need to specify the page address first, and then specify the column low address and column height address respectively, so set 8 pixels in the vertical direction at the same time. In order to be able to flexibly control the pixel points at any position, the software first sets a global one-dimensional array of the same size as the display RAM, first maps the pixel point data to the global array, and the process uses the OR or the operation to ensure that the global array is written before. The data is not corrupted, and the data of the global array is then written to the GRAM so that it can be displayed through the OLED.

2. Introduction to IIC Communication Protocol

The process of writing data on the IIC bus is shown in the following figure:



After the IIC bus starts working, the slave device address is sent first. After receiving the slave device response, it then sends a control byte to inform the slave device whether the next data to be sent is a command written to the IC register or written. The RAM data, after receiving the slave device response, then sends a value of multiple bytes until the transmission is completed and the IIC bus stops working.

among them:

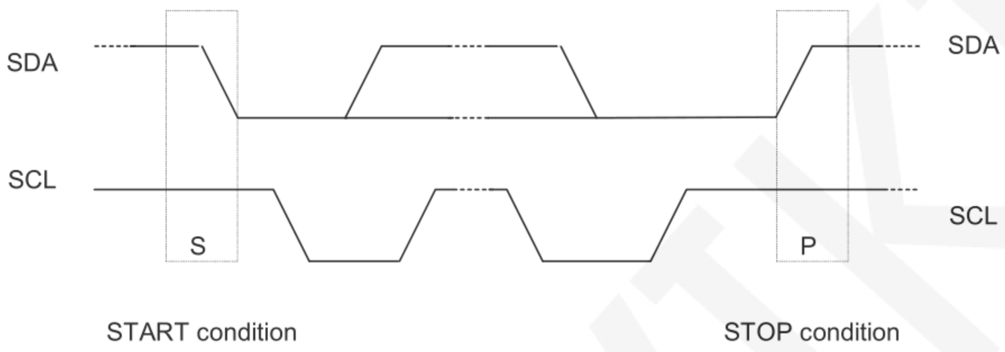
C0=0: This is the last control byte, and all the data bytes sent in the following are all data bytes.

C0=1: The next two bytes to be sent are the data byte and another control byte.

D/C(——)=0: is the register command operation byte

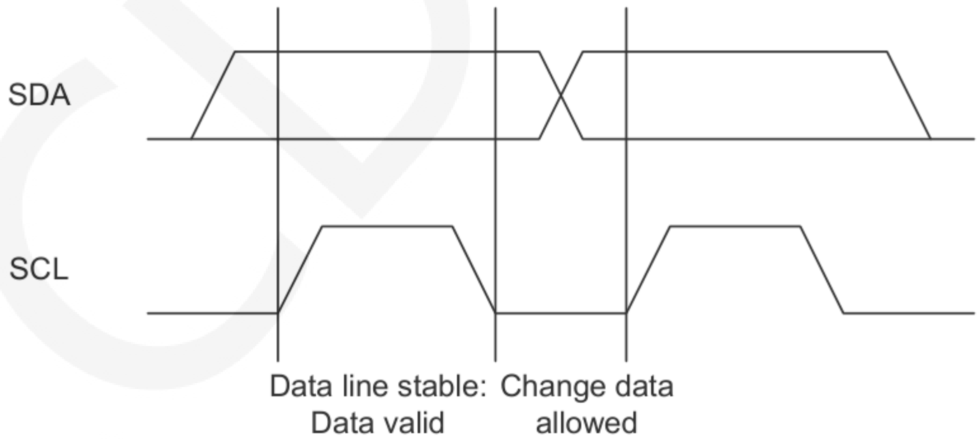
D/C(——)=1: operation byte for RAM data

The IIC start and stop timing diagrams are as follows:



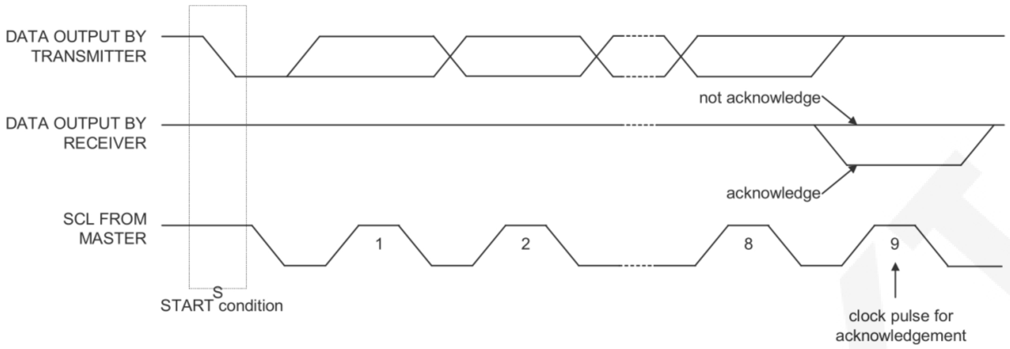
When the data line and the clock line of the IIC are both kept at a high level, the IIC is in an idle state. At this time, the data line changes from a high level to a low level, and the clock line continues to be at a high level, and the IIC bus starts data transmission. When the clock line is held high, the data line changes from low to high, and the IIC bus stops data transmission.

The timing diagram for the IIC to send a bit of data is as follows:



Each clock pulse (the process of pulling high and pulling low) sends 1 bit of data. When the clock line is high, the data line must remain stable, and the data line is allowed to change when the clock line is low.

The ACK transmission timing diagram is as follows:



When the master waits for the ACK of the slave, it needs to keep the clock line high. When the slave sends an ACK, keep the data line low.

Instructions for use

1. Arduino instructions

Wiring instructions:

See the interface description for pin assignments.

Arduino UNO microcontroller test program wiring instructions		
Number	**Module Pin**	**Corresponding to UNO development board wiring pins**
1	**GND**	GND
2	**VCC**	5V/3.3V
3	**SCL**	A5
4	**SDA**	A4

Arduino MEGA2560 microcontroller test program wiring instructions		
Number	**Module Pin**	**Corresponding to MEGA2560 development board wiring pins**
1	**GND**	GND
2	**VCC**	5V/3.3V
3	**SCL**	21

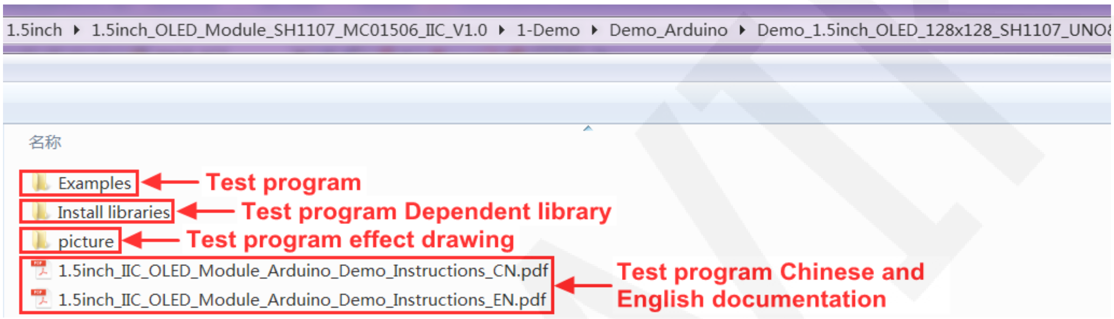
4	SDA	20

Operating Steps：

A. Connect the OLED module and the Arduino MCU according to the above wiring instructions, and power on;

B. Select the example you want to test, as shown below:

(Please refer to the test program description document for test program description)



C. Open the selected sample project, compile and download.

The specific operation methods for the Arduino test program relying on library copy, compile and download are as follows:

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

D. If the OLED module displays characters and graphics normally, the program runs Successfully;

2. RaspberryPi instructions

Wiring instructions:

See the interface description for pin assignments.

NOTE:

Physical pin refers to the GPIO pin code of the RaspBerry Pi development board.

BCM encoding refers to the GPIO pin coding when using the BCM2835 GPIO library.

WiringPi coding refers to the GPIO pin coding when using the wiringPi GPIO library.

Which GPIO library is used in the code, the pin definition needs to use the corresponding GPIO library code, see Picture 1 GPIO map table for details.

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

Picture4. GPIO map

Raspberry Pi test program wiring instructions		
Number	**Module Pin**	**Corresponding to development board wiring pin**
1	GND	GND (Physical pin：6,9,14,20,25,30,34,39)
2	VCC	5V/3.3V （Physical pin：1,2,4）

3	**SCL**	**Physical pin： 5** **BCM coding： 3** **wiringPi coding： 9**
4	**SDA**	**Physical pin： 3** **BCM coding： 2** **wiringPi coding： 8**

Operating Steps：

A. open the IIC function of RaspberryPi

Log in to the RaspberryPi using a serial terminal tool (such as putty) and enter the following command:

sudo raspi-config

Select Interfacing Options->I2C->YES

Start RaspberryPi's I2C kernel driver

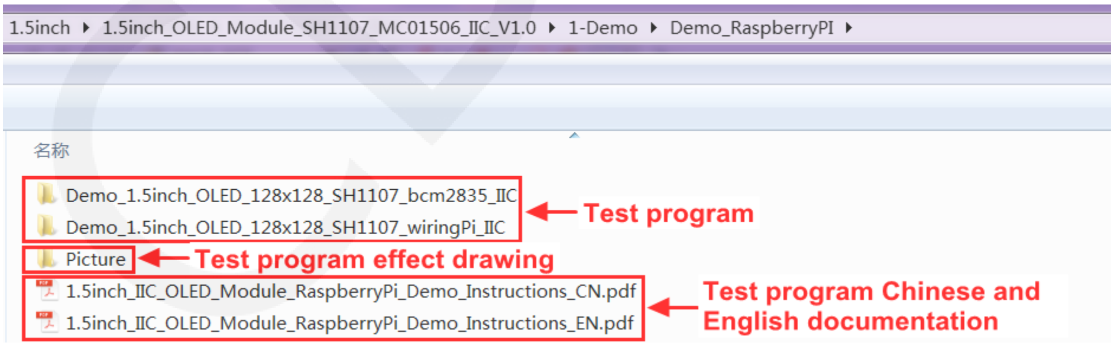
B. install the function library

For detailed installation methods of the bcm2835, wiringPi function libraries of RaspberryPi, see the following documents:

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

C. select the example that needs to be tested, as shown below:

(Please refer to the test program description document for test program description)



D. bcm2835 instructions

a) Connect the OLED module to the RaspberryPi development board according to the above wiring

b) Copy the test program directory

Demo_1.5inch_OLED_128x128_SH1107_bcm2835_IIC to RaspberryPi (can be copied via SD card or via FTP tool (such as FileZilla))

c) Run the following command to run the bcm2835 test program:

cd Demo_1.5inch_OLED_128x128_SH1107_bcm2835_IIC

make

sudo ./ 1.5_IIC_OLED

As shown below:

```
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_b
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 instructions

a) Connect the OLED module to the RaspberryPi development board according to the above wiring

b) Copy the test program directory Demo_1.5inch_OLED_128x128_SH1107_wiringPi_IIC to RaspberryPi (can be copied via SD card or via FTP tool (such as FileZilla))

c) Run the following command to run the wiringPi test program:

cd Demo_1.5inch_OLED_128x128_SH1107_wiringPi_IIC

make

sudo ./ 1.5_IIC_OLED

As shown below:

```
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/cs
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/cs
gcc -g -O0 -c  /home/pi/csl/Demo_OLED_wiringPi_IIC/source/src/delay.c -o /home/pi/c
gcc -g -O0 -c  /home/pi/csl/Demo_OLED_wiringPi_IIC/source/src/oled.c -o /home/pi/cs
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/cs
pi@raspberrypi:~/csl/Demo_OLED_wiringPi_IIC $ sudo ./1.3_IIC_OLED
```

If you want to modify the IIC transfer rate, you need to add the following content to the /boot/config.txt file, then restart raspberryPi

, i2c_arm_baudrate=2000000 (note that the comma is also required)

As shown below (the red box is the added content, the number 2000000 is the set rate, can be changed):

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

3. STM32 instructions

Wiring instructions:

See the interface description for pin assignments.

STM32F103C8T6 microcontroller test program wiring instructions

Number	Module Pin	Corresponding to the F103C8T6 development board wiring pin
1	GND	GND
2	VCC	5V/3.3V
3	SCL	PA5
4	SDA	PA7

STM32F103RCT6 microcontroller test program wiring instructions

Number	Module Pin	Corresponding to the MiniSTM32 development board wiring pin
1	GND	GND
2	VCC	5V/3.3V
3	SCL	PB13
4	SDA	PB15

STM32F103ZET6 microcontroller test program wiring instructions		
Number	Module Pin	Corresponding to the Elite STM32 development board wiring pin
1	GND	GND
2	VCC	5V/3.3V
3	SCL	PB13
4	SDA	PB15

STM32F407ZGT6 microcontroller test program wiring instructions		
Number	Module Pin	Corresponding to the Explorer STM32F4 development board wiring pin
1	GND	GND
2	VCC	5V/3.3V
3	SCL	PB3
4	SDA	PB5

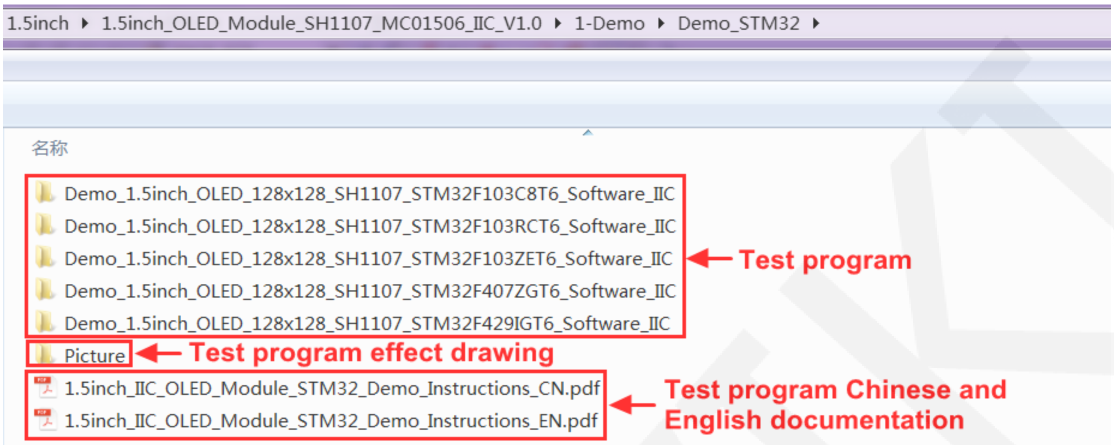
STM32F429IGT6 microcontroller test program wiring instructions		
Number	Module Pin	Corresponding to the Apollo STM32F4/F7 development board wiring pin
1	GND	GND
2	VCC	5V/3.3V
3	SCL	PF7
4	SDA	PF9

Operating Steps:

A. Connect the LCD module and the STM32 MCU according to the above wiring instructions, and power on;

B. Open the directory where the STM32 test program is located and select the example to be tested, as shown below:

(Please refer to the test program description document for test program description)



C. Open the selected test program project, compile and download;

detailed description of the STM32 test program compilation and download can be found in the following document:

http://www.lcdwiki.com/res/PublicFile/STM32_Keil_Use_Illustration_EN.pdf

D. If the OLED module displays characters and graphics normally, the program runs successfully;

4. C51 instructions

Wiring instructions:

See the interface description for pin assignments.

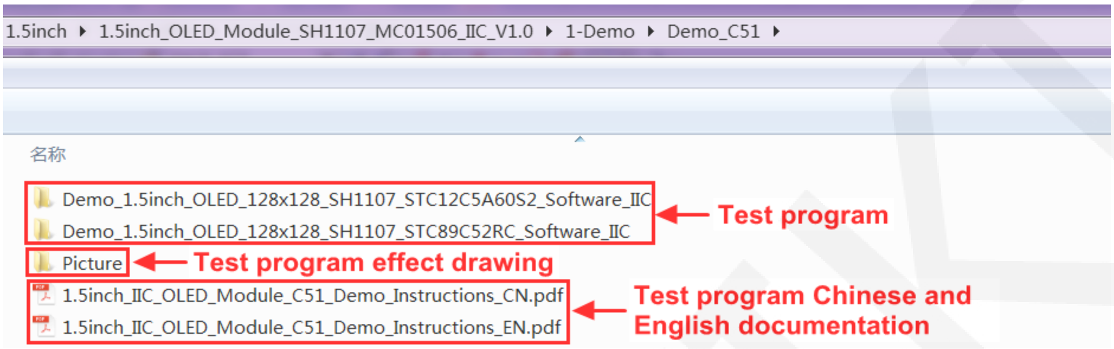
STC89C52RC and STC12C5A60S2 microcontroller test program wiring instructions		
Number	**Module Pin**	**Corresponding to STC89/STC12 development board wiring pin**
1	**GND**	GND
2	**VCC**	5V/3.3V
3	**SCL**	P17
4	**SDA**	P15

Operating Steps:

A. Connect the LCD module and the C51 MCU according to the above wiring instructions, and power on;

B. Open the directory where the C51 test program is located and select the example to be tested, as shown below:

(Please refer to the test program description document for test program description)



C. Open the selected test program project, compile and download;

detailed description of the C51 test program compilation and download can be found in the following document:

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

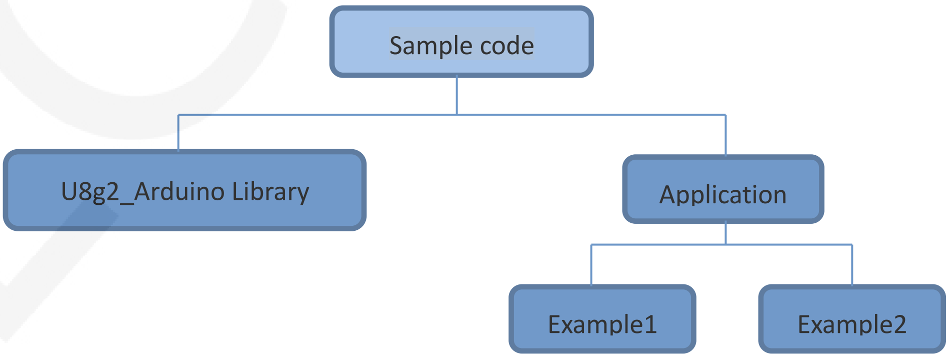
D. If the OLED module displays characters and graphics normally, the program runs successfully;

Software Description

1. Code Architecture

A. Arduino code architecture description

The code architecture is shown below



Arduino's test program code consists of two parts: the U8g2_Arduino library and application code.

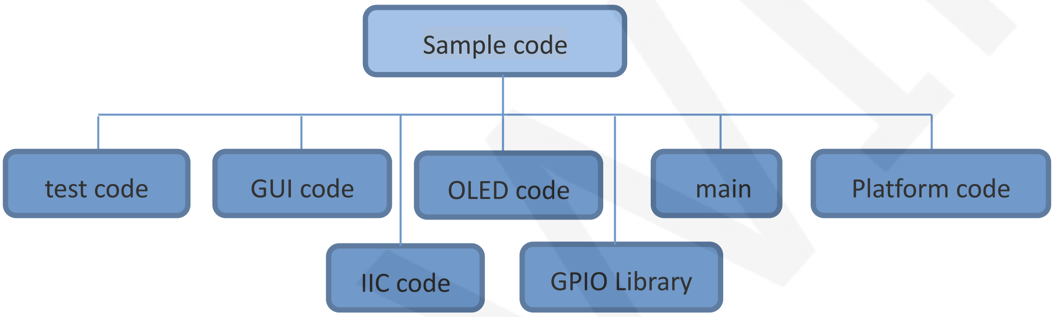
The U8g2_Arduino library contains a variety of control IC configurations, mainly responsible

for operating registers, including hardware module initialization, data and command transfer, pixel coordinates and color settings, display mode configuration, etc.

The application contains several test examples, each of which contains different test content. It uses the API provided by the U8glib library, writes some test examples, and implements some aspects of the test function.

B. RaspberryPi code architecture description

The bcm2835 and wiringPi test program code architecture is as follows:



The Demo API code for the main program runtime is included in the test code;

OLED initialization and related operations are included in the OLED code;

Drawing points, lines, graphics, and Chinese and English character display related operations are included in the GUI code;

The GPIO library provides GPIO operations;

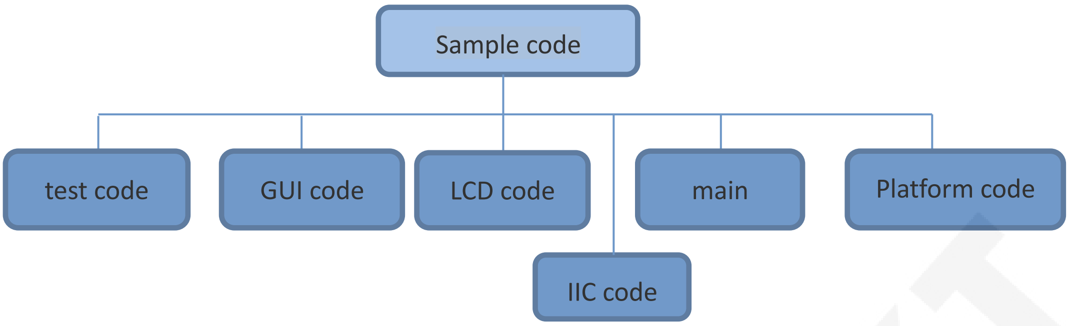
The main function implements the application to run;

Platform code varies by platform;

IIC initialization and configuration related operations are included in the IIC code;

C. C51, STM32 code architecture description

The code architecture is shown below:



The Demo API code for the main program runtime is included in the test code;

OLED initialization and related bin parallel port write data operations are included in the OLED code;

Drawing points, lines, graphics, and Chinese and English character display related operations are included in the GUI code;

The main function implements the application to run;

Platform code varies by platform;

IIC initialization and configuration related operations are included in the IIC code;

2. GPIO definition description

A. Arduino test program GPIO definition description

The Arduino test program uses the hardware IIC function, and the GPIO is fixed.

B. RaspberryPi test program GPIO definition description

The RaspberryPi test program uses the hardware IIC function, and the GPIO is fixed.

C. STM32 test program GPIO definition description

The STM32 test program uses the software simulation IIC function, and the GPIO definition is placed in the iic.h file, as shown in the following figure:

```
//--------------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 and OLED_SCL can be defined as any idle GPIO.

D. C51 test program GPIO definition description

The C51 test program uses the software simulation IIC function, and the GPIO definition is placed in the iic.h file, as shown in the following figure:

```
//--------------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 and OLED_SCL can be defined as any idle GPIO.

3. IIC slave device address modification

A. Arduino test program IIC modified from device address

Use the **setI2CAddress** function to modify the I2C slave device address as follows:

Open the test program, find the setup function, and add the **setI2CAddress** function before the begin function, as shown in the following figure:

```
void setup(void) {
  u8g2.setI2CAddress(0x3D*2);
  u8g2.begin();

}
```

The above operation is to set the IIC slave device address to 0x3d * 2 (0x3c * 2 by default).

B. RaspberryPi test program IIC modified from device address

The slave address of bcm2835 and wiringPi test program IIC is defined in the iic.h file, as shown in the following figure:

```
#define IIC_SLAVE_ADDR 0x3C
```

Directly modify IIC_SLAVE_ADDR(default is 0x3C (corresponding to 0x78)).

For example, change to 0x3D, then the IIC slave address is 0x3D (corresponding to 0x7A);

C. STM32 and C51 test program IIC modified from device address

The slave device address of the STM32 and C51 test program IIC is defined in the iic.h file, as shown in the following figure:

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

Directly modify IIC_SLAVE_ADDR (default is 0x78).For example, change to 0x7A, then the IIC slave address is 0x7A.

4. IIC communication code implementation

A. RaspberryPi test program IIC communication code implementation

wiringPi test program IIC communication code is implemented in iic.c, as shown below:

```
uint32_t iic_fd;


uint32_t IIC_init(void)
{
    uint32_t fd;
    fd = wiringPiI2CSetup (IIC_SLAVE_ADDR);
    return fd;
}

/*****************************************************************************
 * @name       :void IIC_WriteCmd(uint8_t I2C_Command)
 * @date       :2018-09-14
 * @function   :write a byte of command with iic bus
 * @parameters :I2C_Command:command to be writen
 * @retvalue   :None
******************************************************************************/
void IIC_WriteCmd(uint8_t I2C_Command)
{
    wiringPiI2CWriteReg8(iic_fd, IIC_COMMAND, I2C_Command);
}

/*****************************************************************************
 * @name       :void IIC_WriteDat(uint8_t I2C_Data)
 * @date       :2018-09-14
 * @function   :write a byte of data with iic bus
 * @parameters :I2C_Data:data to be writen
 * @retvalue   :None
******************************************************************************/
void IIC_WriteDat(uint8_t I2C_Data)
{
    wiringPiI2CWriteReg8(iic_fd, IIC_DATA, I2C_Data);
}
```

First call IIC_init to initialize, set the IIC slave address, get the IIC device file descriptor, and then use the IIC device file descriptor to write the register command and memory data respectively.

The bcm2835 test program IIC communication code is implemented in iic.c, as shown below:

```
void IIC_WriteCmd(uint8_t I2C_Command)
{
    char buf[2] = {0};
    int ref;
    buf[0] = IIC_COMMAND;
    buf[1] = I2C_Command;
    ref = bcm2835_i2c_write(buf, 2);
    while(ref != 0)
    {
        ref = bcm2835_i2c_write(buf, 2);
        if(ref == 0)
        {
            break;
        }
    }
}
/*****************************************************************************
 * @name       :void IIC_WriteDat(uint8_t I2C_Data)
 * @date       :2018-09-14
 * @function   :write a byte of data with iic bus
 * @parameters :I2C_Data:data to be writen
 * @retvalue   :None
******************************************************************************/
void IIC_WriteDat(uint8_t I2C_Data)
{
    char buf[2] = {0};
    int ref;
    buf[0] = IIC_DATA;
    buf[1] = I2C_Data;
    ref = bcm2835_i2c_write(buf, 2);
    while(ref != 0)
    {
        ref = bcm2835_i2c_write(buf, 2);
        if(ref == 0)
        {
            break;
        }
    }
}
/*****************************************************************************
 * @name       :void IIC_init(void)
 * @date       :2018-09-14
 * @function   :initialise iic bus
 * @parameters :None
 * @retvalue   :None
******************************************************************************/
void IIC_init(void)
{
    bcm2835_i2c_begin();
    bcm2835_i2c_setSlaveAddress(IIC_SLAVE_ADDR);    //7 bits i2c address
    bcm2835_i2c_set_baudrate(2000000);    //1M I2C rate
}
```

First call IIC_init to initialize, set the IIC slave address, get the IIC device file descriptor, and then use the IIC device file descriptor to write the register command and memory data respectively.

Arduino test program IIC communication code implementation

Arduino test program IIC communication code is implemented by U8glib, the specific implementation method can refer to U8glib code

B. STM32 test program IIC communication code implementation

The STM32 test program IIC communication code is implemented in iic.c (there are subtle differences between different MCU implementations), as shown in the following figure:

```
/*****************************************************************************
 * @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();
}
```

```c
/*****************************************************************************
 * @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();
    }
}
```

```c
/*****************************************************************************
 * @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 test program IIC communication code implementation

C51 test program IIC communication code is implemented in iic.c, as shown below:

```
/*****************************************************************************
 * @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();
}
```

Common software

This set of test examples needs to display Chinese and English, symbols and pictures, so PCtoLCD2002 modulo software is used. Here, the setting of the modulo software is explained only for the test program.The **PCtoLCD2002** modulo software settings are as follows:

Dot matrix format select Dark code

the modulo mode select the progressive mode(C51 test program needs to choose determinant)

Take the model to choose the direction (high position first) (C51 test program needs to choose reverse (low position first))

Output number system selects hexadecimal number

Custom format selection C51 format

The specific setting method is as follows:

http://www.lcdwiki.com/Chinese_and_English_display_modulo_settings