



# LCDWIKI MC130VX IIC OLED Module User Manual

[Home](#) » [LCDWIKI](#) » LCDWIKI MC130VX IIC OLED Module User Manual 

## Contents

- [1 LCDWIKI MC130VX IIC OLED Module](#)
- [2 Product Information](#)
- [3 Product Features](#)
- [4 Product Parameters](#)
- [5 Interface Description](#)
- [6 working principle](#)
- [7 Instructions for use](#)
- [8 STM32 instructions](#)
- [9 C51 instructions](#)
- [10 Software Description](#)
- [11 FAQ](#)
- [12 Documents / Resources](#)
  - [12.1 References](#)

# LCD wiki

LCDWIKI MC130VX IIC OLED Module



## Product Information

### Specifications

- **Name:** OLED Module MC01506
- **Display Color:** Black white / black blue
- **Screen Size:** 1.5 inch
- **Type:** OLED
- **Driver IC:** SH1107
- **Resolution:** 128×128
- **Module Interface:** IIC
- **Active Area:** TDB
- **Touch Screen Type:** Not available
- **Touch IC:** Not available
- **Module PCB Size:** 15(g)
- **Visual Angle:** Not specified
- **Operating Temperature:** Not specified
- **Storage Temperature:** Not specified
- **Operating Voltage:** Wide voltage supply (3V~5V), compatible with 3.3V and 5V logic levels, no level shifting chip required
- **Power Consumption:** Ultra-low power consumption, normal display is only 0.06W
- **Product Weight (With packaging):** Not specified

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

### Product Description

The OLED module is a 1.5 inch display with a resolution of 128×128. It uses IIC communication mode and the internal driver IC is SH1107. The OLED display technology provides self-illumination, wide viewing angle, almost

infinite contrast, low power consumption, high reaction speed, flexible panel, wide temperature range, simple structure, and process.

## **Product Features**

- 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
- 1.5 inch OLED screen with black and white or black and blue color display
- 128×128 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

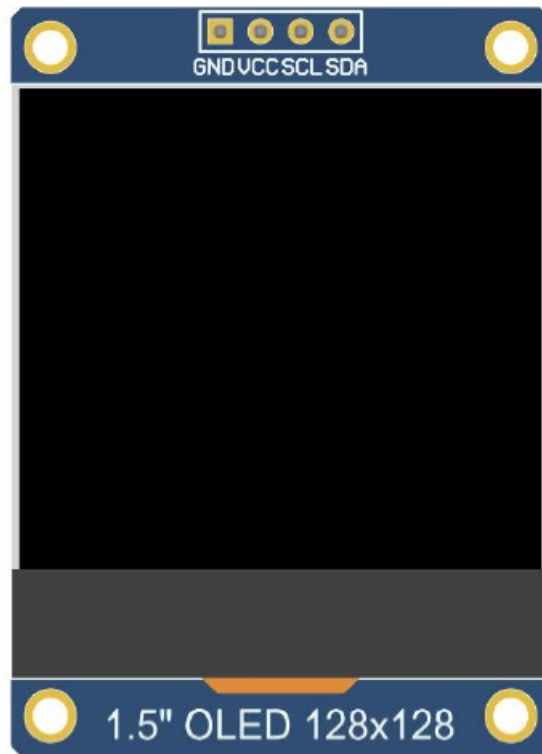
## **Product Parameters**

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	26.86×26.86(mm)
Touch Screen type	No touch screen
Touch IC	No touch IC
Module PCB Size	45.50×34.30(mm)
Visual angle	>160°
Operating Temperature	-10°C~60°C
Storage Temperature	-10°C~70°C
Operating Voltage	3.3V / 5V
Power Consumption	TDB
Product Weight(With packaging)	15(g)

## Interface Description

The module has four pins for the interface:

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



Module pin silk screen



Rear view of the module

**NOTE:**

1. This module supports IIC slave device address switching (shown in red box in Picture 4), as follows:
  - Solder the 0x78 side resistance, disconnect the 0x7A side, then select the 0x78 slave address (default);
  - 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	<b>GND</b>	OLED power ground
2	<b>VCC</b>	OLED power positive (3.3V~5V)
3	<b>SCL</b>	OLED IIC bus clock signal
4	<b>SDA</b>	OLED IIC bus data signal

### Hardware Configuration

The OLED module does not have a backlight control circuit. It only has the OLED display control circuit and the IIC slave device address selection control circuit. 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.

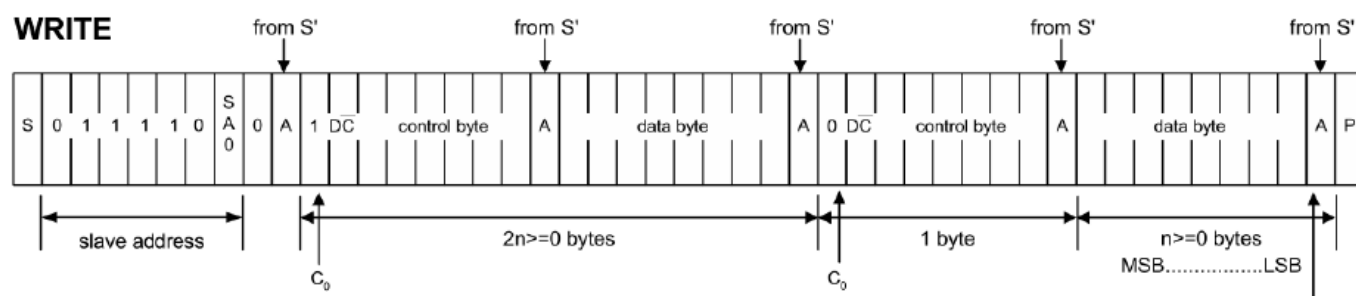
### working principle

#### 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.

#### Introduction to IIC Communication Protocol

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



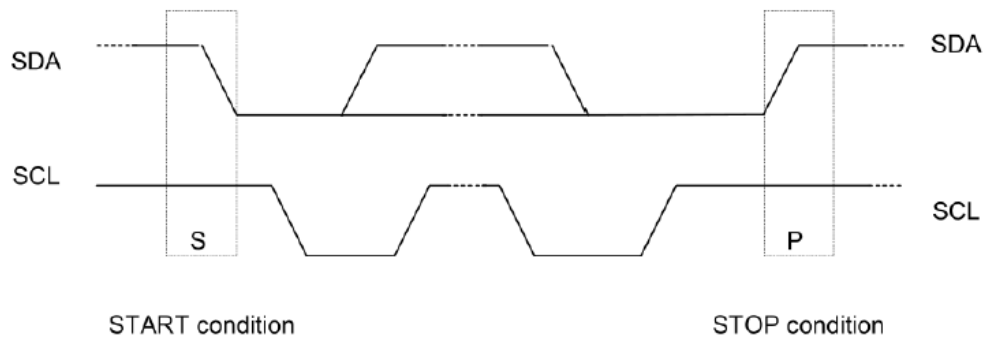
After the IIC bus starts working, the slave device address is sent first. After receiving the slave device response, it then sends a control byte to inform the slave device whether the next data to be sent is a command written to the IC register or written. The RAM data, after receiving the slave device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

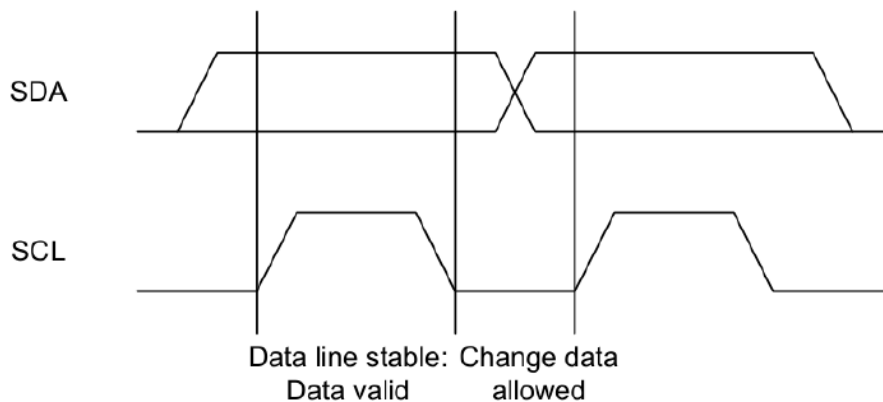
- C0=1: The next two bytes to be sent are the data byte and another control byte.
- D/C(—)=0: is the register command operation byte
- D/C(—)=1: operation byte for RAM data

**The IIC start and stop timing diagrams are as follows:**



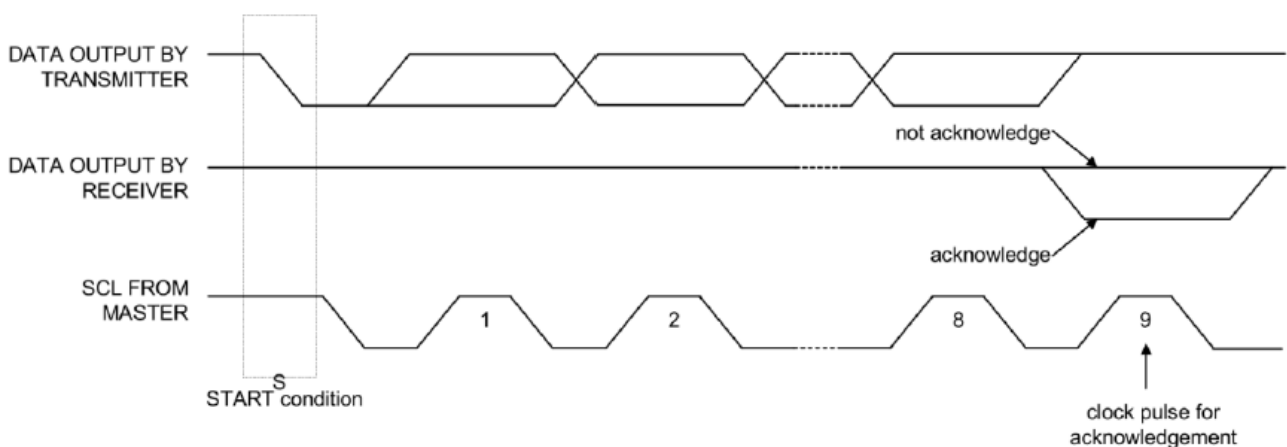
When the data line and the clock line of the IIC are both kept at a high level, the IIC is in an idle state. At this time, the data line changes from a high level to a low level, and the clock line continues to be at a high level, and the IIC bus starts data transmission. When the clock line is held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

### 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

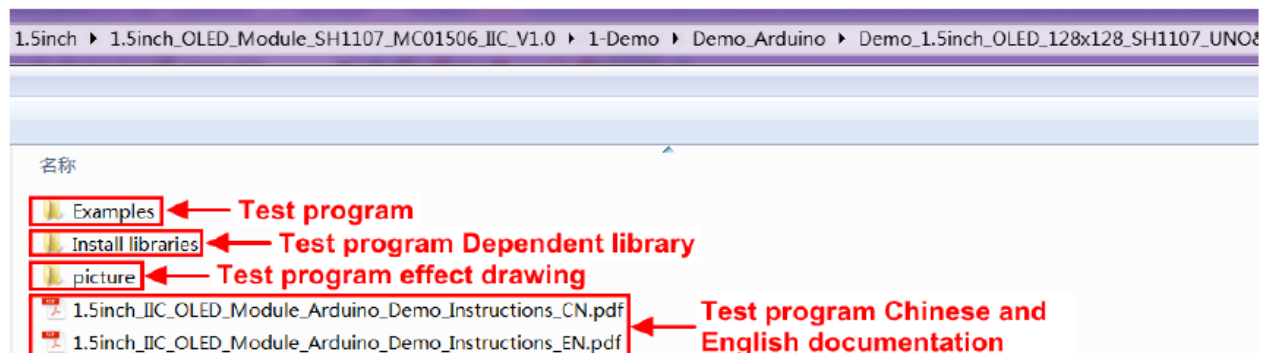
Arduino MEGA2560 microcontroller test program wiring instructions		
Number	Module Pin	Corresponding to MEGA2560 development board wiring pins
1	GND	GND
2	VCC	5V/3.3V
3	SCL	21

4	SDA	20
---	-----	----

## Operating Steps

- Connect the OLED module and the Arduino MCU according to the above wiring instructions, and power on;
- Select the example you want to test, as shown below:

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



- 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](http://www.lcdwiki.com/res/PublicFile/Arduino_IDE_Use_Illustration_EN.pdf)



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

## 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

### 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 wiring Pi coding 8

### Operating Steps

- 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

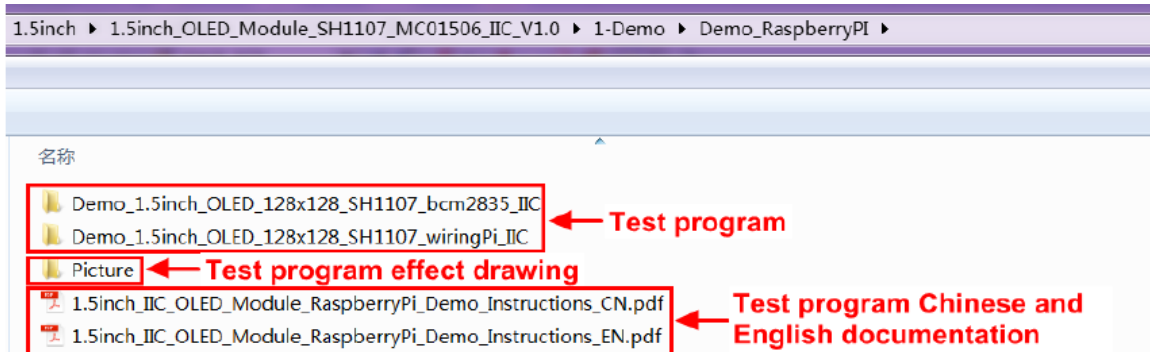
Start RaspberryPi's I2C kernel driver

- 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](http://www.lcdwiki.com/res/PublicFile/Raspberrypi_Use_Illustration_EN.pdf)

- select the example that needs to be tested, as shown below: (Please refer to the test program description document for test program description)



- bcm2835 instructions

- Connect the OLED module to the RaspberryPi development board according to the above wiring

- 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))

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

- wiringPi instructions

- Connect the OLED module to the RaspberryPi development board according to the above wiring

- 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))

- Run the following command to run the wiringPi test program:

```
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/csl/Demo_OLED_wiringPi_IIC/output/test.o
gcc -g -O0 -c /home/pi/csl/Demo_OLED_wiringPi_IIC/source/src/gui.c -o /home/pi/csl/Demo_OLED_wiringPi_IIC/output/gui.o
gcc -g -O0 -c /home/pi/csl/Demo_OLED_wiringPi_IIC/source/src/main.c -o /home/pi/csl/Demo_OLED_wiringPi_IIC/output/main.o
gcc -g -O0 -c /home/pi/csl/Demo_OLED_wiringPi_IIC/source/src/delay.c -o /home/pi/csl/Demo_OLED_wiringPi_IIC/output/delay.o
gcc -g -O0 -c /home/pi/csl/Demo_OLED_wiringPi_IIC/source/src/oled.c -o /home/pi/csl/Demo_OLED_wiringPi_IIC/output/oled.o
gcc -g -O0 -c /home/pi/csl/Demo_OLED_wiringPi_IIC/source/src/iic.c -o /home/pi/csl/Demo_OLED_wiringPi_IIC/output/iic.o
gcc -g -O0 /home/pi/csl/Demo_OLED_wiringPi_IIC/output/test.o /home/pi/csl/Demo_OLED_wiringPi_IIC/output/gui.o /home/pi/csl/Demo_OLED_wiringPi_IIC/output/main.o /home/pi/csl/Demo_OLED_wiringPi_IIC/output/delay.o /home/pi/csl/Demo_OLED_wiringPi_IIC/output/oled.o /home/pi/csl/Demo_OLED_wiringPi_IIC/output/iic.o -o /home/pi/csl/Demo_OLED_wiringPi_IIC/output/1.3_IIC_OLED
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

```

## 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

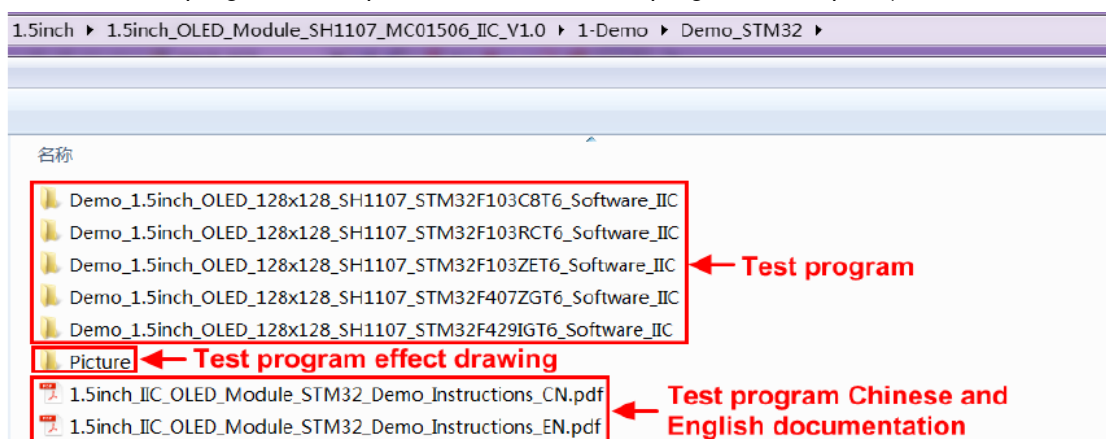
STM32F407ZGT6 microcontroller test program wiring instructions		
Number	Module Pin	Corresponding to the Explorer STM32F4 development board wiring pin
1	GND	GND
2	VCC	5V/3.3V
3	SCL	PB3
4	SDA	PB5

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

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



- 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](http://www.lcdwiki.com/res/PublicFile/STM32_Keil_Use_Illustration_EN.pdf)

- If the OLED module displays characters and graphics normally, the program runs successfully

## C51 instructions

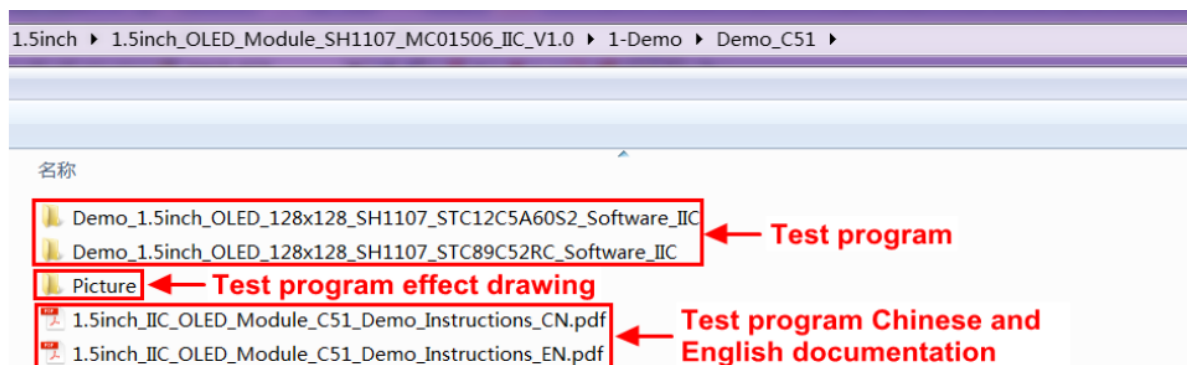
### Wiring instructions:

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

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



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

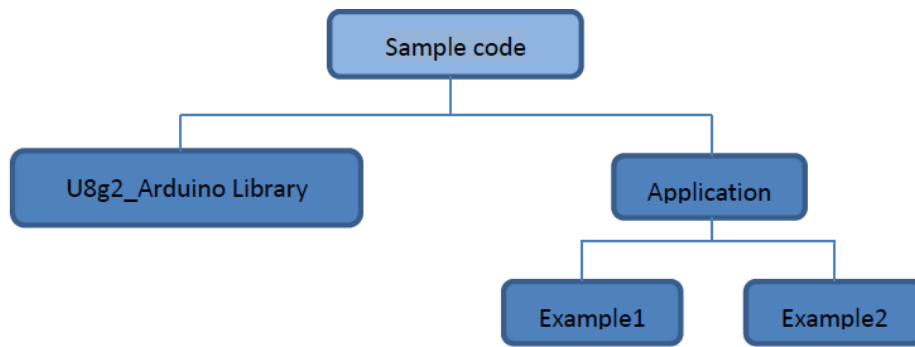
- If the OLED module displays characters and graphics normally, the program runs successfully;

## Software Description

### Code Architecture

#### Arduino code architecture description

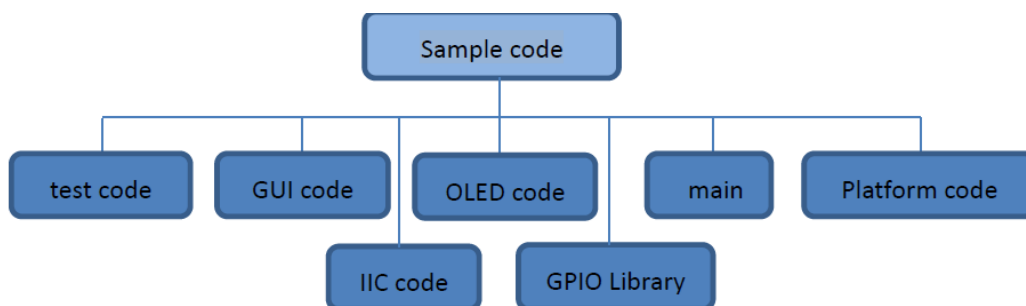
The code architecture is shown below



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.

- **RaspberryPi code architecture description**

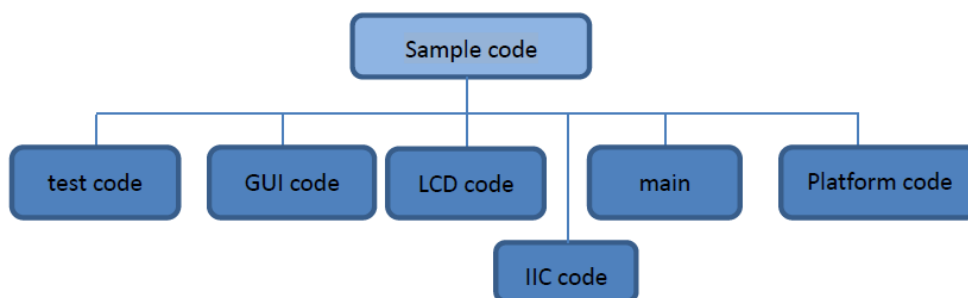
The bcm2835 and wiring Pi test program code architecture is as follows follows:



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;

**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 code; OLED initialization and related bin parallel port write data operations are included in the OLED codecode; Drawing points, lines, graphics, and Chinese and English character display related operations are included in the GUI code code; The main function implements the application to run run; Platform code varies by platform platform; IIC initialization and configuration related operations are included in the IIC code



code;

## GPIO definition description

- **Arduino test program GPIO definition description**

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

- **RaspberryPi test program GPIO definition description**

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

- **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 GPIO.

- **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 GPIO.

## IIC slave device address modification

- **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).

- **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);

- **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 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.

## IIC communication code implementation

### RaspberryPi test program IIC communication code implementation

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

```
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 written
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 written
78:  * @retvalue   :None
79:  *****/
80: void IIC_WriteDat(uint8_t I2C_Data)
81: {
82:     wiringPiI2CWriteReg8(iic_fd, IIC_DATA, I2C_Data);
83: }
84:
```

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:



```

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 written
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: }

```

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

### 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:

```

47 /*****
48 * @name      :void IIC_Start(void)
49 * @date      :2018-09-13
50 * @function   :start iic bus
51 * @parameters :None
52 * @retvalue   :None
53 *****/
54 void IIC_Start(void)
55 {
56     OLED_SCL_SET();
57     OLED_SDA_SET();
58     OLED_SDA_CLR();
59     OLED_SCL_CLR();
60 }
61
62 /*****
63 * @name      :void IIC_Stop(void)
64 * @date      :2018-09-13
65 * @function   :stop iic bus
66 * @parameters :None
67 * @retvalue   :None
68 *****/
69 void IIC_Stop(void)
70 {
71     OLED_SCL_SET();
72     OLED_SDA_CLR();
73     OLED_SDA_SET();
74 }
75
76 /*****
77 * @name      :void IIC_Wait_Ack(void)
78 * @date      :2018-09-13
79 * @function   :wait iic ack
80 * @parameters :None
81 * @retvalue   :None
82 *****/
83 void IIC_Wait_Ack(void)
84 {
85     OLED_SCL_SET();
86     OLED_SCL_CLR();
87 }

```

```

89 /*****
90 * @name      :void Write_IIC_Byte(u8 IIC_Byte)
91 * @date      :2018-09-13
92 * @function   :Write a byte of content with iic bus
93 * @parameters :IIC_Byte
94 * @retvalue   :None
95 *****/
96 void Write_IIC_Byte(u8 IIC_Byte)
97 {
98     u8 i;
99     u8 m,da;
100     da=IIC_Byte;
101     OLED_SCL_CLR();
102     for(i=0;i<8;i++)
103     {
104         m=da;
105         m=m&0x80;
106         if(m==0x80)
107         {
108             OLED_SDA_SET();
109         }
110         else
111         {
112             OLED_SDA_CLR();
113         }
114         da=da<<1;
115         OLED_SCL_SET();
116         OLED_SCL_CLR();
117     }
118 }

```

```

120  /*****
121  * @name      :void Write_IIC_Command(u8 IIC_Command)
122  * @date      :2018-09-13
123  * @function   :Write a byte of command to oled screen
124  * @parameters :IIC_Command:command to be written
125  * @retvalue   :None
126  *****/
127  void Write_IIC_Command(u8 IIC_Command)
128  {
129      IIC_Start();
130      Write_IIC_Byte(IIC_SLAVE_ADDR);          //Slave address,SA0=0
131      IIC_Wait_Ack();
132      Write_IIC_Byte(0x00);          //write command
133      IIC_Wait_Ack();
134      Write_IIC_Byte(IIC_Command);
135      IIC_Wait_Ack();
136      IIC_Stop();
137  }
138
139  /*****
140  * @name      :void Write_IIC_Data(u8 IIC_Data)
141  * @date      :2018-09-13
142  * @function   :Write a byte of data to oled screen
143  * @parameters :IIC_Data:data to be written
144  * @retvalue   :None
145  *****/
146  void Write_IIC_Data(u8 IIC_Data)

```

```

147  {
148      IIC_Start();
149      Write_IIC_Byte(IIC_SLAVE_ADDR);          //D/C#=0; R/W#=0
150      IIC_Wait_Ack();
151      Write_IIC_Byte(0x40);          //write data
152      IIC_Wait_Ack();
153      Write_IIC_Byte(IIC_Data);
154      IIC_Wait_Ack();
155      IIC_Stop();
156  }
157

```

### C51 test program IIC communication code implementation

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

```

47 /*****
48  * @name      :void IIC_Start(void)
49  * @date      :2018 09 13
50  * @function   :start iic bus
51  * @parameters :None
52  * @retvalue   :None
53 *****/
54 void IIC_Start(void)
55 {
56     OLED_SCL_SET();
57     OLED_SDA_SET();
58     OLED_SDA_CLR();
59     OLED_SCL_CLR();
60 }
61
62 /*****
63  * @name      :void IIC_Stop(void)
64  * @date      :2018-09-13
65  * @function   :stop iic bus
66  * @parameters :None
67  * @retvalue   :None
68 *****/
69 void IIC_Stop(void)
70 {
71     // OLED_SCL_SET();
72     OLED_SDA_CLR();
73     OLED_SDA_SET();
74 }

```

```

75
76 /*****
77  * @name      :void IIC_Wait_Ack(void)
78  * @date      :2018-09-13
79  * @function   :wait iic ack
80  * @parameters :None
81  * @retvalue   :None
82 *****/
83 void IIC_Wait_Ack(void)
84 {
85     OLED_SCL_SET();
86     OLED_SCL_CLR();
87 }
88

```

```

89  /*****
90  * @name      :void Write_IIC_Byte(u8 IIC_Byte)
91  * @date      :2018-09-13
92  * @function   :Write a byte of content with iic bus
93  * @parameters :IIC_Byte
94  * @retvalue   :None
95  *****/
96  void Write_IIC_Byte(u8 IIC_Byte)
97  {
98      u8 i;
99      u8 m,da;
100     da=IIC_Byte;
101     OLED_SCL_CLR();
102     for(i=0;i<8;i++)
103     {
104         m=da;
105         m=m&0x80;
106         if(m==0x80)
107         {
108             OLED_SDA_SET();
109         }
110         else
111         {
112             OLED_SDA_CLR();
113         }
114         da=da<<1;
115         OLED_SCL_SET();
116         OLED_SCL_CLR();
117     }
118 }

```

```

120  /*****
121  * @name      :void Write_IIC_Command(u8 IIC_Command)
122  * @date      :2018-09-13
123  * @function   :Write a byte of command to oled screen
124  * @parameters :IIC_Command:command to be written
125  * @retvalue   :None
126  *****/
127  void Write_IIC_Command(u8 IIC_Command)
128  {
129      IIC_Start();
130      Write_IIC_Byte(IIC_SLAVE_ADDR);          //Slave address,SA0=0
131      IIC_Wait_Ack();
132      Write_IIC_Byte(0x00);          //write command
133      IIC_Wait_Ack();
134      Write_IIC_Byte(IIC_Command);
135      IIC_Wait_Ack();
136      IIC_Stop();
137  }
138
139  /*****
140  * @name      :void Write_IIC_Data(u8 IIC_Data)
141  * @date      :2018-09-13
142  * @function   :Write a byte of data to oled screen
143  * @parameters :IIC_Data:data to be written
144  * @retvalue   :None
145  *****/
146  void Write_IIC_Data(u8 IIC_Data)
147  {

```

## 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 follows:

[http://www.lcdwiki.com/Chinese\\_and\\_English\\_display\\_modulo\\_settings](http://www.lcdwiki.com/Chinese_and_English_display_modulo_settings)

## FAQ

### Can I use this module with a voltage other than 3V~5V?

No, this module is specifically designed to work with a voltage supply between 3V and 5V.


### Can I use this module with a touchscreen?

No, this module does not support touch screen functionality.

Website: [www.lcdwiki.com](http://www.lcdwiki.com)

---

## Documents / Resources

	<p><a href="#">LCDWIKI MC130VX IIC OLED Module</a> [pdf] User Manual MC01506, MC130GX, MC130VX, MC130VX IIC OLED Module, MC130VX, IIC OLED Module, OLED Module, Module</p>
--	--

## References

- [🌐 Chinese and English display modulo settings - LCD wiki](#)
- [User Manual](#)