1.3inch 4-line-SPI IPS Module MSP1308 User Manual

Product Description

This product is a 1.3inch IPS display module,it has a resolution of 240x240.it uses a 4-wire SPI communication method and the inner IC is ST7789.The module contains an LCD display and PCB backboard.

Product Features

- 1.3-inch color screen,support 65K color display,display rich colors
- 240X240 resolution, clear display
- Large viewing angle(full angle),display color undistorted.
- Using the 4-line-SPI serial bus, it only takes a few IOs to illuminate the display
- Provide a rich STM32, C51,MSP430 and RaspberryPi sample program
- Military-grade process standards, long-term stable work
- Provide underlying driver technical support

Product Parameters

Name	Description
Display Color	RGB 65K color
SKU	MSP1308
Screen Size	1.3(inch)
Type	TFT
Driver IC	ST7789
Resolution	240*240 (Pixel)
Module Interface	4-line SPI interface
Active Area	23.40x23.40 (mm)
Touch Screen Type	have no touch screen
Touch IC	have no touch IC
Module PCB Size	39.22x27.78 (mm)

Angle of view	all angle
Operating Temperature	-10℃~60℃
Storage Temperature	-20℃~70℃
Operating Voltage	3.3V
Power Consumption	TBD
Product Weight(With packaging)	9(g)

Interface Description

Picture1. Module pin Label picture

important:

1. **The following pin numbers 1~7 refer to the module pin numbers of our company with PCB backplane. If you are buying a bare screen, please refer to the pin definition of the bare screen specification, refer to the wiring according to the signal type instead of directly according to the following. The module pin number is used for wiring. For example: DC is 6 feet on our module. It may be x pin on different size bare screen.**
2. **About VCC supply voltage: The IPS display module can only be connected to 3.3V.**
3. **About backlight voltage: The module with PCB backplane has integrated triode backlight control circuit, only need to input high level or PWM wave on BL pin to backlight. If you are buying a bare screen, the LEDAx is connected to 3.0V-3.3V, and the LEDKx can be grounded.**

Number	Module Pin	Pin Description
1	**GND**	LCD Power ground
2	**VCC**	LCD power supply is positive (3.3V)
3	**SCL**	LCD SPI bus clock signal
4	**SDA**	LCD SPI bus write data signal
5	**RES**	LCD reset control signal(Low level reset)
6	**DC**	LCD register / data selection control signal(Low level: register, high level: data)
7	**BLK**	LCD backlight control signal (high level lighting, if you do not need control, please connect 3.3V)

Hardware Configuration

The LCD module hardware circuit comprises two parts: an LCD display control circuit and a backlight control circuit.

The LCD display control circuit is used to control the pins of the LCD, including control pins and data transfer pins.

The backlight control circuit is used to control the backlight to be on and off. Of course, if the backlight is not required to be be on and off, can be directly connected to the 3.3V power supply.

working principle

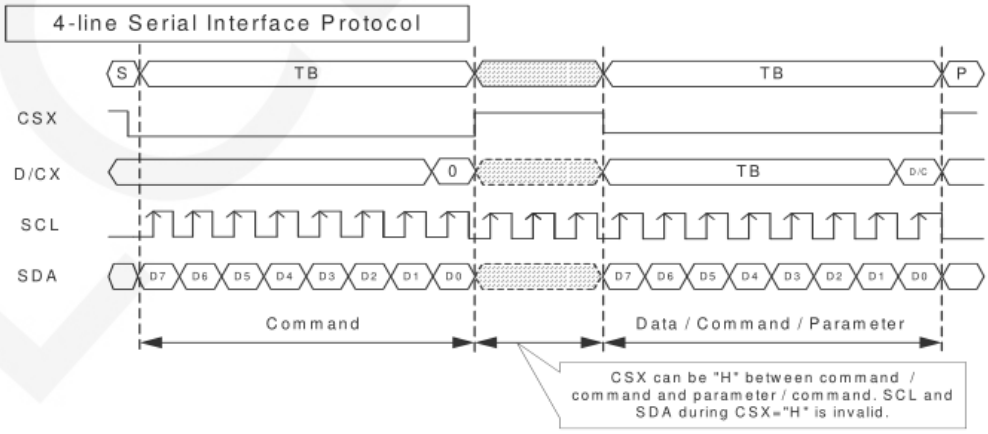
1. Introduction to ST7789 Controller

The ST7789 controller supports a maximum resolution of 240*320 and a 172800-byte GRAM. It also supports 8-bit, 9-bit, 16-bit, and 18-bit parallel port data buses. It also supports 3-wire and 4-wire SPI serial ports. Since parallel control requires a large number of IO ports, the most common one is SPI serial port control. The ST7789 also supports 65K, 262K RGB color display, display color is very rich, while supporting rotating display and scroll display and video playback, display in a variety of ways.

The ST7789 controller uses 16bit (RGB565) to control a pixel display, so it can display up to 65K colors per pixel. The pixel address setting is performed in the order of rows and columns, and the incrementing and decreasing direction is determined by the scanning mode. The ST7789 display method is performed by setting the address and then setting the color value.

2. Introduction to SPI communication protocol

The 4-wire SPI bus write mode timing is shown in the following figure:

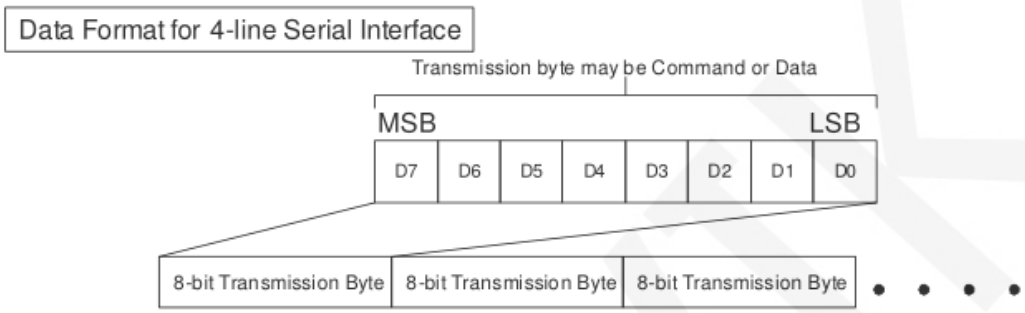


CSX is a slave chip select, and the chip is enabled only when CSX is low.

D/CX is the data/command control pin of the chip. When DCX is low, the command is written. When it is high, the data is written.

SCL is the SPI bus clock, and each rising edge transmits 1 bit of data;

SDA is the data transmitted by SPI, and it transmits 8-bit data at a time. The data format is as shown below:



The high position is in front and transmitted first.

For SPI communication, the data has a transmission timing, that is, a combination of clock phase (CPHA) and clock polarity (CPOL):

The CPOL level determines the idle state level of the serial synchronous clock, CPOL = 0, which is low. CPOL does not have a lot of impact on the transport protocol;

The level of CPHA determines whether the serial synchronous clock is acquired on the first clock transition edge or the second clock transition edge.

When CPHL = 0, data acquisition is performed on the first edge of the transition;

The combination of the two becomes the four SPI communication methods. SPI0 is usually used in China, that is, CPHL = 0, CPOL = 0.

Instructions for use

1. STM32 instructions

Wiring instructions:

See the interface description for pin assignments.

STM32F103RCT6 microcontroller test program wiring instructions		
Number	Module Pin	Corresponding to MiniSTM32 development board wiring pin
1	GND	GND
2	VCC	3.3V
3	SCL	PB13
4	SDA	PB15
5	RES	PB12
6	DC	PB10
7	BLK	PB9

STM32F103ZET6 microcontroller test program wiring instructions		
Number	Module Pin	Corresponding to Elite STM32 development board wiring pin
1	GND	GND
2	VCC	3.3V
3	SCL	PB13
4	SDA	PB15
5	RES	PB12
6	DC	PB10
7	BLK	PB9

STM32F407ZGT6 microcontroller test program wiring instructions		
Number	Module Pin	Corresponding to Explorer STM32F4 development board wiring pin
1	GND	GND

2	VCC	3.3V
3	SCL	PB3
4	SDA	PB5
5	RES	PB12
6	DC	PB14
7	BLK	PB13

STM32F429IGT6 microcontroller test program wiring instructions		
Number	Module Pin	Corresponding to Apollo STM32F4/F7 development board wiring pin
1	GND	GND
2	VCC	3.3V
3	SCL	PF7
4	SDA	PF9
5	RES	PD12
6	DC	PD5
7	BLK	PD6

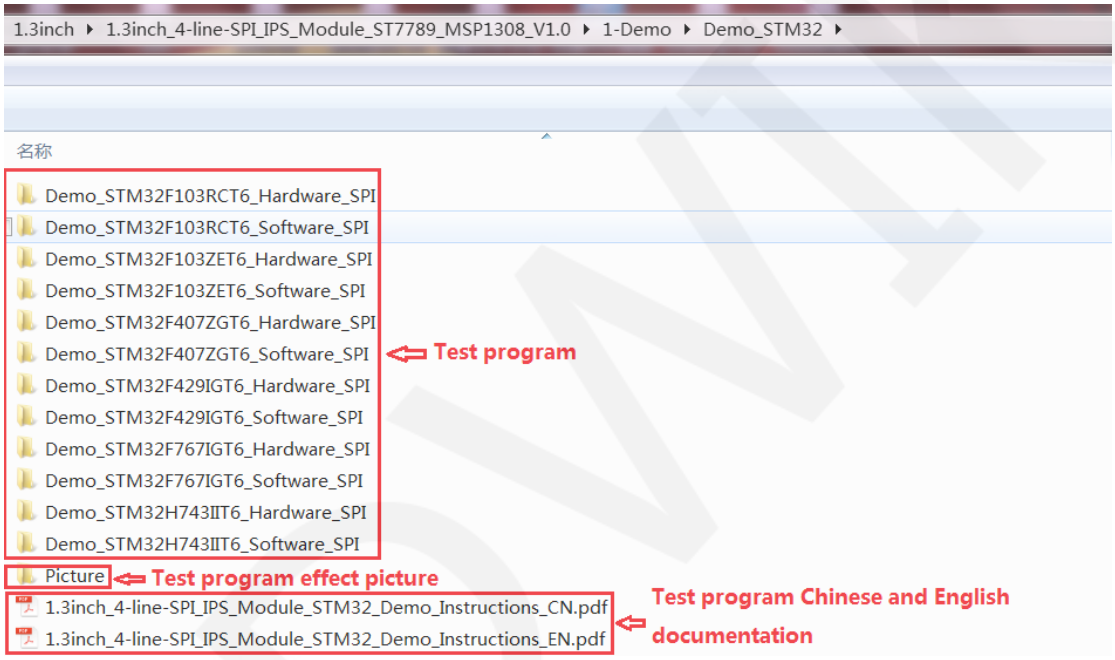
STM32F767IGT6 and STM32H743IIT6 microcontroller test program wiring instructions		
Number	Module Pin	Corresponding to Apollo STM32F4/F7 development board wiring pin
1	GND	GND
2	VCC	3.3V
3	SCL	PB13
4	SDA	PB15
5	RES	PD12
6	DC	PD5
7	BLK	PD6

Operating Steps:

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

B. Select the test example according to the model of the microcontroller, as shown in the following figure:

(Please refer to the test program description document in the test package for the 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 IPS module displays characters and graphics normally, the program runs successfully;

2. C51 instructions

Wiring instructions:

See the interface description for pin assignments.

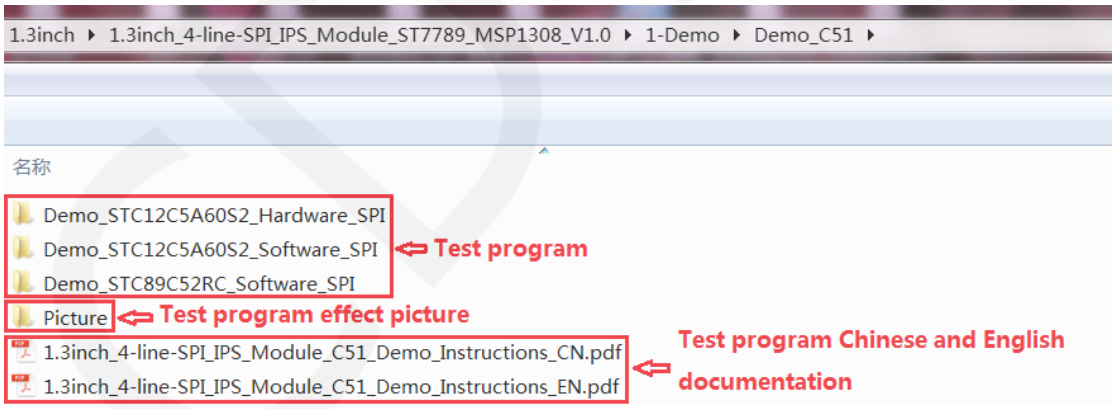
STC89C52RC and STC12C5A60S2 microcontroller test program wiring instructions		
Number	**Module Pin**	**Corresponding to STC89/STC12 development board wiring pin**
1	**GND**	GND
2	**VCC**	3.3V
3	**SCL**	P17
4	**SDA**	P15
5	**RES**	P33
6	**DC**	P12
7	**BLK**	P32

Operating Steps:

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

B. Select the C51 test program to be tested, as shown below:

(Please refer to the test program description document in the test package for the 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 IPS module displays characters and graphics normally, the program runs

successfully;

3. MSP430 instructions

Wiring instructions:

See the interface description for pin assignments.

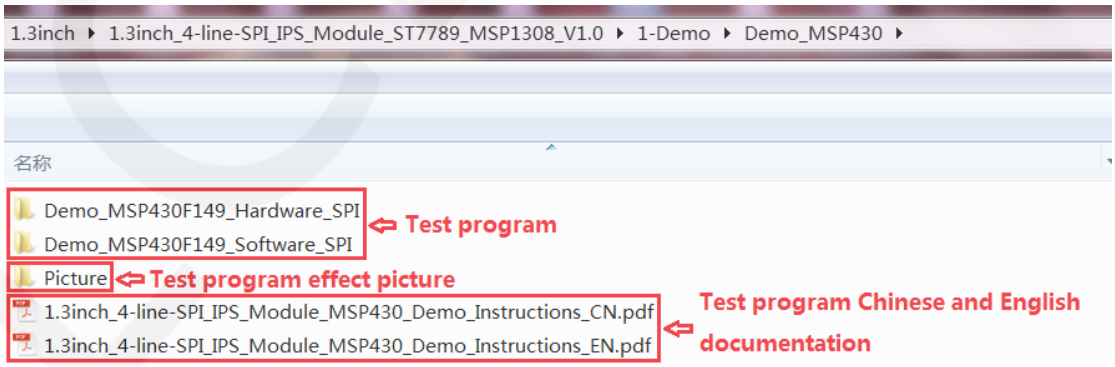
MSP430F149 microcontroller test program wiring instructions		
Number	**Module Pin**	**Corresponding to MSP430 development board wiring pin**
1	**GND**	GND
2	**VCC**	3.3V
3	**SCL**	P33
4	**SDA**	P31
5	**RES**	P22
6	**DC**	P21
7	**BLK**	P20

Operating Steps:

A. Connect the IPS module and the MSP430 MCU according to the above wiring instructions, and power on;

B. Select the MSP430 test program to be tested, as shown below:

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



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

detailed description of the MSP430 test program compilation and download can

be found in the following document:

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

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

4. 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**	**3.3V** **（Physical pin：1,17）**
3	**SCL**	**Physical pin：23** **BCM coding：11** **wiringPi coding：14**
4	**SDA**	**Physical pin：19** **BCM coding：10** **wiringPi coding：12**
5	**RES**	**Physical pin：5**

		BCM coding：3 wiringPi coding：9
6	DC	Physical pin：3 BCM coding：2 wiringPi coding：8
7	BLK	Physical pin：12 BCM coding：18 wiringPi coding：1

Operating Steps：

A. open the SPI 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->SPI->YES

Start RaspberryPi's SPI kernel driver

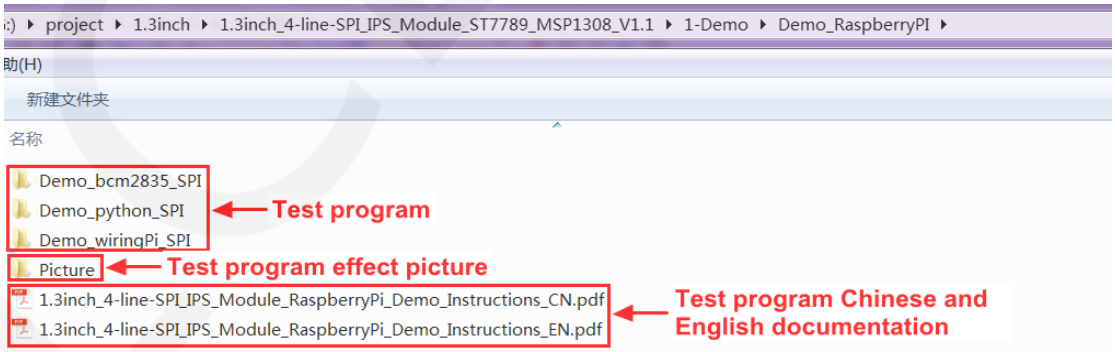
B. install the function library

For detailed installation methods of the bcm2835, wiringPi, and python 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 LCD module to the RaspberryPi development board according to

the above wiring

b) Copy the test program directory Demo_bcm2835 _SPI 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_bcm2835 _SPI

make

sudo ./1.3_IPS_LCD

As shown below:

```
pi@raspberrypi:~/0407 $ cd Demo_bcm2835_SPI/
pi@raspberrypi:~/0407/Demo_bcm2835_SPI $ make
gcc -g -O0 -c  /home/pi/0407/Demo_bcm2835_SPI/source/src/lcd.c -o /home/pi/0407/Demo_bcm2835_
gcc -g -O0 -c  /home/pi/0407/Demo_bcm2835_SPI/source/src/test.c -o /home/pi/0407/Demo_bcm2835_
gcc -g -O0 -c  /home/pi/0407/Demo_bcm2835_SPI/source/src/spi.c -o /home/pi/0407/Demo_bcm2835_
gcc -g -O0 -c  /home/pi/0407/Demo_bcm2835_SPI/source/src/gui.c -o /home/pi/0407/Demo_bcm2835_
gcc -g -O0 -c  /home/pi/0407/Demo_bcm2835_SPI/source/src/main.c -o /home/pi/0407/Demo_bcm2835_
gcc -g -O0 -c  /home/pi/0407/Demo_bcm2835_SPI/source/src/delay.c -o /home/pi/0407/Demo_bcm283
gcc -g -O0 /home/pi/0407/Demo_bcm2835_SPI/output/lcd.o /home/pi/0407/Demo_bcm2835_SPI/output/
o /home/pi/0407/Demo_bcm2835_SPI/output/main.o /home/pi/0407/Demo_bcm2835_SPI/output/delay.o
pi@raspberrypi:~/0407/Demo_bcm2835_SPI $ sudo ./1.3_IPS_LCD
```

E. wiringPi instructions

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

b) Copy the test program directory Demo_wiringPi _SPI 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_wiringPi _SPI

make

sudo ./1.3_IPS_LCD

As shown below:

```
pi@raspberrypi:~/0407 $ cd Demo_wiringPi_SPI/
pi@raspberrypi:~/0407/Demo_wiringPi_SPI $ make
gcc -g -O0 -c  /home/pi/0407/Demo_wiringPi_SPI/source/src/lcd.c -o /home/pi/0407/Demo_wir
gcc -g -O0 -c  /home/pi/0407/Demo_wiringPi_SPI/source/src/test.c -o /home/pi/0407/Demo_wi
gcc -g -O0 -c  /home/pi/0407/Demo_wiringPi_SPI/source/src/spi.c -o /home/pi/0407/Demo_wir
gcc -g -O0 -c  /home/pi/0407/Demo_wiringPi_SPI/source/src/gui.c -o /home/pi/0407/Demo_wir
gcc -g -O0 -c  /home/pi/0407/Demo_wiringPi_SPI/source/src/main.c -o /home/pi/0407/Demo_wi
gcc -g -O0 -c  /home/pi/0407/Demo_wiringPi_SPI/source/src/delay.c -o /home/pi/0407/Demo_w
gcc -g -O0 /home/pi/0407/Demo_wiringPi_SPI/output/lcd.o /home/pi/0407/Demo_wiringPi_SPI/o
gui.o /home/pi/0407/Demo_wiringPi_SPI/output/main.o /home/pi/0407/Demo_wiringPi_SPI/outpu
pi@raspberrypi:~/0407/Demo_wiringPi_SPI $ sudo ./1.3_IPS_LCD
```

F. python instructions

a) The image processing library PIL needs to be installed before running the python test program. The specific installation method is as follows:

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

b) Connect the LCD module to the RaspberryPi development board as described above.

c) Copy the test program directory Demo_python _SPI to RaspberryPi (either via SD card or via FTP tool (such as FileZilla))

d) Run the following command to run python test programs:

cd Demo_python _SPI/source

sudo python 1.3_IPS_LCD.py

As shown below:

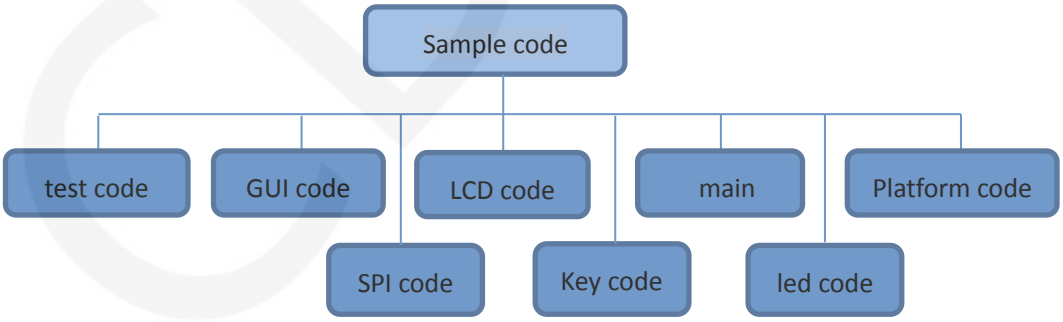
```
pi@raspberrypi:~/0408 $ cd Demo_python_SPI/source/
pi@raspberrypi:~/0408/Demo_python_SPI/source $ sudo python 1.3_IPS_LCD.py
```

Software Description

1. Code Architecture

A. C51, STM32 and MSP430 code architecture description

The code architecture is shown below:



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

LCD initialization and related operations are included in the LCD 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;

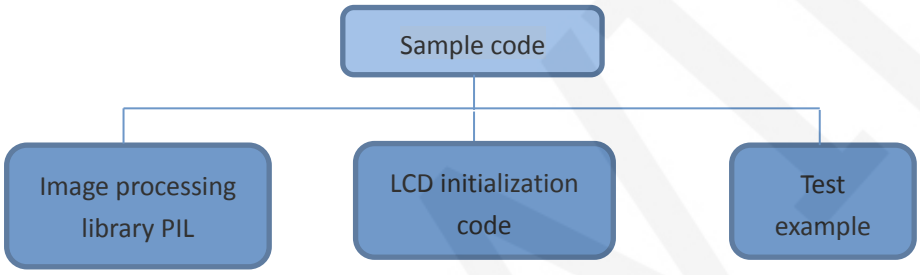
SPI initialization and configuration related operations are included in the SPI code;

The key processing related code is included in the key code (the C51 and MSP 430 platform does not have a button processing code);

The code related to the led configuration operation is included in the led code(the MSP 430 platform does not have a led code);

B. RaspberryPi code architecture description

The python test program code architecture is shown below:



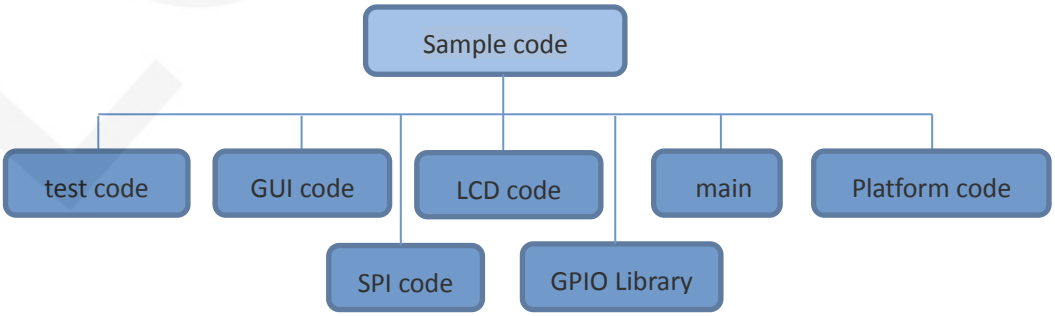
The python test program consists of but part: PIL image processing library, LCD initialization code, test sample code

PIL image processing library is responsible for image drawing, character and text display operations, etc.

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

The test example is to use the API provided by the above two parts of the code to implement some test functions.

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;

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

2. software SPI and hardware SPI description

The IPS module provides software SPI and hardware SPI sample code (except STC89C52RC, because it does not have hardware SPI function), the two sample code does not make any difference in the display content, but the following aspects are different:

A. display speed

The hardware SPI is significantly faster than the software SPI, which is determined by the hardware.

B. GPIO definition

The software SPI all control pins must be defined, any idle pin can be used, the hardware SPI data and clock signal pins are fixed (depending on the platform), other control pins should be defined by themselves, or any idle reference can be used. foot.

C. initialization

When the software SPI is initialized, only the GPIO for pin definition needs to be initialized (not required by the C51 platform). When the hardware SPI is initialized, the relevant control registers and data registers need to be initialized.

3. GPIO definition description

A. STM32 test program GPIO definition description

The STM32's LCD non-SPI GPIO definition is placed in lcd.h as shown below(take STM32F407ZGT6 microcontroller FSMC test program as an example):

```
#define LED  13          //背光控制引脚
//#define CS   15          //片选引脚
#define RS   14         //寄存器/数据选择引脚
#define RST  12         //复位引脚
```

All pin definitions can be modified and can be defined as any other free GPIO.

The GPIO definition of the STM32 LCD SPI is placed in spi.h as shown below(take STM32F407ZGT6 microcontroller FSMC test program as an example):

```
#define SCLK        3 //PB13--->>TFT --SCL/SCK
#define MISO        4
#define MOSI        5 //PB15 MOSI--->>TFT --SDA/DIN
```

If the software SPI is used, all pin definitions can be modified and can be defined as any other free GPIO.

If hardware SPI is used, these pins do not need to be defined. At the same time, the SCLK, MISO and MOSI pins need to be initialized and removed in the LCD_ GPIOInit function in

the lcd.c file, as shown in the following figure(take STM32F407ZGT6 microcontroller FSMC test program as an example):

```
void LCD_GPIOInit(void)
{
  GPIO_InitTypeDef  GPIO_InitStructure;

  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB ,ENABLE);

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3|GPIO_Pin_5|GPIO_Pin_12| GPIO_Pin_13|GPIO_Pin_14| GPIO_Pin_15;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;//普通输出模式
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;   //推挽输出
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;//上拉
  GPIO_Init(GPIOB, &GPIO_InitStructure);

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
  GPIO_Init(GPIOB, &GPIO_InitStructure);
}
```

The contents of the red circle must be removed.

B. C51 test program GPIO definition description

C51 test program GPIO definition is placed in the lcd.h file, as shown below:

```
//IO连接
sbit LCD_RS = P1^2;       //数据/命令切换
sbit LCD_SDI = P1^5;       //SPI写
sbit LCD_SDO = P1^6;          //SPI读
//sbit LCD_CS = P1^3;    //片选
sbit LCD_CLK = P1^7;    //SPI时钟
sbit LCD_RESET = P3^3;         //复位
sbit LCD_BL=P3^2;    //背光控制，如果不需要控制，接3.3V
```

If the software SPI is used, all pin definitions can be modified and can be defined as any other free GPIO.

If hardware SPI is used, the LCD_BL, LCD_RS, LCD_CS, and LCD_RST pin definitions can be modified and can be defined as any other free GPIO. LCD_CLK and LCD_SDI do not need to be defined.

C. MSP430 test program GPIO definition description

The MSP430's LCD non-SPI GPIO definition is placed in lcd.h as shown below:

```
//////////////////////////////////////////////////////////////////////
//-----------------LCD端口定义----------------
#define LED      BIT0        //背光控制引脚   P20
#define LCD_RS   BIT1        //寄存器/数据选择引脚 P21
#define LCD_RST  BIT2        //复位引脚            P22
```

All pin definitions can be modified and can be defined as any other free GPIO.

The GPIO definition of the MSP430 LCD SPI is placed in spi.h as shown below:

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

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

If the software SPI is used, all pin definitions can be modified and can be defined as any other free GPIO.

If hardware SPI is used, these pins do not need to be defined.

D. RaspberryPi test program GPIO definition description

The RaspberryPi test program uses hardware SPI, so only three GPIO ports need to be defined. The bcm2835 and WiringPi test programs place the GPIO definition in the oled.h file, as shown in the following figure:

```
//////////////////////////////////////////////////////////////////////
//-----------------LCD port definition----------------
#define LCD_LED  18          //LCD backlight control signal bcm:18
#define LCD_RS   2           //data or command selection control signal bcm:2
#define LCD_RST  3           //reset control signal  bcm:3
```

The Python test program places the GPIO definition in each test example, as shown in the following figure:

```
DC =  2
RES = 3
BLK = 18
```

These three GPIOs can be modified according to the corresponding GPIO library code.

4. SPI communication code implementation

A. STM32 test program SPI communication code implementation

Hardware SPI communication is implemented by the system. We only need to operate the register and call the relevant function. For details, please refer to the MCU related documentation.

The software SPI communication code is implemented in spi.c ,as shown in the following figure:

```
void  SPIv_WriteData(u8 Data)
{
  unsigned char i=0;
  for(i=8;i>0;i--)
  {
    if(Data&0x80)
      LCD_SDA_SET; //输出数据
    else
      LCD_SDA_CLR;
    LCD_SCL_CLR;
    LCD_SCL_SET;
    Data<<=1;
  }
}
```

If the transmitted data bit is 1, the SPI data pin is pulled high. When it is 0, the SPI data pin is pulled low, one byte of data is transferred each time, the upper bit is first, and one bit of data is transmitted on each rising edge of the clock.

B. C51 test program SPI communication code implementation

The software SPI communication code is implemented in lcd.c ,as shown in the following figure:

```
void spi_write_byte(u8 d)
{
  u8 val=0x80;
  while(val)
  {
    if(d&val)
    {
      LCD_SDI = 1;
    }
    else
    {
      LCD_SDI = 0;
    }
    LCD_CLK = 0;
    LCD_CLK = 1;
    val>>=1;
  }
}
```

If the transmitted data bit is 1, the SPI data pin is pulled high. When it is 0, the SPI data pin is pulled low, one byte of data is transferred each time, the upper bit is first, and one bit of data is transmitted on each rising edge of the clock.

C. MSP430 test program SPI communication code implementation

The software SPI communication code is implemented in spi.c ,as shown in the following figure:

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

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

If the transmitted data bit is 1, the SPI data pin is pulled high. When it is 0, the SPI data pin is pulled low, one byte of data is transferred each time, the upper bit is first, and one bit of data is transmitted on each rising edge of the clock.

D. RaspberryPi test program SPI communication code implementation

The SPI communication code for the bcm2835 and wiringPi test programs is implemented in spi.c.

The SPI communication code for the python test program is implemented in lcd_spi.py.

The bcm2835 test program hardware SPI code is implemented as shown below:

```
/*****************************************************************************
 * @name        :void SPI_WriteByte(uint8_t byte)
 * @date        :2021-04-06
 * @function    :Write a byte of data using RaspberryPi hardware SPI
 * @parameters  :Byte:Data to be written
 * @retvalue    :Data received by the bus
******************************************************************************/
void SPI_WriteByte(uint8_t byte)
{
    bcm2835_spi_transfer(byte);
}
```

The wiringPi test program hardware SPI code is implemented as shown below:

```
/*****************************************************************************
 * @name        :void SPI_WriteByte(uint8_t byte)
 * @date        :2021-04-06
 * @function    :Write a byte of data using RaspberryPi hardware SPI
 * @parameters  :Byte:Data to be written
 * @retvalue    :Data received by the bus
******************************************************************************/
void SPI_WriteByte(uint8_t byte)
{
    wiringPiSPIDataRW(CHANNEL,&byte,1);
}
```

The python test program hardware SPI code is implemented as shown below:

```
class lcdspi(object):
    def __init__(self):
        self.spi=spidev.SpiDev()
        self.spi.open(bus,device) #open spi device
        self.spi.max_speed_hz = 64000000
        self.spi.mode = 0b10
    def spiwritebyte(self,val):
        self.spi.writebytes([val])
#   self.spi.xfer([val],64000000)
```

Common software

This set of test examples requires the display of Chinese and English, symbols and pictures, so the modulo software is used. There are two types of modulo software: Image2Lcd and PCtoLCD2002. Here is only the setting of the modulo software 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

Take the model to choose the direction (high 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

Image2Lcd modulo software settings are shown below:

The Image2Lcd software needs to be set to horizontal, left to right, top to bottom, and low position to the front scan mode.