

# 1.8inch SPI Module MSP1803 User Manual

## Product Description

The LCD module uses a 4-wire SPI communication method with a driver IC of ST7735S and a resolution of 128x160. The module contains an LCD display and backlight control circuitry.

## Product Features

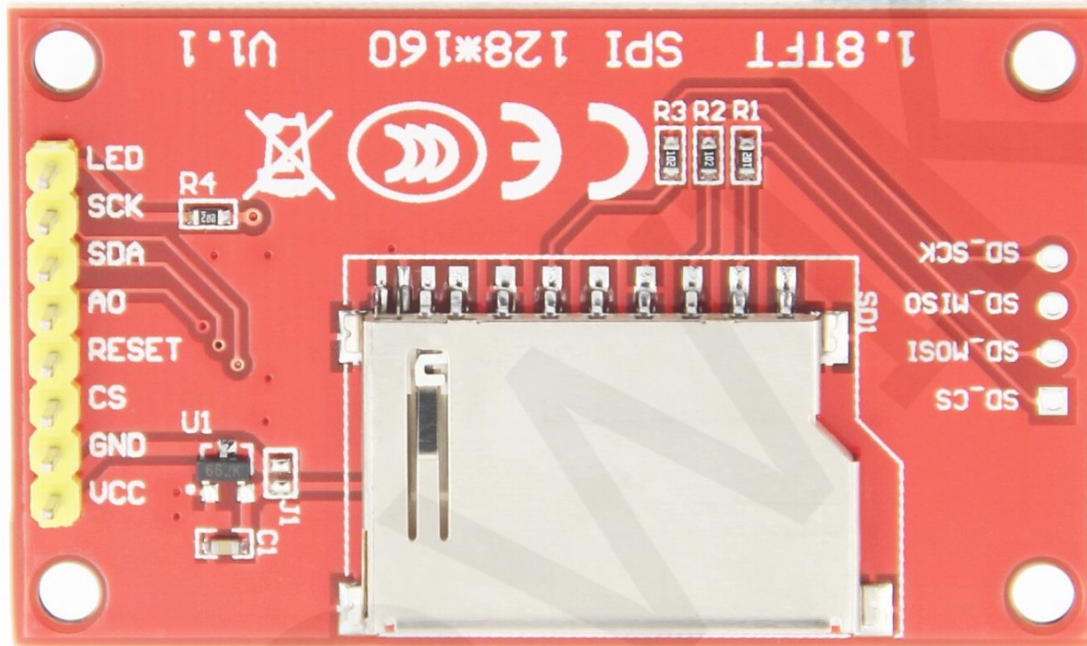
- 1.8-inch color screen, support 65K color display, display rich colors
- 128X160 resolution, clear display
- Using the SPI serial bus, it only takes a few IOs to illuminate the display
- Easy to expand the experiment with SD card slot
- Provide a rich sample program
- Military-grade process standards, long-term stable work
- Provide underlying driver technical support

## Product Parameters

Name	Description
Display Color	RGB 65K color
SKU	<b>MSP1803</b>
Screen Size	1.8(inch)
Type	TFT
Driver IC	ST7735S
Resolution	128*160 (Pixel)
Module Interface	4-wire SPI interface
Active Area	28.03x35.04 (mm)
Module PCB Size	34.5x58.0 (mm)
Angle of view	>60°
Operating Temperature	-10°C~60°C

Storage Temperature	-20°C~70°C
Operating Voltage	3.3V / 5V
Power Consumption	About 90mA
Product Weight	About 25(g)

## Interface Description



Pin silkscreen picture

Number	Module Pin	Pin Description
1	VCC	LCD power supply is positive (3.3V~5V)
2	GND	LCD Power ground
3	CS	LCD selection control signal
4	RESET	LCD reset control signal
5	A0	LCD register / data selection control signal
6	SDA	LCD SPI bus write data signal
7	SCK	LCD SPI bus clock signal
8	LED	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 controlled, the backlight control pin can be directly connected to the 3.3V power supply without using the circuit.

## working principle

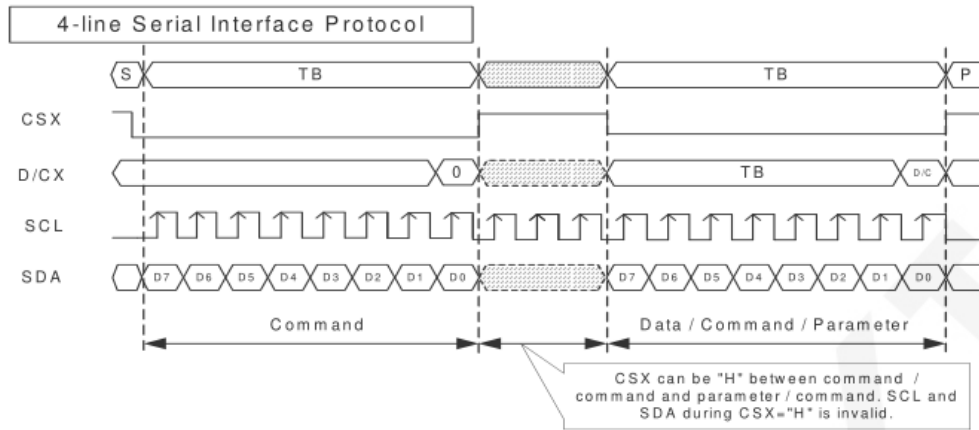
### 1. Introduction to ST7735S Controller

The ST7735S controller supports a maximum resolution of 132\*162 and a 48114-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 ST7735S 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 ST7735S 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 ST7735S 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:



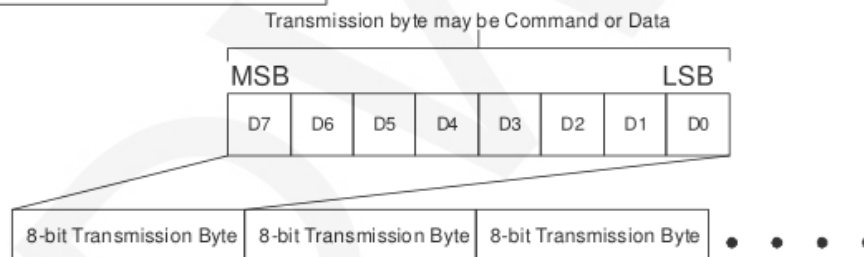
CSX is a slave chip select, and the chip is enabled only when CSX is low.

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:

**Data Format for 4-line Serial Interface**



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. Arduino instructions

Wiring instructions:

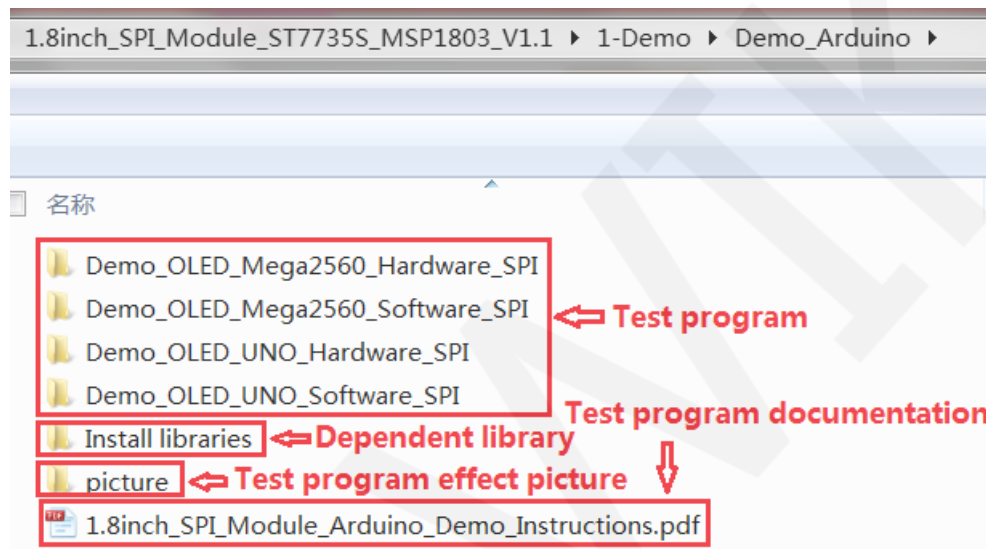
Arduino UNO microcontroller test program wiring		
Number	Module Pin	Corresponding to UNO development board wiring pins
1	LED	A0
2	SCK	13
3	SDA	11
4	A0	A3
5	RESET	A4
6	CS	A5
7	GND	GND
8	VCC	5V/3.3V

Arduino MEGA2560 microcontroller test program wiring		
Number	Module Pin	Corresponding to MEGA2560 development board wiring pins
1	LED	A0
2	SCK	52
3	SDA	51
4	A0	A3
5	RESET	A4
6	CS	A5
7	GND	GND
8	VCC	5V/3.3V

#### Operating Steps:

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

- B. Copy the dependent libraries in the Install libraries directory of the test package to the libraries folder of the Arduino project directory (if you do not need to depend on the libraries, you do not need to copy them);
- C. Open the directory where the Arduino test program is located and select the example you want to test, as shown below:
- (Please refer to the test program description document in the test package for the test program description)



- D. 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)
- E. If the LCD module displays characters and graphics normally, the program runs successfully;

## 2. C51 instructions

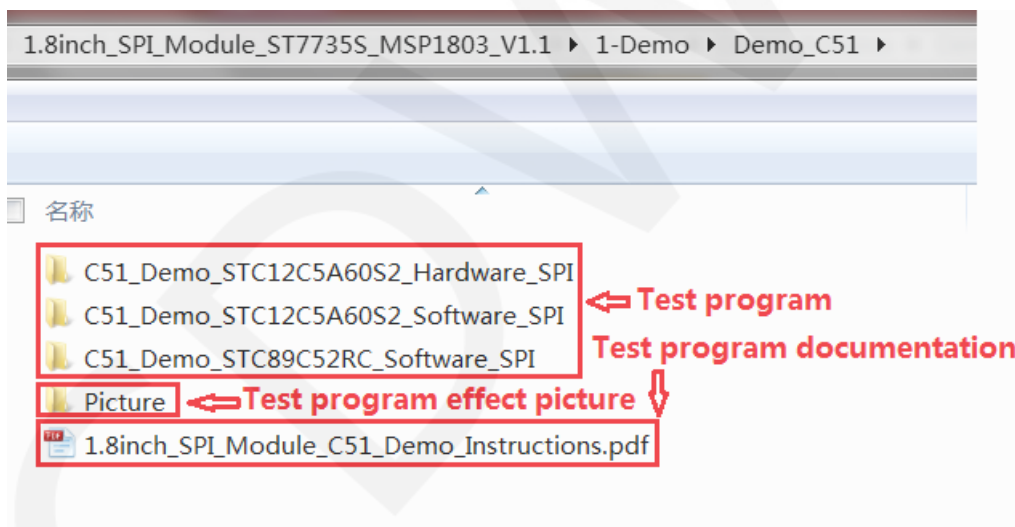
### Wiring instructions:

STC89C52RC and STC12C5A60S2 microcontroller test program wiring instructions		
Number	Module Pin	Corresponding to STC89/STC12 development board wiring pin
1	VCC	5V/3.3V

2	GND	GND
3	CS	P13
4	RESET	P33
5	A0	P12
6	SDA	P15
7	SCK	P17
8	LED	P32

### Operating Steps:

- A. Connect the LCD 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](http://www.lcdwiki.com/res/PublicFile/C51_Keil%26stc-isp_Use_Illustration_EN.pdf)
- D. If the LCD module displays characters and graphics normally, the program runs successfully;



### 3. STM32 instructions

#### Wiring instructions:

Because the pin positions of different development boards are different, and the external pins are reserved differently (some development boards do not have externally required pins). In order to facilitate wiring, the wiring pins of each development board are inconsistent.

<b>STM32F103RCT6 microcontroller test program wiring instructions</b>		
<b>Number</b>	<b>Module Pin</b>	<b>Corresponding to MiniSTM32 development board wiring pin</b>
1	VCC	5V/3.3V
2	GND	GND
3	CS	PB11
4	RESET	PB12
5	A0	PB10
6	SDA	PB15
7	SCK	PB13
8	LED	PB9

<b>STM32F103ZET6 microcontroller test program wiring instructions</b>		
<b>Number</b>	<b>Module Pin</b>	<b>Corresponding to Elite STM32 development board wiring pin</b>
1	VCC	5V/3.3V
2	GND	GND
3	CS	PB11
4	RESET	PB12
5	A0	PB10
6	SDA	PB15

7	SCK	PB13
8	LED	PB9

### STM32F407ZGT6 microcontroller test program wiring instructions

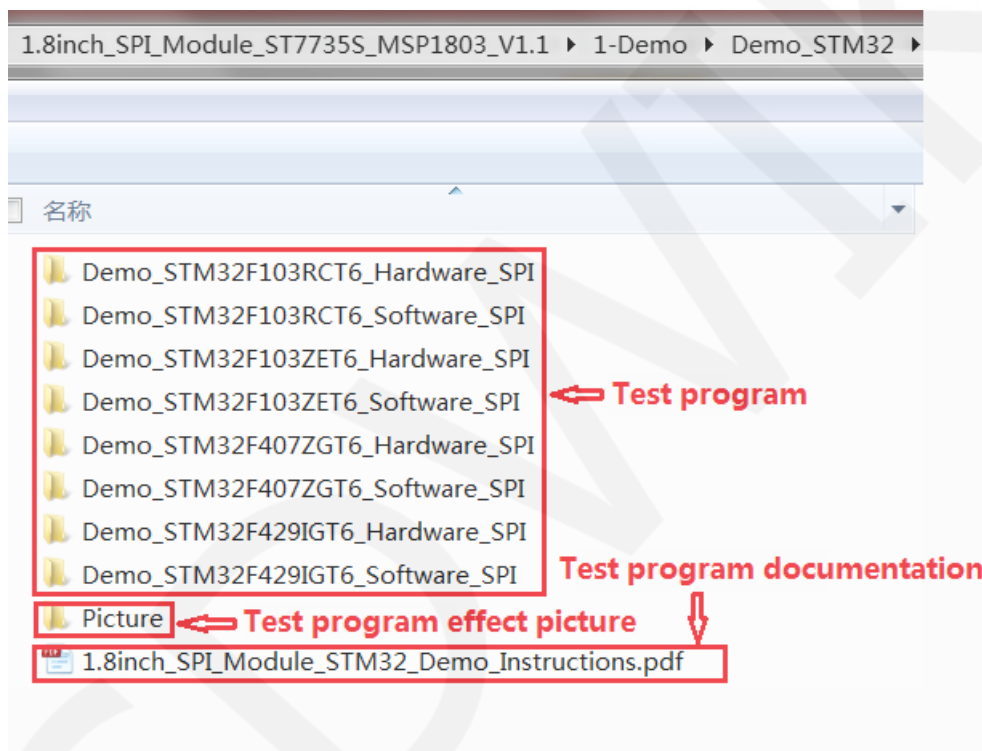
Number	Module Pin	Corresponding to Explorer STM32F4 development board wiring pin
1	VCC	5V/3.3V
2	GND	GND
3	CS	PB15
4	RESET	PB12
5	A0	PB14
6	SDA	PB5
7	SCK	PB3
8	LED	PB13

### STM32F429IGT6 microcontroller test program wiring instructions

Number	Module Pin	Corresponding to Apollo STM32F4/F7 development board wiring pin
1	VCC	5V/3.3V
2	GND	GND
3	CS	PD11
4	RESET	PD12
5	A0	PD5
6	SDA	PF9
7	SCK	PF7
8	LED	PD6

### Operating Steps:

- A. Connect the LCD 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)



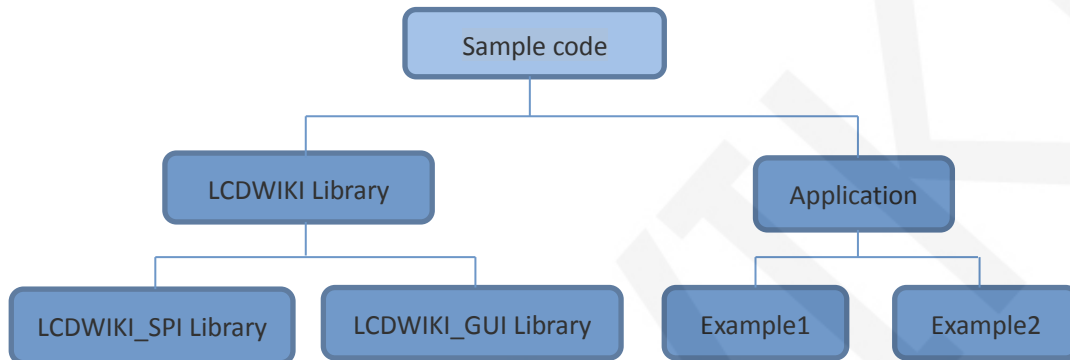
- C. Open the selected test program project, compile and download;  
detailed description of the STM32 test program compilation and download can be found in the following document:  
[http://www.lcdwiki.com/res/PublicFile/STM32\\_Keil\\_Use\\_Illustration\\_EN.pdf](http://www.lcdwiki.com/res/PublicFile/STM32_Keil_Use_Illustration_EN.pdf)
- D. If the LCD 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 LCDWIKI library and application code.

The LCDWIKI library consists of two parts: the LCDWIKI\_SPI library and the LCDWIKI\_GUI library.

The application contains several test examples, each of which contains different test content.

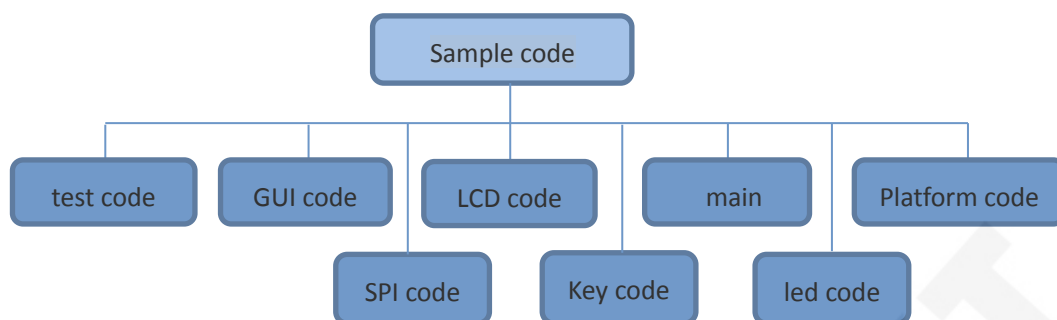
LCDWIKI\_SPI is the underlying library, which is associated with hardware. It is mainly responsible for operating registers, including hardware module initialization, data and command transmission, pixel coordinates and color settings, and display mode configuration.

LCDWIKI\_GUI is a middle-tier library, which is responsible for drawing graphics and displaying characters using the API provided by the underlying library.

The application uses the API provided by the LCDWIKI library to write some test examples to implement some aspects of the test function.

#### B. C51 and STM32 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 platform does not have a button processing code);

The code related to the led configuration operation is included in the led code;

## 2. software SPI and hardware SPI description

The LCD 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. Arduino test program GPIO definition description

The GPIO definitions of the Arduino test program are placed separately in each application, This means that each application can flexibly define GPIOs as needed.

As shown below:

```
//paramters define
#define MODEL ST7735S128
#define CS    A5
#define CD    A3
#define RST   A4
#define SDA   11
#define SCK   13
#define LED   A0 //if you don't need
```

### 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. STM32 test program GPIO definition description

The STM32's LCD non-SPI GPIO definition is placed in lcd.h as shown below:

```
#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:

```
#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:

```
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.

## 4. SPI communication code implementation

### A. Arduino 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 the LCDWIKI\_SPI.cpp file of the LCDWIKI\_SPI library, as shown below:

```

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

```

⇒ 软件spi

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 and 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 lcd.c and spi.c respectively, and the software SPI implementation method is the same, 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..

## 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.