1.6inch SPI Module MSP1601 User Manual

Product Description

The LCD module uses a 4-wire SPI communication method with a driver IC of SSD1283A and a resolution of 130x130. The module contains an LCD display and backlight control circuitry.

Product Features

- 1.6-inch color screen,support 65K color display,display rich colors
- 130X130 resolution, clear display
- Semi-permeable trans-LCD, has superior performance and is clearly visible in strong light.
- Using the SPI serial bus, it only takes a few IOs to illuminate the display
- On-board 3.3V/5V power conversion chip compatible with 3.3V and 5V voltage levels
- The working temperature range is industrial grade (- 20 ℃ ~ 70 ℃)
- Provide a rich sample program
- Military-grade process standards, long-term stable work
- Provide underlying driver technical support
- Support backlight control

Product Parameters

Name	Description
Display Color	RGB 65K color
SKU	MSP1601
Screen Size	1.6(inch)
Type	TFT
Driver IC	SSD1283A
Resolution	130*130 (Pixel)
Module Interface	4-wire SPI interface

Active Area	28.86x28.86 (mm)
Module PCB Size	35.40x53.82 (mm)
Angle of view	>60°
Operating Temperature	-10℃~60℃
Storage Temperature	-20℃~70℃
Operating Voltage	3.3V / 5V
Power Consumption	TBD
Product Weight(including package)	15(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	**RST**	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 three parts: an LCD display control circuit, a Level conversion 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 level conversion circuit is used to control the level conversion from 5V to 3.3V

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

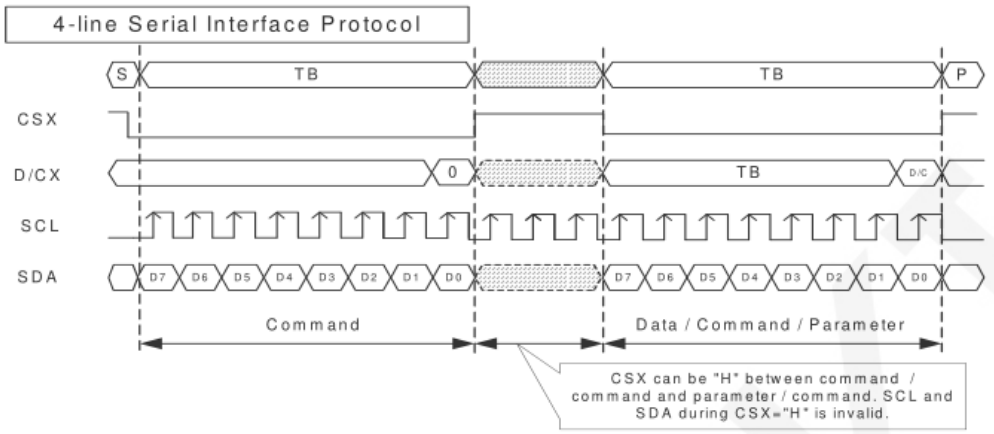
1. Introduction to SSD1283A Controller

The SSD1283A controller supports a maximum resolution of 132*132 and a 39204-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 SSD1283A 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 SSD1283A 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 SSD1283A 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:

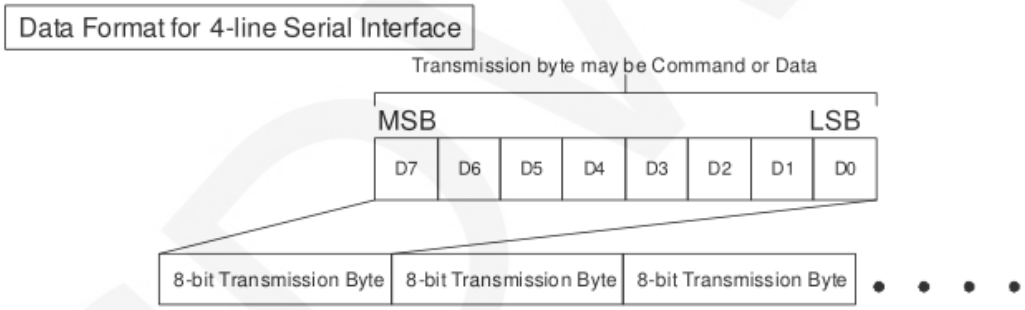


CSX is a slave chip select, and the chip is enabled only when CSX is low.

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. 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**	Software SPI:A1
		Hardware SPI:13
3	**SDA**	Software SPI:A2
		Hardware SPI:11
4	**A0**	9
5	**RST**	8
6	**CS**	10
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**	Software SPI:A1
		Hardware SPI:52
3	**SDA**	Software SPI:A2
		Hardware SPI:51
4	**A0**	9
5	**RST**	8
6	**CS**	10
7	**GND**	GND

8	VCC	5V/3.3V

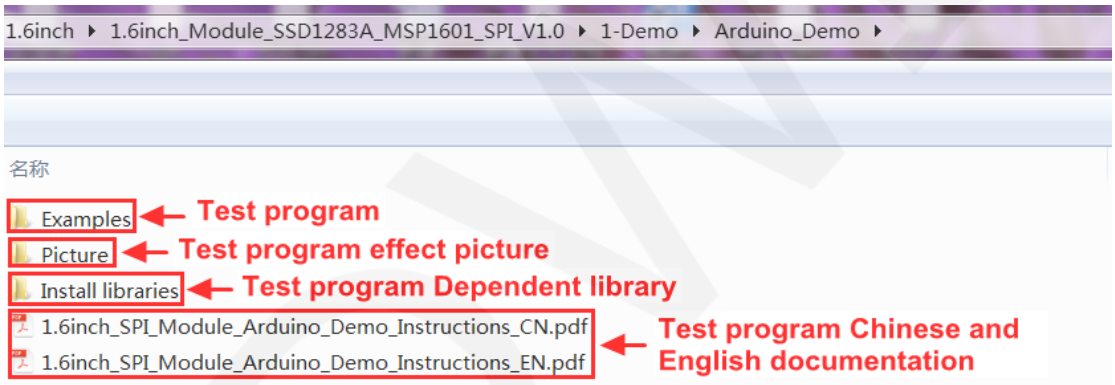
Operating Steps：

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

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

2. C51 instructions

Wiring instructions:

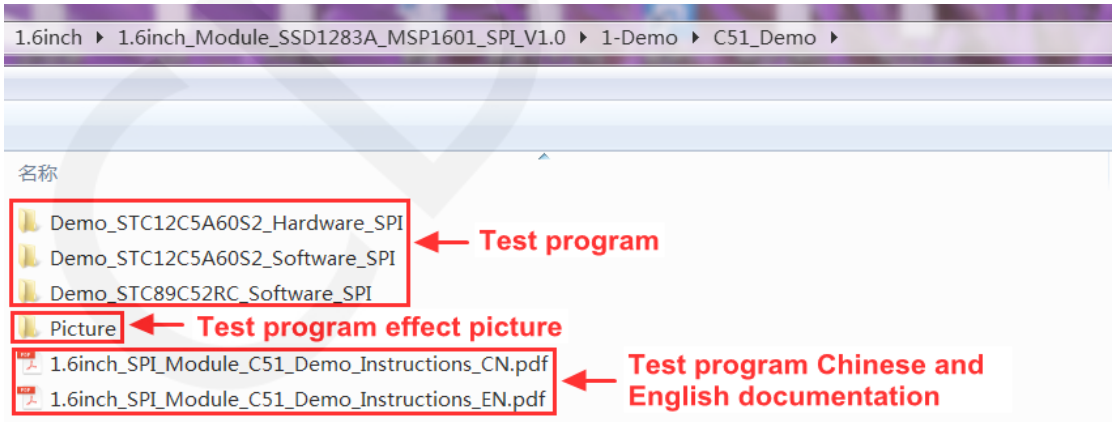
STC89C52RC and STC12C5A60S2 microcontroller test program wiring instructions		
Number	**Module Pin**	**Corresponding to STC89/STC12 development board wiring pin**
1	**VCC**	5V/3.3V
2	**GND**	GND
3	**CS**	P13
4	**RST**	P33
5	**A0**	P12
6	**SDA**	P15
7	**SCK**	P17
8	**LED**	P32

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

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.

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

STM32F103ZET6 microcontroller test program wiring instructions		
Number	**Module Pin**	**Corresponding to Elite STM32 development board wiring pin**
1	**VCC**	5V/3.3V
2	**GND**	GND
3	**CS**	PB11
4	**RST**	PB12

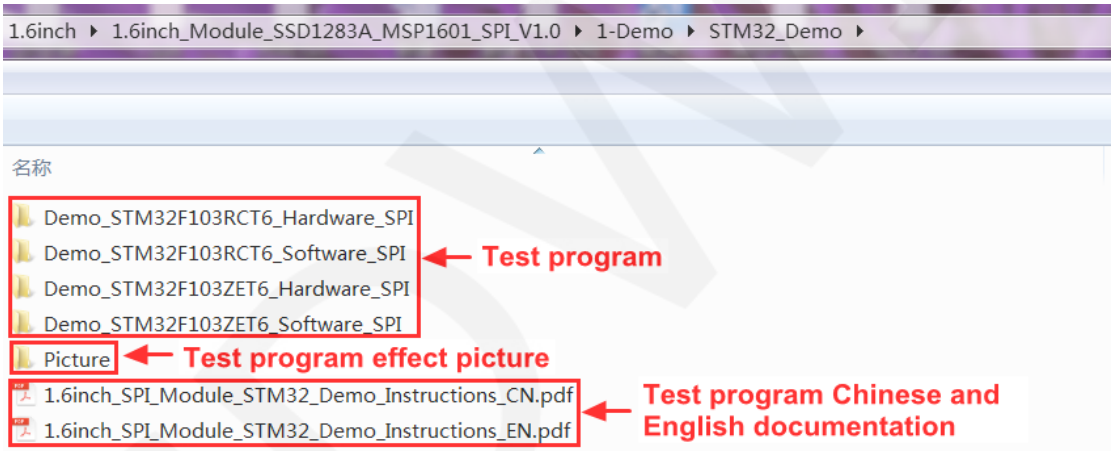
5	**A0**	PB10
6	**SDA**	PB15
7	**SCK**	PB13
8	**LED**	PB9

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

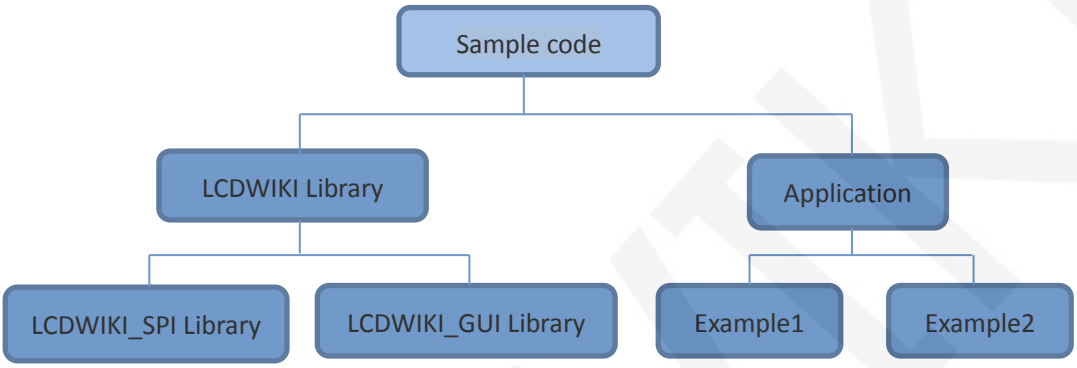
D. If the LCD module displays characters and graphics normally, the program runs successfully；

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.

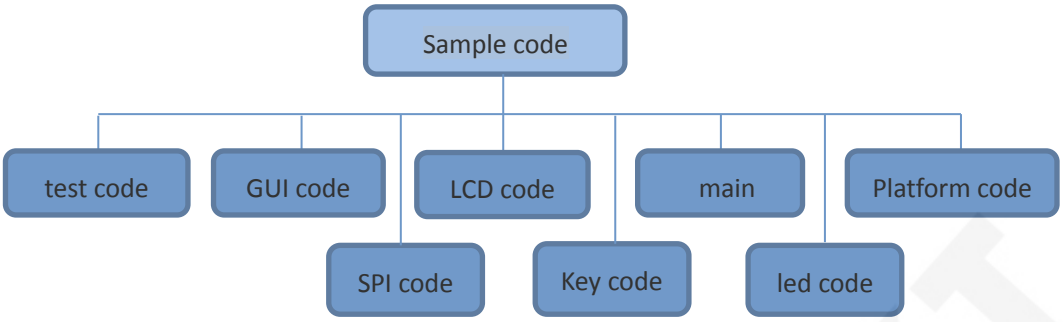
LCDWIKI_SPI is the underlying library, which is associated with hardware. It is mainly responsible for operating registers, including hardware module initialization, data and command transmission, pixel coordinates and color settings, and display mode configuration.

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 SSD1283A
#define CS   10
#define CD   9
#define SDA  A2   //if you use the hardware spi,this pin is no need to set
#define SCK  A1   //if you use the hardware spi,this pin is no need to set
#define RST  8
#define LED  A3   //if you don't need to control the LED pin,you should set it to -1 and set it to 3.3V
```

B. C51 test program GPIO definition description

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

```
sbit LCD_LED = P3^2;  //MCU_P32--->>TFT --BL
sbit LCD_RS  = P1^2;  //P12--->>TFT --RS/DC
sbit LCD_CS  = P1^3;  //MCU_P13--->>TFT --CS/CE
sbit LCD_RST = P3^3;  //P33--->>TFT --RST
sbit LCD_SCL = P1^7;  //P17--->>TFT --SCL/SCK
sbit LCD_SDA = P1^5;  //P15 MOSI--->>TFT --SDA/DIN

//液晶控制口置1操作语句宏定义
#define LCD_CS_SET    LCD_CS=1
#define LCD_RS_SET    LCD_RS=1
#define LCD_SDA_SET   LCD_SDA=1
#define LCD_SCL_SET   LCD_SCL=1
#define LCD_RST_SET   LCD_RST=1
#define LCD_LED_SET   LCD_LED=1

//液晶控制口置0操作语句宏定义
#define LCD_CS_CLR    LCD_CS=0
#define LCD_RS_CLR    LCD_RS=0
#define LCD_SDA_CLR   LCD_SDA=0
#define LCD_SCL_CLR   LCD_SCL=0
#define LCD_RST_CLR   LCD_RST=0
#define LCD_LED_CLR   LCD_LED=0
```

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_Driver.h as shown below:

```
#define LCD_CTRL        GPIOB    //定义TFT数据端口
#define LCD_LED         GPIO_Pin_9  //MCU_PB9--->>TFT --BL
#define LCD_RS          GPIO_Pin_10 //PB11--->>TFT --RS/DC
#define LCD_CS          GPIO_Pin_11 //MCU_PB11--->>TFT --CS/CE
#define LCD_RST         GPIO_Pin_12 //PB10--->>TFT --RST
#define LCD_SCL         GPIO_Pin_13 //PB13--->>TFT --SCL/SCK
#define LCD_SDA         GPIO_Pin_15 //PB15 MOSI--->>TFT --SDA/DIN

//液晶控制口置1操作语句宏定义
#define LCD_CS_SET    LCD_CTRL->BSRR=LCD_CS
#define LCD_RS_SET    LCD_CTRL->BSRR=LCD_RS
#define LCD_SDA_SET   LCD_CTRL->BSRR=LCD_SDA
#define LCD_SCL_SET   LCD_CTRL->BSRR=LCD_SCL
#define LCD_RST_SET   LCD_CTRL->BSRR=LCD_RST
#define LCD_LED_SET   LCD_CTRL->BSRR=LCD_LED

//液晶控制口置0操作语句宏定义
#define LCD_CS_CLR    LCD_CTRL->BRR=LCD_CS
#define LCD_RS_CLR    LCD_CTRL->BRR=LCD_RS
#define LCD_SDA_CLR   LCD_CTRL->BRR=LCD_SDA
#define LCD_SCL_CLR   LCD_CTRL->BRR=LCD_SCL
#define LCD_RST_CLR   LCD_CTRL->BRR=LCD_RST
#define LCD_LED_CLR   LCD_CTRL->BRR=LCD_LED
```

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

If the hardware SPI,LCD_LED,LCD_RS,LCD_CS and LCD_RST pin definitions can be modified, they can be defined as any other idle GPIO. LCD_SCL and LCD_SDA do not need to be defined, LCD_SDA_SET, LCD_SDA_CRL,LCD_SCL_SET, and LCD_SCL_CLR do not need to be defined, and LCD_SCL and LCD_SDA initialization needs to be removed from the LCD_GPIO_Init function in the Lcd_Driver.c file, as shown in the following figure:

```
void LCD_GPIO_Init(void)
{
  GPIO_InitTypeDef  GPIO_InitStructure;
  RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOB ,ENABLE);
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9| GPIO_Pin_10| GPIO_Pin_11| GPIO_Pin_12 | GPIO_Pin_13 | GPIO_Pin_15;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_Init(GPIOB, &GPIO_InitStructure);
}
```

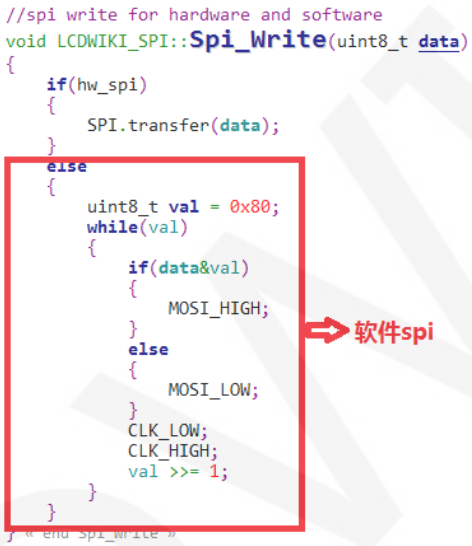
The contents of the red circle must be removed.

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:



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)

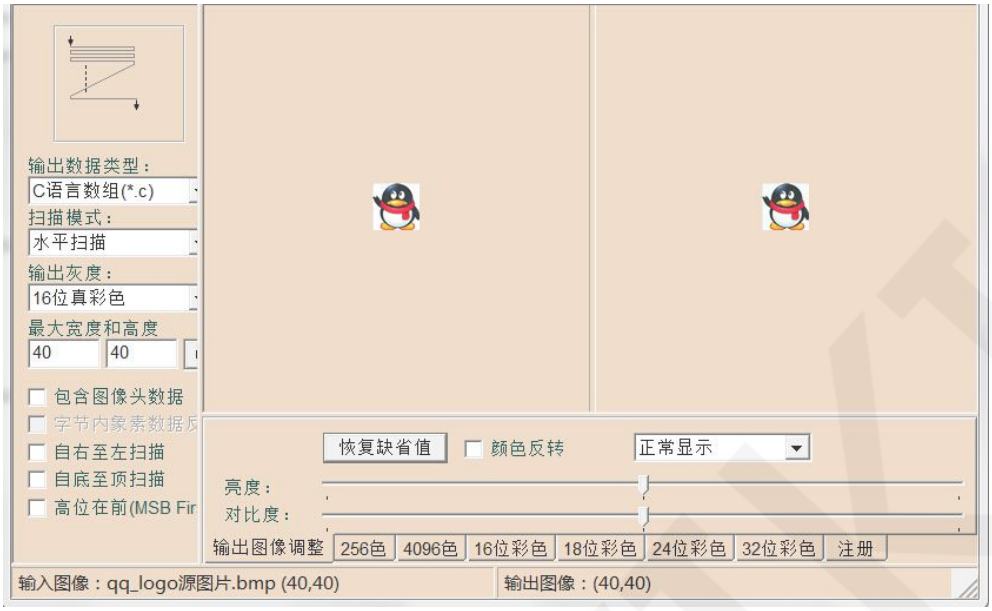
Output number system selects hexadecimal number

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.