1.6inch SPI Module MSP1601 用户手册

产品概述

该款 LCD 模块采用 4 线制 SPI 通信方式，驱动 IC 为 SSD1283A，分辨率为 130x130。该模块包含有 LCD 显示屏，背光控制电路。

产品特点

- 1.6 寸彩屏，支持 RGB 65K 色显示，显示色彩丰富
- 130X130 分辨率，显示清晰
- 半透反式 LCD，性能优越，强光下显示清晰可见
- 采用 SPI 串行总线，只需几个 IO 即可点亮显示
- 板载 3.3V/5V 电源转换芯片，兼容 3.3V 和 5V 电压等级
- 工作温度范围为工业级(-20℃~70℃)
- 提供丰富的示例程序
- 军工级工艺标准,长期稳定工作
- 提供底层驱动技术支持
- 支持背光控制

产品参数

名称	描述
显示颜色	RGB 65K 彩色
SKU	MSP1601
尺寸	1.6(inch)
类型	TFT
驱动芯片	SSD1283A
分辨率	130*130 (Pixel)
模块接口	4-wire SPI interface
有效显示区域	28.86x28.86 (mm)
模块尺寸	35.40x53.82 (mm)
视角	>60°

工作温度	-10℃~60℃
存储温度	-20℃~70℃
工作电压	3.3V / 5V
功耗	待定
产品重量（含包装）	15(g)

接口说明

引脚丝印图

标号	模块引脚	引脚说明
1	VCC	LCD 电源正(3.3V~5V)
2	GND	LCD 电源地
3	CS	LCD 片选信号
4	RST	LCD 复位信号
5	A0	LCD 寄存器/数据选择信号
6	SDA	LCD SPI 总线写数据信号

7	SCK	LCD SPI 总线时钟信号
8	LED	背光控制信号（高电平点亮，如不需要控制，请接 3.3V）

硬件配置

该 LCD 模块硬件电路包含两大部分：LCD 显示控制电路、电平转换控制电路以及背光控制电路。

LCD 显示控制电路用于控制 LCD 的引脚，包括控制引脚和数据传输引脚。

电平转换电路用于控制电平从 5V 到 3.3V 转换。

背光控制电路用于用于控制背光亮和灭，当然如果不需要控制背光，可以不使用该电路，直接将背光控制引脚接到 3.3V 电源上。

工作原理

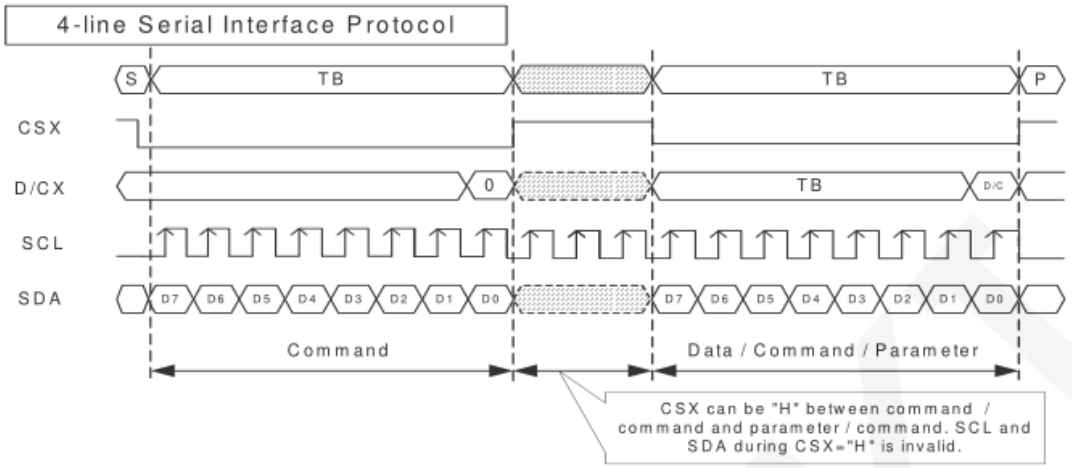
1、SSD1283A 控制器简介

SSD1283A 控制器支持的最大分辨率为 132*132，拥有一个 39204 字节大小的 GRAM。同时支持 8 位、9 位、16 位、18 位并口数据总线，还支持 3 线制和 4 线制 SPI 串口。由于并行控制需要大量的 IO 口，所以最常用的还是 SPI 串口控制。SSD1283A 还支持 65K、262K RGB 颜色显示，显示色彩很丰富，同时支持旋转和滚动显示以及视频播放，显示方式多样。

SSD1283A 控制器使用 16bit（RGB565）来控制一个像素点显示，因此可以每个像素显示颜色多达 65K 种。像素点地址设置按照行列的顺序进行，递增递减方向由扫描方式决定。SSD1283A 显示方法按照先设置地址再设置颜色值进行。

2、SPI 通信协议简介

4 线制 SPI 总线写模式时序如下图所示：

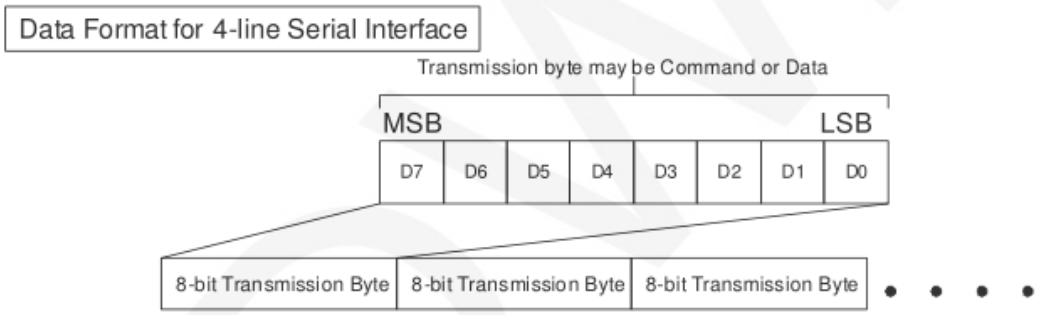


CSX 为从机片选， 仅当 CSX 为低电平时，芯片才会被使能。

D/CX 为芯片的数据/命令控制引脚，当 DCX 为低电平时写命令，为高电平时写数据

SCL 为 SPI 总线时钟，每个上升沿传输 1bit 数据；

SDA 为 SPI 传输的数据，一次传输 8bit 数据，数据格式如下图所示：



高位在前，先传输。

对于 SPI 通信而言，数据是有传输时序的，即时钟相位（CPHA）与时钟极性(CPOL)的组合：

CPOL 的高低决定串行同步时钟的空闲状态电平，CPOL = 0，为低电平。CPOL 对传输协议没有很多的影响；

CPHA 的高低决定串行同步时钟是在第一时钟跳变沿还是第二个时钟跳变沿数据被采集，当 CPHL = 0，在第一个跳变沿进行数据采集；

这两者组合就成为四种 SPI 通信方式，国内通常使用 SPI0,即 CPHL = 0，CPOL = 0

使用说明

1、Arduino 使用说明

接线说明：

Arduino UNO单片机测试程序接线说明		
序号	**模块引脚**	**对应UNO开发板接线引脚**
1	LED	A0
2	SCK	软件SPI：A1
		硬件SPI：13
3	SDA	软件SPI：A2
		硬件SPI：11
4	A0	9
5	RST	8
6	CS	10
7	GND	GND
8	VCC	5V/3.3V

Arduino MEGA2560单片机测试程序接线说明		
序号	**模块引脚**	**对应MEGA2560开发板接线引脚**
1	LED	A0
2	SCK	软件SPI：A1
		硬件SPI：52
3	SDA	软件SPI：A2
		硬件SPI：51
4	A0	9
5	RST	8
6	CS	10
7	GND	GND
8	VCC	5V/3.3V

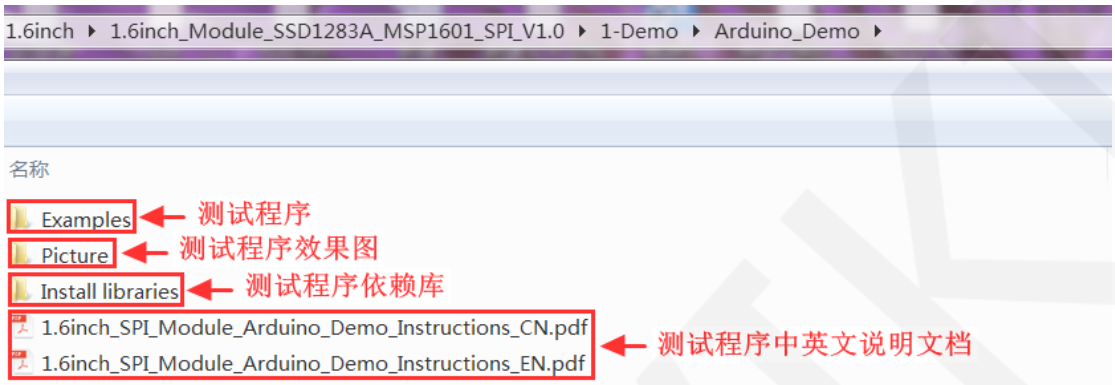
操作步骤：

A、按照上述接线说明将 LCD 模块和 Arduino 单片机连接起来，并上电；

B、将测试程序目录中 Install libraries 目录下的依赖库拷贝到 Arduino 工程目录的 libraries 文件夹下（如果不需要依赖库，则不需要拷贝）；

C、打开 Arduino 测试程序所在目录，选择需要测试的示例，如下图所示：

（测试程序说明请查阅测试程序包中测试程序说明文档）



D、打开所选的示例工程，进行编译和下载。

关于 Arduino 测试程序依赖库拷贝、编译和下载的具体操作方法见如下文档：

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

E、LCD 模块如果正常显示字符和图形，则说明程序运行成功；

2、C51 使用说明

接线说明：

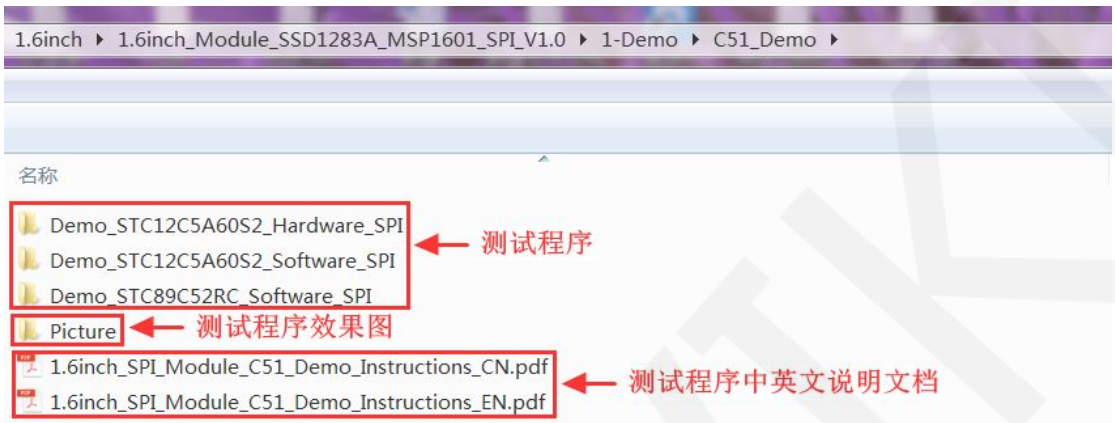
STC89C52RC和STC12C5A60S2单片机测试程序接线说明		
序号	**模块引脚**	**对应STC89/STC12开发板接线引脚**
1	**VCC**	5V/3.3V
2	**GND**	GND
3	**CS**	P13
4	**RST**	P33
5	**A0**	P12
6	**SDA**	P15
7	**SCK**	P17
8	**LED**	P32

操作步骤：

A、 按照上述接线说明将 LCD 模块和 C51 单片机连接起来，并上电；

B、 选择需要测试的 C51 测试程序，如下图所示：

（测试程序说明请查阅测试程序包中测试程序说明文档）



C、打开所选的测试程序工程，进行编译和下载；

关于 C51 测试程序编译和下载的详细说明见如下文档：

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

D、LCD 模块如果正常显示字符和图形，则说明程序运行成功；

3、STM32 使用说明

接线说明：

由于不同的开发板引脚位置不一样，而且预留外接的引脚也不一样（有些开发板没有将需要的引脚外接），为了方便接线，所以每种开发板的接线引脚不一致。

STM32F103RCT6单片机测试程序接线说明		
序号	**模块引脚**	**对应MiniSTM32开发板接线引脚**
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

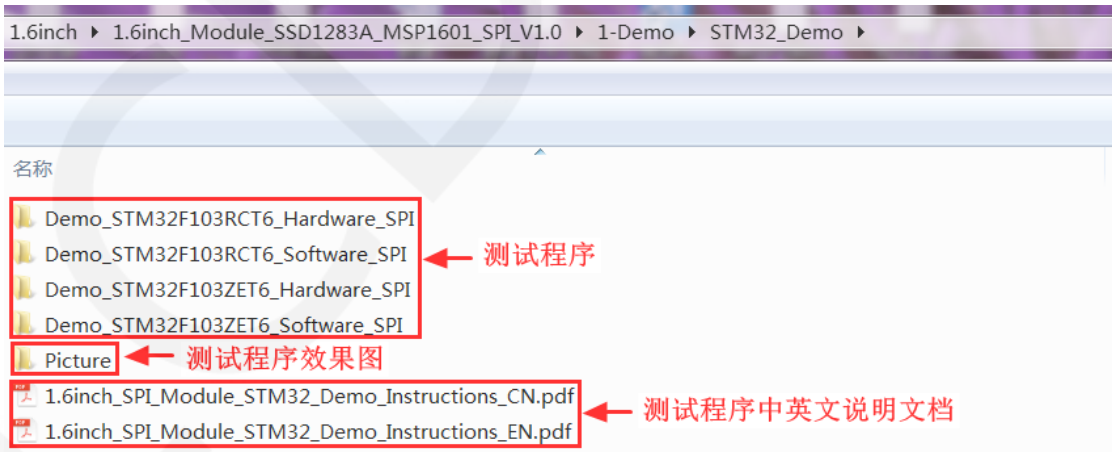
STM32F103ZET6单片机测试程序接线说明		
序号	引脚丝印	对应Elite STM32开发板接线引脚
1	VCC	5V/3.3V
2	GND	GND
3	CS	PB11
4	RST	PB12
5	A0	PB10
6	SDA	PB15
7	SCK	PB13
8	LED	PB9

操作步骤：

A、 按照上述接线说明将 LCD 模块和 STM32 单片机连接起来，并上电；

B、 根据单片机型号选择测试示例，如下图所示：

（测试程序说明请查阅测试程序包中测试程序说明文档）



C、打开所选的测试程序工程，进行编译和下载；

关于 STM32 测试程序编译和下载的详细说明见如下文档：

http://www.lcdwiki.com/res/PublicFile/STM32_Keil_Use_Illustration_CN.pdf

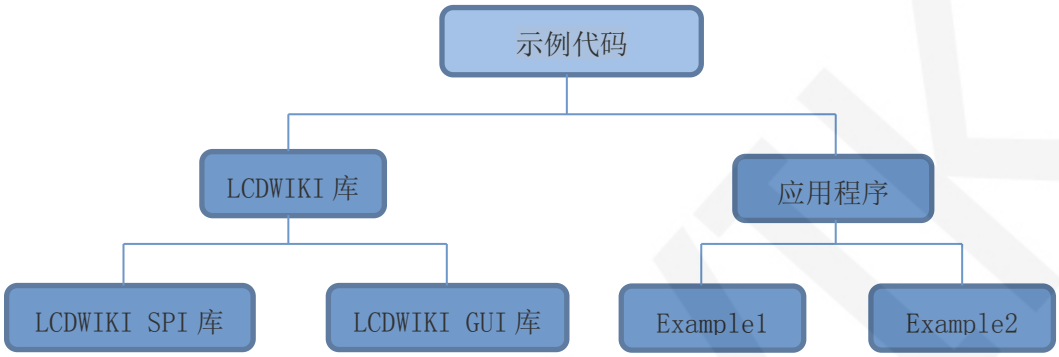
D、LCD 模块如果正常显示字符和图形，则说明程序运行成功；

软件说明

1、代码架构

A、 Arduino 代码架构说明

代码架构如下图所示：



Arduino 的测试程序代码由两部分组成：LCDWIKI 库和应用代码。

LCDWIKI 库包含两部分内容：LCDWIKI_SPI 库、LCDWIKI_GUI 库。

应用程序包含几个测试示例，每个测试示例包含不同的测试内容。

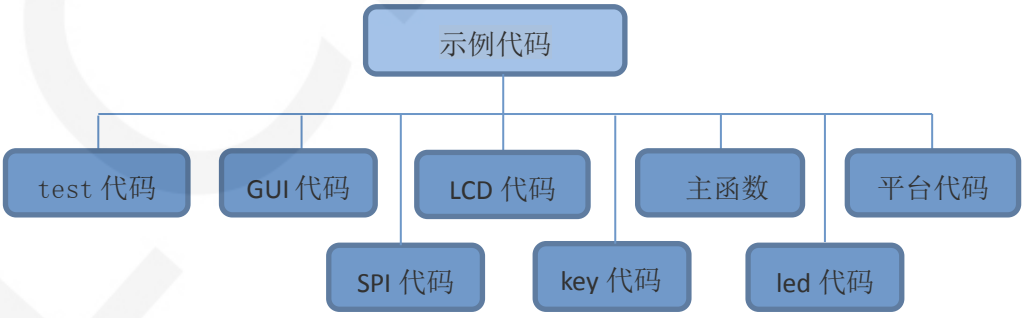
LCDWIKI_SPI 为底层库，和硬件有关联，主要负责操作寄存器，包括硬件模块初始化，数据和命令传输，像素点坐标和颜色设置，显示方式配置等。

LCDWIKI_GUI 为中间层库，负责使用底层库提供的 API 实现图形的绘制，字符显示。

应用程序是利用 LCDWIKI 库提供的 API，编写一些测试示例，实现某方面的测试功能。

B、C51 和 STM32 代码架构说明

代码架构如下图所示：



主程序运行时的 Demo API 代码包含在 test 代码中；

LCD 初始化以及相关的操作都包含在 LCD 代码中；

画点、线、图形以及中英文字符显示相关的操作都包含在 GUI 代码中；

主函数实现应用程序运行；

平台代码因平台而异；

SPI 初始化及配置相关的操作包含在 SPI 代码中；

按键处理相关的代码都包含在 key 代码中(C51 平台没有按键处理代码)；

led 配置操作相关的代码都包含在 led 代码中；

2、软件 SPI 和硬件 SPI 说明

该 LCD 模块分别提供了软件 SPI 和硬件 SPI 示例代码（STC89C52RC 除外，因为其没有硬件 SPI 功能），两台示例代码在显示内容上没任何区别，但是如下方面有区别：

A、显示速度

硬件 SPI 明显比软件 SPI 要快，这是由硬件决定的。

B、 GPIO 定义

软件 SPI 全部控制引脚都要定义，可以使用任何空闲引脚，硬件 SPI 的数据和时钟信号引脚是固定的（因平台而异），其他控制引脚要自己定义，也可使用任何空闲引脚。

C、初始化

软件 SPI 初始化时，只需要对用于引脚定义的 GPIO 进行初始化（C51 平台不需要），硬件 SPI 初始化时，需要对相关的控制寄存器以及数据寄存器进行初始化。

3、模块 GPIO 定义说明

A、Arduino 测试程序 GPIO 定义说明

Arduino 测试程序的 GPIO 定义都单独放在每个应用程序里，也就是说每个应用程序可以根据需求灵活定义 GPIO。如下图所示：

```
//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 测试程序 GPIO 定义说明

C51 的液晶屏相关的 GPIO 定义放在 lcd.h 文件里面，如下图所示：

```
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
```

如果使用软件 SPI，所有引脚定义都可以修改，可以定义成其他任何空闲的 GPIO。

如果使用硬件 SPI，LCD_BL、LCD_RS、LCD_CS 以及 LCD_RST 引脚定义可以修改，可以定义成其他任何空闲的 GPIO。LCD_CLK 和 LCD_SDI 不需要定义。

C、STM32 测试程序 GPIO 定义说明

STM32 的 GPIO 定义放在 Lcd_Driver.h 里面，如下图所示：

```
#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
```

如果使用软件 SPI，所有引脚定义都可以修改，可以定义成其他任何空闲的 GPIO。

如果使用硬件 SPI，LCD_LED、LCD_RS、LCD_CS 以及 LCD_RST 引脚定义可以修改，可以定义成其他任何空闲的 GPIO。LCD_SCL 和 LCD_SDA 不需要定义，LCD_SDA_SET 、LCD_SDA_CRL、LCD_SCL_SET 以及 LCD_SCL_CLR 也不需要定义，同时需要在 Lcd_Driver.c 文件里面的 LCD_GPIO_Init 函数里将 LCD_SCL 和 LCD_SDA 初始化去掉，如下图所示：

```
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);
}
```

红圈里面的内容都要去掉。

4、SPI 通信代码实现

A、Arduino 测试程序 SPI 通信代码实现

硬件 SPI 通信都是由系统实现好的，我们只需要操作寄存器，调用相关函数就可以了，具体说明请查阅 MCU 相关的说明文档。

软件 SPI 通信代码在 LCDWIKI_SPI 库的 LCDWIKI_SPI.cpp 文件里实现，如下图所示：

```
//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;
        }
    }
} // end spi_write //
```

软件spi

传输的数据位为 1，则将 SPI 数据引脚拉高，为 0，则将 SPI 数据引脚拉低，每次传输一个字节数据，高位在前，每个时钟上升沿传输 1 位数据。

B、C51 和 STM32 测试程序 SPI 通信代码实现

硬件 SPI 通信都是由系统实现好的，我们只需要操作寄存器，调用相关函数就可以了，具体说明请查阅 MCU 相关的说明文档。

软件 SPI 通信代码分别在 lcd.c 和 spi.c 中实现，软件 SPI 实现方法一样，如下图所示：

```
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;
  }
}
```

传输的数据位为 1，则将 SPI 数据引脚拉高，为 0，则将 SPI 数据引脚拉低，每次传输一个字节数据，高位在前，每个时钟上升沿传输 1 位数据。

常用软件

本套测试示例需要显示中英文、符号以及图片，所以要用到取模软件。取模软件有两种：Image2Lcd 和 PCtoLCD2002。这里只针对该套测试程序说明一下取模软件的设置。

PCtoLCD2002 取模软件设置如下：

点阵格式选择**阴码**

取模方式选择**逐行式**

取模走向选择**顺向（高位在前）**

输出数制选择**十六进制数**

自定义格式选择 **C51 格式**

具体设置方法见如下网页：

http://www.lcdwiki.com/zh/%E3%80%90%E6%95%99%E7%A8%8B%E3%80%91%E4%B8%AD%E8%8B%B1%E6%96%87%E6%98%BE%E7%A4%BA%E5%8F%96%E6%A8%A1%E8%AE%BE%E7%BD%AE

Image2Lcd 取模软件设置如下图所示：

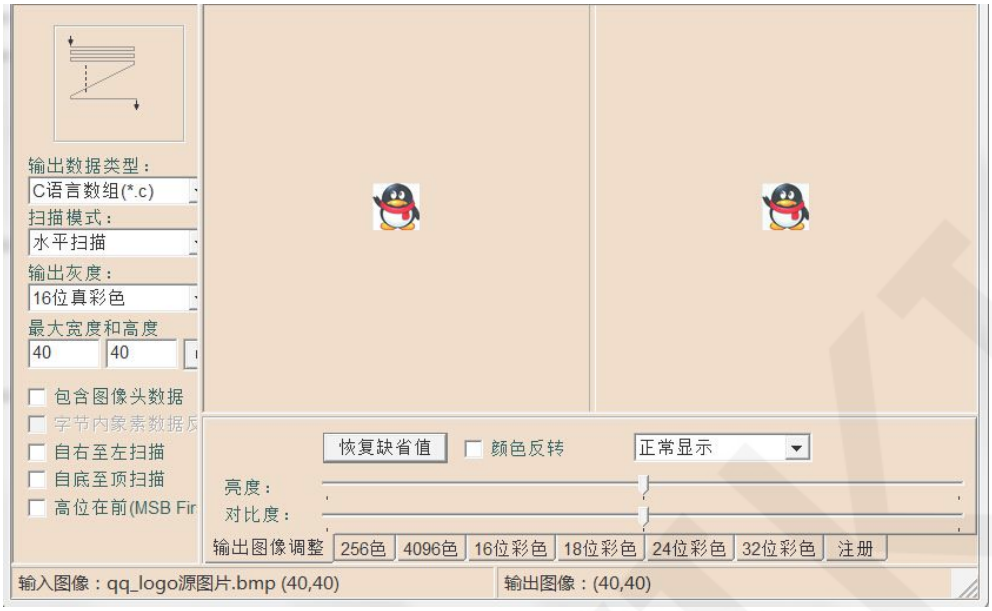


Image2Lcd 软件需要设置为水平、自左向右、自上向下、低位在前扫描方式。