1.6inch SPI Module
用户手册

产品概述

1.6 寸 LCD 模块是一种半透反式 TFT LCD 模块。该模块包含有 1.6 寸 LCD 显示屏，5V~3.3V 电源转换电路以及背光控制电路。

产品特点

- 1.6 寸彩屏，支持 65K 色显示，显示色彩丰富
- 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 口即可点亮显示
- 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℃)
- 军工级工艺标准,长期稳定工作
- 提供丰富的多平台例程
- 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	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	35.40X53.82X1.6(mm)
视角	>60°
工作温度	-20℃~70℃

存储温度	-30℃~80℃
工作电压	3.3V / 5V
功耗	全亮约为 100mA，全灭约为 10mA。
产品重量	14(g)

接口说明

标号	PIN	引脚说明
1	GND	LCD 电源地
2	VCC	LCD 电源正(3.3V~5V)
3	CS	LCD 片选信号
4	RESET	LCD 复位信号
5	A0/DC	LCD 寄存器/数据选择信号
6	SDA	SPI 总线写数据信号
7	SCK	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 1.6 寸 LCD 模块硬件电路包含三大部分：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

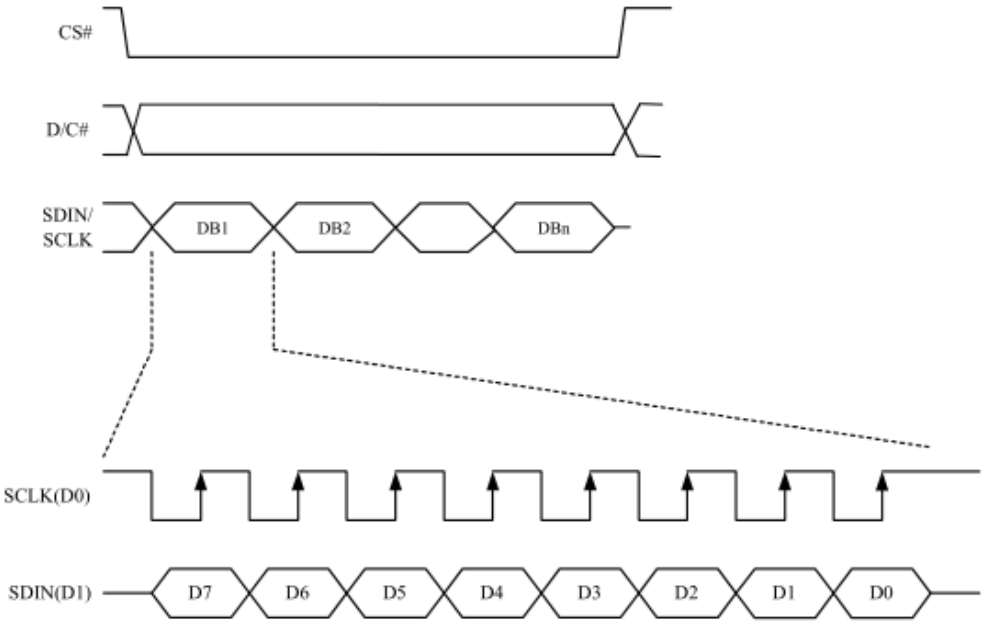
该 LCD 模块的控制 IC 为 SSD1283A。SSD1283A 控制器支持的最大分辨率为 132*132，同时支持 8 位、9 位、16 位以及 18 位 6800 和 8080 并口，还支持 SPI 串口。由于并行控制会浪费 IO 口，所以最常用的还是 SPI 串口控制。SSD1283A 还支持 65K 和 262K RGB 颜色显示，

显示色彩很丰富，同时支持旋转显示和滚动显示，显示方式多样。

SSD1283A 控制器使用 16bit（RGB565）来控制一个像素点显示，因此可以每个像素点显示颜色多达 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

SPI 总线写模式数据格式如下图所示：



CS#为从机片选， 仅当 CS 为低电平时，芯片才会被使能。

D/C#为芯片的数据/命令控制引脚，当 DC = 0 时写命令，当 DC = 1 时写数据

SDIN 为传输的数据，即 16 位灰度图片；

SCLK 为 SPI 通信时钟。

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

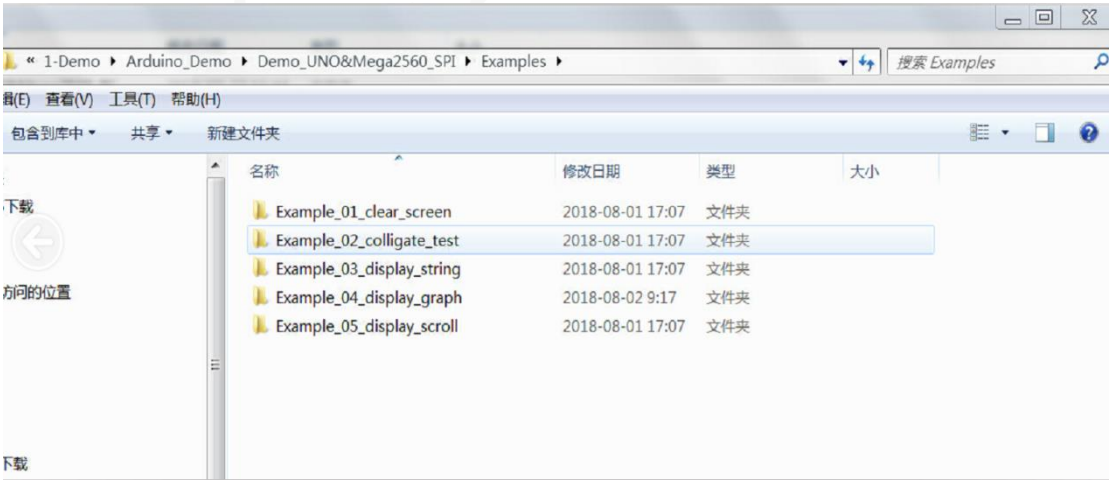
接线说明：

模块引脚	对应 UNO 单片机接线引脚	对应 Mega2560 单片机接线引脚
VCC	5V/3.3V	5V/3.3V
GND	GND	GND
CS	10	10
RESET	8	8
AO/DC	9	9
SDA	硬件 SPI：11 软件模拟 SPI：A2	硬件 SPI：51 软件模拟 SPI：A2
SCK	硬件 SPI：13 软件模拟 SPI：A1	硬件 SPI：52 软件模拟 SPI：A1
LED	A3 （如不需要控制，请接 3.3V）	A3 （如不需要控制，请接 3.3V）

操作步骤：

A、按照上述接线说明，将 1.6 寸 LCD 模块和单片机连接起来，然后上电；

B、打开“1-Demo\Arduino_Demo\Demo_UNO&Mega2560_SPI\Examples”目录，选择想要测试的示例，如下图所示：



C、使用 Arduino IDE 软件对示例进行编译和下载（Arduino IDE 软件具体使用方法见 Arduino_IDE_Use_Illustration_CN.pdf）；

D、下载成功后，1.6 寸 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

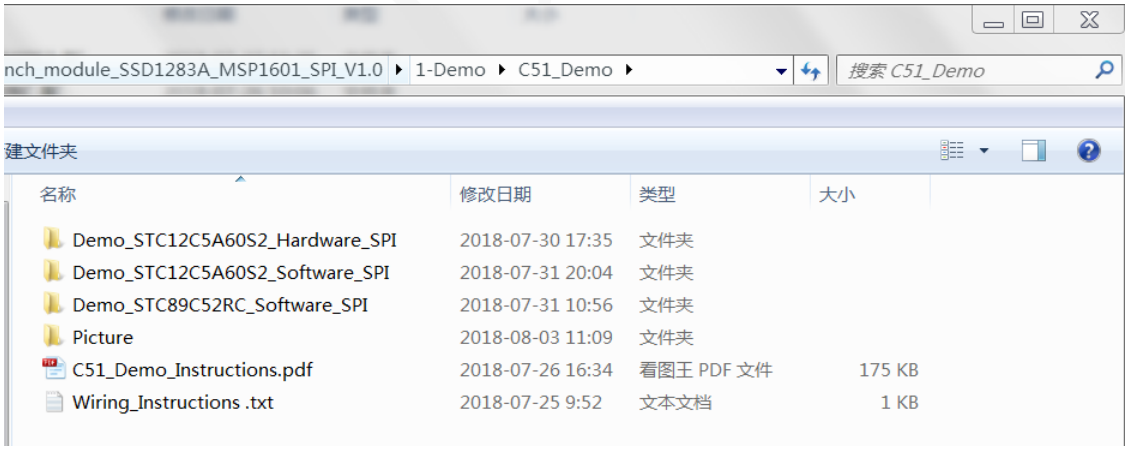
接线说明：

OLED 模块引脚	对应 Raspberry Pi 接线引脚
VCC	5V/3.3V
GND	GND
CS	P13
RESET	P33
AO/DC	P12
SDA	P15
SCK	P17
LED	P32（如果不需要控制请接3.3V）

操作步骤：

A、按照上述接线说明将 1.6 寸 LCD 模块和 C51 单片机连接起来，并上电；

B、打开“1-Demo\Demo_C51”目录，根据单片机型号选择测试示例，如下图所示：



C、打开所选的示例工程，进行编译和下载（C51keil 具体操作方法见 C51_Keil&stc-isp_Use_Illustration_CN.pdf）；

D、1.6 寸 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

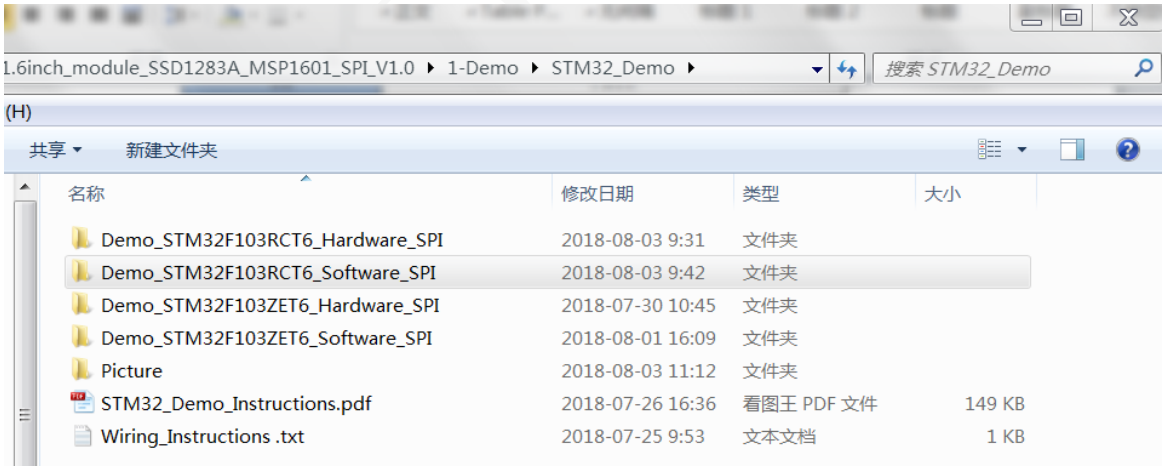
接线说明：

OLED 模块引脚	对应 Raspberry Pi 接线引脚
VCC	5V/3.3V
GND	GND
CS	PB11
RESET	PB12
AO/DC	PB10
SDA	PB15
SCK	PB13
LED	PB9（如果不需要控制请接3.3V）

操作说明：

A、 按照上述接线说明将 1.6 寸 LCD 模块和 STM32 单片机连接起来，并上电；

B、 打开“1-Demo\Demo_STM32”目录，根据单片机型号选择测试示例，如下图所示：



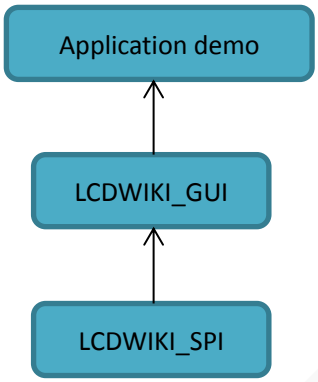
C、 打开所选的示例工程，进行编译和下载（STM32keil 具体操作方法见 STM32_Keil_Use_Illustration_CN.pdf）；

D、1.6 寸 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 测试示例需要依赖 LCDWIKI_SPI 库和 LCDWIKI_GUI 库，代码架构如下：

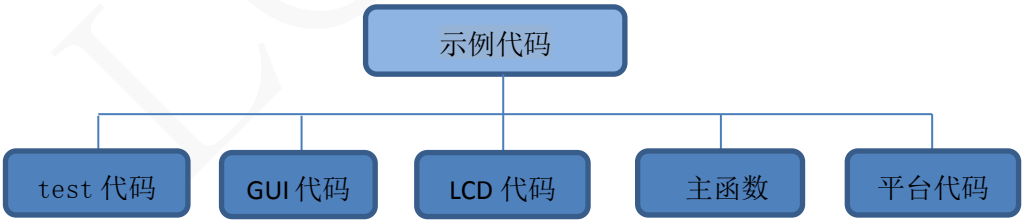


LCDWIKI_SPI 为底层库，操作 GPIO，写数据和命令，读数据以及初始化都是由它完成。

LCDWIKI_GUI 为应用层库，画点、画线、画矩形、画圆形、画三角形、图形填充以及文字和字符显示都是由它完成。

Application demo 为应用示例，就是利用上述两个库提供的 API 来完成一些图形和文字显示。

C、 C51 和 STM32 测试示例代码架构如下：



主程序运行时的 Demo API 代码包含在 test 代码中；

SPI 和 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；

2、软件 SPI 和硬件 SPI 说明

1.6inch 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 定软件

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，也可以使用任何空闲引脚。

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 定义修改

A、Arduino 的 GPIO 定义直接在示例程序里面修改，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
```

MODE 为模块控制 IC 的型号，不需要修改

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，SDA 和 SCK 可以不定义（系统硬件已经定义好了，如果已经定义也不起作用），其他引脚定义都可以修改，可以定义成其他任何空闲的 GPIO。

B、 C51 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_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。

C、 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);
}
```

将 GPIO_Pin_13 和 GPIO_Pin_15 去掉。

4、SPI 通信代码实现

A、Arduino 的 SPI 通信代码在 LCDWIKI_SPI 库里面，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 和硬件 SPI 选择通过 hw_spi 的值来完成。硬件 SPI 数据传输使用 SPI 库。软件 SPI 数据传输则通过软件模拟，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 的 SPI 通信代码分别在 LCD.c 和 Lcd_Driver.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。

STM32 和 C51 的硬件 SPI 的通信实现方法是一样的，都是通过操作 SPI 控制寄存器和数据寄存器，具体可以查看平台相关的示例。

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。（具体使用方法见 PCtoLCD2002_Use_Illustration_CN.pdf、Image2Lcd_Use_Illustration_CN.pdf）

1、Image2Lcd 取模软件设置

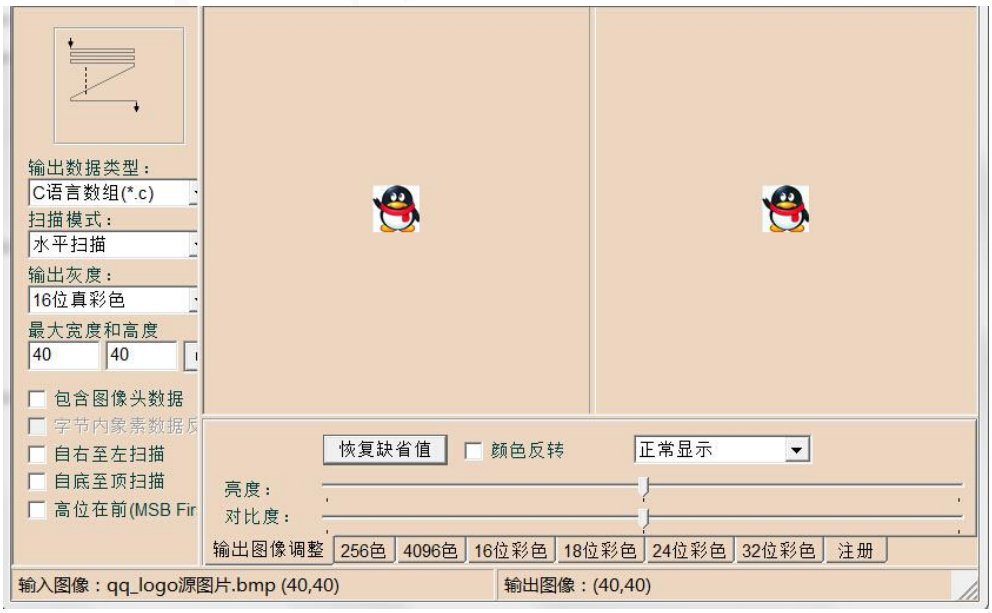
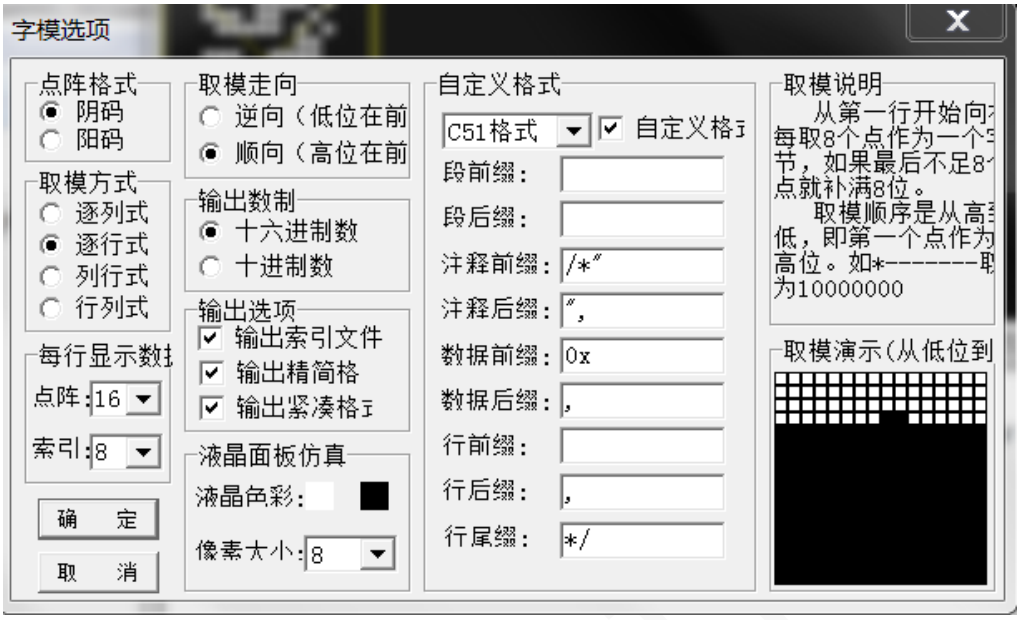


Image2Lcd 软件需要设置为水平、自左向右、自上向下、低位在前扫描方式。

2、PCtoLCD2002 取模软件设置



PCtoLCD2002 软件需要设置位逐行式、顺向扫描方式。