

1.8inch Arduino SPI Module MAR1801 用户手册

产品概述

该产品是一款 1.8 寸 LCD 显示模块。其拥有 128x160 分辨率，支持 16BIT RGB 65K 彩色显示，内部驱动 IC 为 ST7735S，采用 4 线制 SPI 通信方式。该模块包含有 LCD 显示屏和 PCB 控制底板。

产品特点

- 1.8 寸彩屏，支持 16BIT RGB 65K 彩色显示，显示色彩丰富
- 128X160 分辨率，显示清晰
- 采用 SPI 串行总线，只需几个 IO 即可点亮显示
- 带 SD 卡槽方便功能扩展
- 提供底层库和丰富的 Arduino、C51 以及 STM32 平台示例程序
- 军工级工艺标准,长期稳定工作
- 提供底层驱动技术支持

产品参数

名称	描述
显示颜色	16BIT RGB 65K 彩色
SKU	MAR1801
尺寸	1.8(inch)
类型	TFT
驱动芯片	ST7735S
分辨率	128*160 (Pixel)
模块接口	4-wire SPI interface
背光	2 White Led
有效显示区域	28.03x35.04 (mm)
模块尺寸	38.30x62.48 (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	约为 90mA
产品重量	约为 25(g)

接口说明

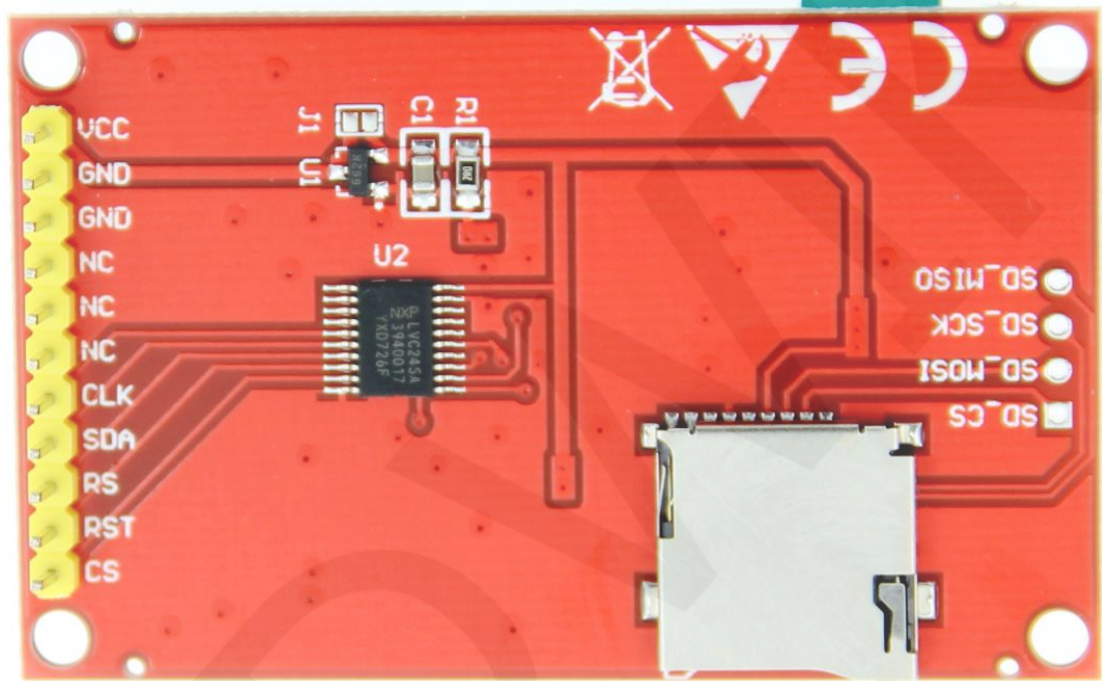


图1. 引脚丝印图

标号	模块引脚	引脚说明
1	VCC	液晶屏电源正极引脚
2	GND	液晶屏电源地引脚
3	GND	液晶屏电源地引脚
4	NC	没定义，保留
5	NC	没定义，保留
6	NC	没定义，保留
7	CLK	液晶屏SPI总线时钟引脚

8	SDA	液晶屏SPI总线写数据引脚
9	RS	液晶屏数据/命令选择控制引脚 (低电平：命令；高电平：数据)
10	RST	液晶屏复位控制引脚（低电平复位）
11	CS	液晶屏片选控制引脚（低电平使能）
12	SD_CS	SD卡SPI总线片选引脚（SD卡功能扩展应用）
13	SD_MOSI	SD卡SPI总线写数据引脚（SD卡功能扩展应用）
14	SD_SCK	SD卡SPI总线时钟引脚（SD卡功能扩展应用）
15	SD_MISO	SD卡SPI总线读数据引脚（SD卡功能扩展应用）

说明：

- 1、手动接线时，按如下方法可以减少占用开发板IO口：
 - A、不作SPI复用片选时，将模块**CS**引脚接地，节省1个IO口；
 - B、将模块**RST**引脚接单片机复位端，节省1个IO口；
- 2、短路PCB底板上的J1焊盘，则**VCC**此时使用3.3V电压接入，千万不能再接5V，会烧毁；

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转换，将输入 5V 转换为 3.3V 输出。

工作原理

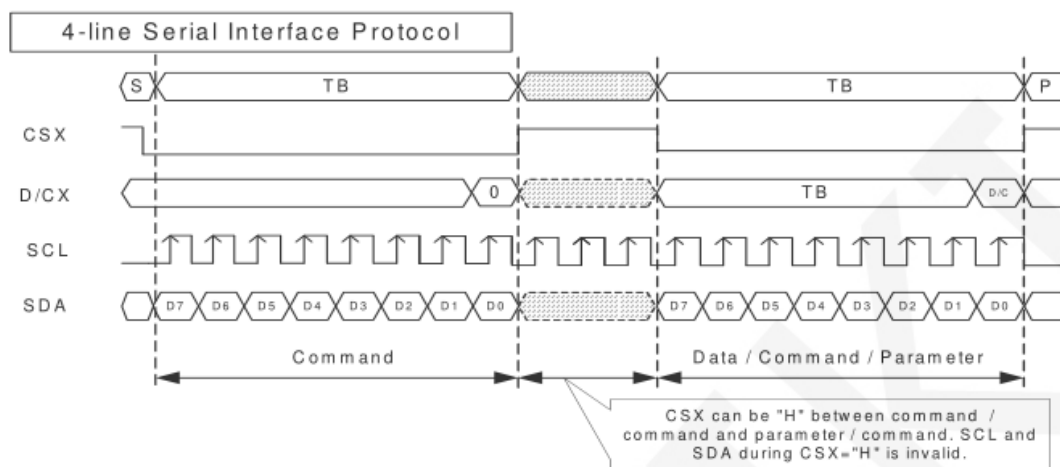
1、ST7735S 控制器简介

ST7735S 控制器支持的最大分辨率为 132*162, 拥有一个 48114 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。ST7735S 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。

ST7735S 控制器使用 16bit (RGB565) 来控制该显示模块的一个像素点颜色显示，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。ST7735S 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：

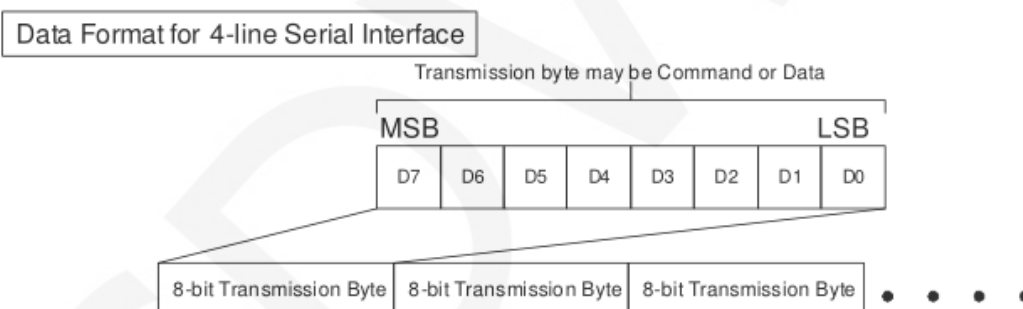


CSX 为从机片选，仅当 CSX 为低电平时，芯片才会被使能。

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 CPHA = 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 CPHA = 0，CPOL = 0

使用说明

1、Arduino 使用说明

接线说明：

使用软件SPI测试程序时，该显示模块可以直插到Arduino UNO和Mega2560开发板上，不需要手动接线（如图2和图3所示）。

使用硬件SPI测试程序时，该显示模块需要使用杜邦线手动接到Arduino UNO和Mega2560开发板上。

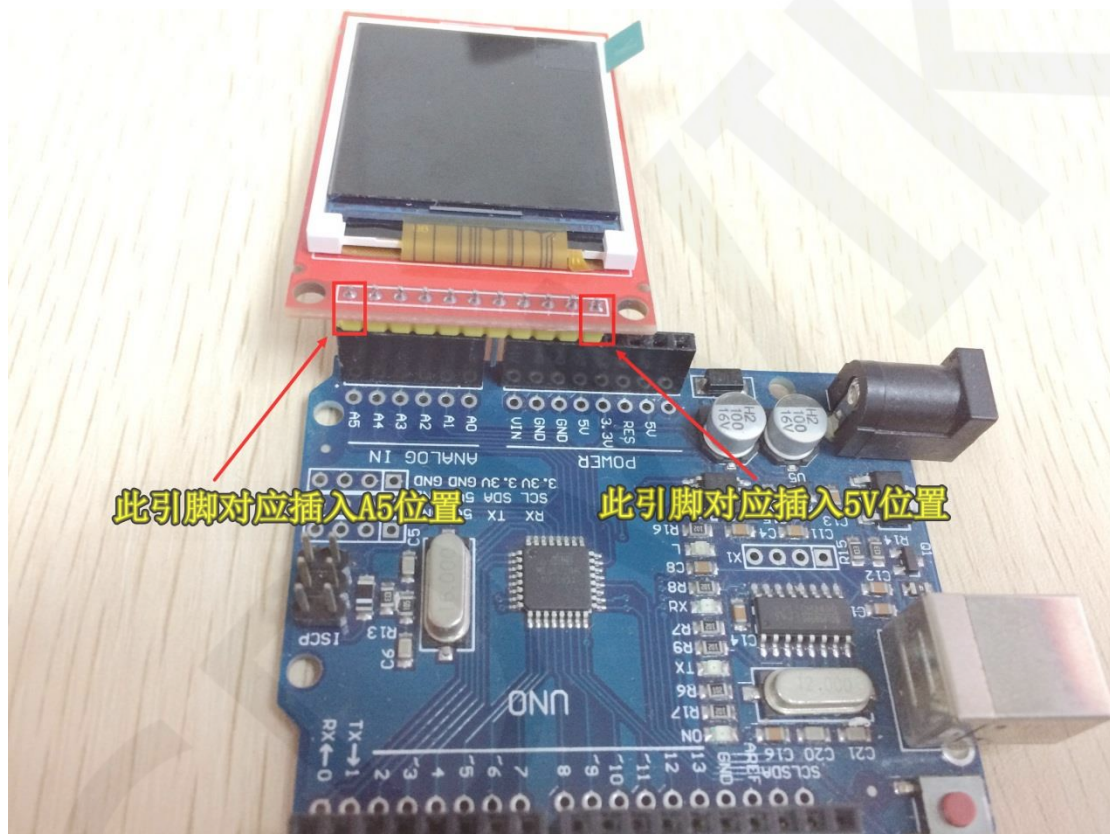


图2. 模块直插Arduino UNO图



图3. 模块直插Arduino Mega2560图

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	VCC	5V/3.3V	
2	GND	GND	
3	GND	GND	
4	NC	不需要接	
5	NC	不需要接	
6	NC	不需要接	
7	CLK	软件SPI	A1
		硬件SPI	13
8	SDA	软件SPI	A2
		硬件SPI	11
9	RS	A3	
10	RST	A4	

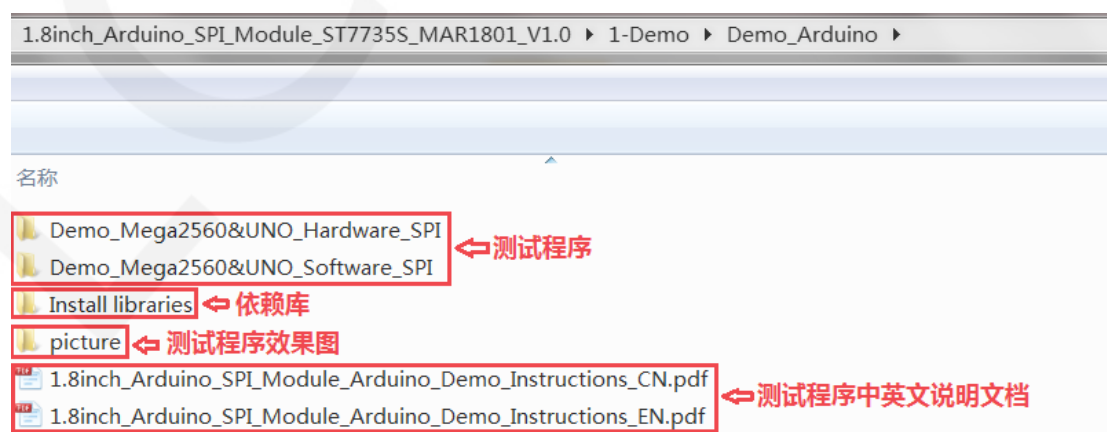
11	CS	A5
----	----	----

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	VCC	5V/3.3V
2	GND	GND
3	GND	GND
4	NC	不需要接
5	NC	不需要接
6	NC	不需要接
7	CLK	软件SPI A1
		硬件SPI 52
8	SDA	软件SPI A2
		硬件SPI 51
9	RS	A3
10	RST	A4
11	CS	A5

操作步骤:

- 按照上述接线说明将 LCD 模块（如图 1 所示）和 Arduino 单片机连接起来，并上电；
- 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）；
- 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：

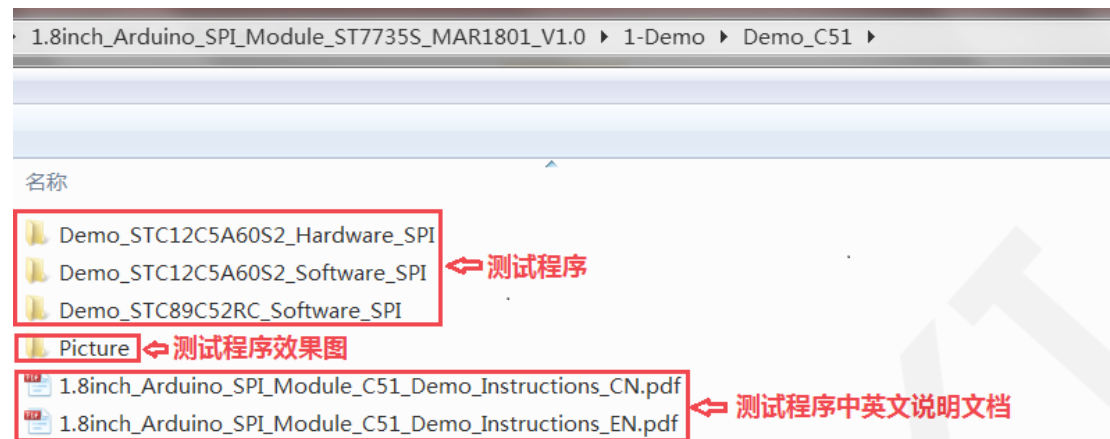
STC89C52RC和STC12C5A60S2单片机测试程序接线说明		
序号	模块引脚	对应STC89/STC12开发板接线引脚
1	VCC	5V/3.3V
2	GND	GND
3	GND	GND
4	NC	不需要接
5	NC	不需要接
6	NC	不需要接
7	CLK	P17
8	SDA	P15
9	RS	P12
10	RST	P33
11	CS	P13

操作步骤：

A、按照上述接线说明将 LCD 模块（如图 1 所示）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	GND	GND
4	NC	不需要接
5	NC	不需要接
6	NC	不需要接
7	CLK	PB13
8	SDA	PB15
9	RS	PB10
10	RST	PB12

11	CS	PB11
----	----	------

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	GND	GND
4	NC	不需要接
5	NC	不需要接
6	NC	不需要接
7	CLK	PB13
8	SDA	PB15
9	RS	PB10
10	RST	PB12
11	CS	PB11

STM32F407ZGT6单片机测试程序接线说明

序号	引脚丝印	对应Explorer STM32F4开发板接线引脚
1	VCC	5V/3.3V
2	GND	GND
3	GND	GND
4	NC	不需要接
5	NC	不需要接
6	NC	不需要接
7	CLK	PB3
8	SDA	PB5

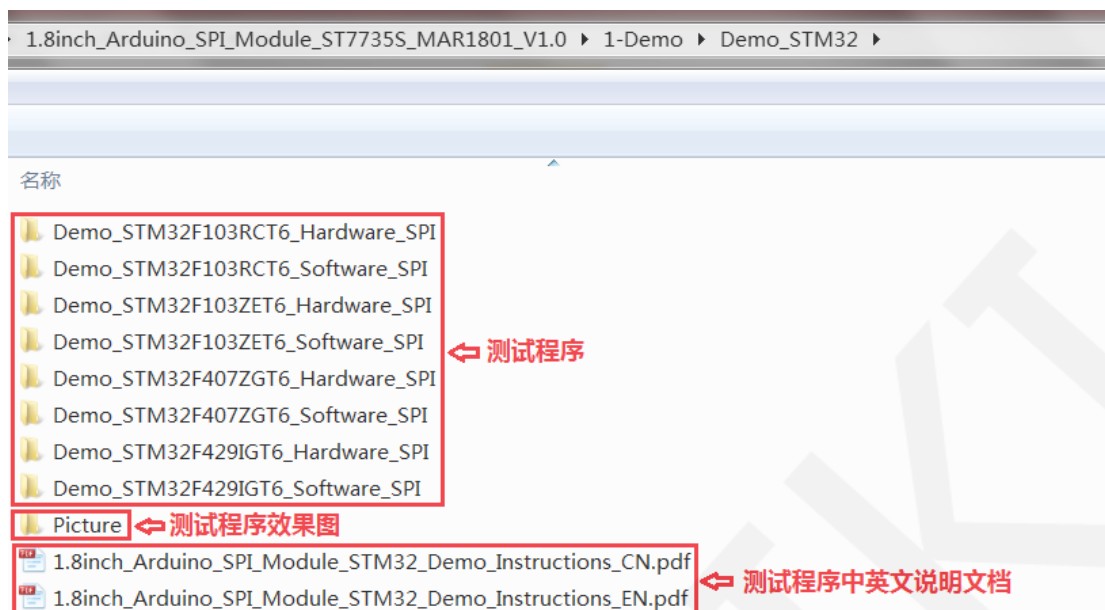
9	RS	PB14
10	RST	PB12
11	CS	PB15

STM32F429IGT6单片机测试程序接线说明

序号	引脚丝印	对应Apollo STM32F4/F7开发板接线
1	VCC	5V/3.3V
2	GND	GND
3	GND	GND
4	NC	不需要接
5	NC	不需要接
6	NC	不需要接
7	CLK	PF7
8	SDA	PF9
9	RS	PD5
10	RST	PD12
11	CS	PD11

操作步骤:

- A、按照上述接线说明将 LCD 模块（如图 1 所示）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

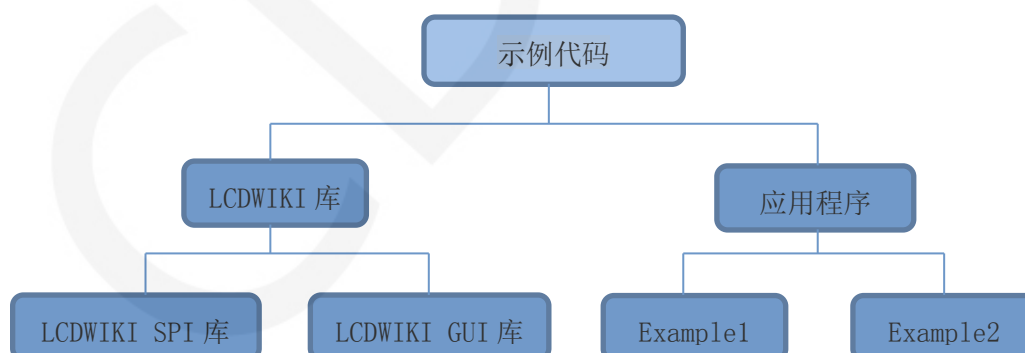
D、LCD 模块如果正常显示字符和图形，则说明程序运行成功；

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，

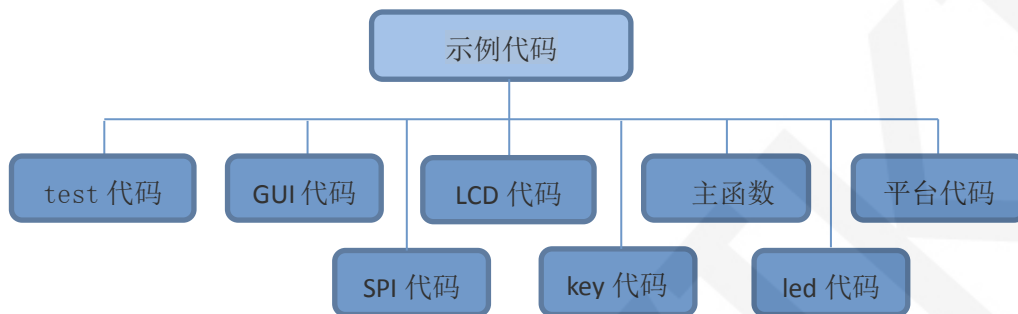
数据和命令传输，像素点坐标和颜色设置，显示方式配置等。

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 (C51 的 SPI 代码放在 LCD 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有任何区别，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。如下图所示(以软件 SPI 测试程序为例)：

```
//parameters define
#define MODEL ST7735S
#define CS    A5
#define CD    A3
#define RST   A4
#define SDA   A2
#define SCK   A1
#define LED   -1    //if you don't need to control the LED
```

使用 Arduino 软件 SPI 测试程序时，由于模块可直插到 Arduino 开发板上，所以其 GPIO 定义不能修改。

使用 Arduino 硬件 SPI 测试程序时，SDA 和 SCK 引脚不能修改（其由系统定义，不需要再定义），其他 GPIO 可以根据需求灵活定义。

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（以 STC12C5A60S2 软件 SPI 测程序为例）：

```
//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
```

如果使用软件 SPI，所有引脚定义都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 引脚可以定义成其他任何空闲的 GPIO。LCD_CLK 和 LCD_SDI 引脚不能修改（其由系统定义，不需要再定义）。

C、STM32 测试程序 GPIO 定义说明

STM32 的液晶屏非 SPI 的 GPIO 定义放在 lcd.h 里面，如下图所示（以 STM32F103RCT6 软件测试程序为例）：

```
//-----LCD端口定义-----
#define GPIO_TYPE  GPIOB  //GPIO组类型
//#define LED      9      //背光控制引脚      PB9
#define LCD_CS     11     //片选引脚          PB11
#define LCD_RS     10     //寄存器/数据选择引脚 PB10
#define LCD_RST    12     //复位引脚          PB12
```

所有引脚定义都可以定义成其他任何空闲 GPIO。

STM32 的液晶屏 SPI 的 GPIO 定义放在 spi.h 里面，如下图所示（以 STM32F103RCT6 软件测试程序为例）：

```
#define LCD_CTRL      GPIOB  //定义TFT数据端口
#define SPI_SCLK      GPIO_Pin_13 //PB13--->>TFT --SCL/SCK
#define SPI_MISO      GPIO_Pin_14
#define SPI_MOSI      GPIO_Pin_15 //PB15 MOSI--->>TFT --SDA/DIN
```

如果使用软件 SPI，所有引脚定义都可以定义成其他任何空闲的 GPIO。

如果使用硬件 SPI，这些引脚都不能修改（其由系统定义，不需要再定义），同时需要在 lcd.c 文件里面的 LCD_GPIOInit 函数里将 SPI_SCLK、SPI_MISO 以及 SPI_MOSI 引脚初始化去掉，如下图所示（以 STM32F103RCT6 软件测试程序为例）：

```
void LCD_GPIOInit(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; //推挽输出
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_14;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);
}
```

红圈里面的内容都要去掉。

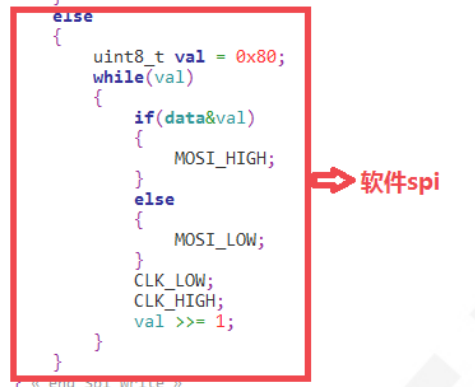
4、SPI 通信代码实现

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
```



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、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 通信代码在 spi.c 中实现，如下图所示：

```
void SPIv_WriteData(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;
    }
}
```

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。

C、C51 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 lcd.c 中实现，如下图所示：

```
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;
    }
}
```

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。