技术文章系列——

MzLH08的MCS51驱动演示说明

——MzLH08 模块应用文章

V1.0 – 2010.02

北京铭正同创科技有限公司技术资料

http://www.mzdesign.com.cn

版权声明

目录

引言

北京铭正同创科技有限公司给广大用户提供了一款功能强大的字库液晶模块——MzLH08-12864 模块；并为其配备了完整的驱动函数库，本文将介绍该模块所配套的演示程序，演示的平台为 MCS51，并附带铭正同创为 MzLH08 模块所配的基本功能例程的函数说明。

关键字

MzLH08　　LCD　　中文字库 LCD　　中文显示　　铭正同创　MCS51

1 技术概述

MzLH08 中文字库型液晶模块为北京铭正同创科技有限公司为广大用户提供的一款功能强大的 LCD 显示模块，并为用户提供了完整的驱动程序；而驱动程序当中包括了基本的绘图功能函数、中西文显示功能函数等。

本文着重介绍应用 MzLH08 模块配套的驱动程序进行基本显示操作的原理，以及过程；并介绍该模块所配的基本驱动程序的功能函数接口。

本文所介绍的例程均为 Keil C51 的环境下的工程。

2 MzLH08 模块简介

2.1 模块特点

MzLH08-12864 为一块 128×64 点阵的 LCD 显示模组，模组自带两种字号的汉字库（包含一、二级汉字库）以及两种字号的 ASCII 码西文字库；并且自带基本绘图功能，包括画点、画直线、矩形、圆形等；此外该模块特色的地方就是还自带有直接数字显示。模组为串行 SPI 接口，接口简单、操作方便；与各种 MCU 均可进行方便简单的接口操作。

- 128 × 64 点阵 STN ；
- 串行 SPI 接口方式（仅写入）；
- 自带 12×12 点和 16×16 点汉字库（包含一级和二级汉字库）；
- 自带 6×10、8×16 点 ASCII 码西文字库（96 个字符）；
- 自带基本绘图 GUI 功能（绘点、直线、矩形、矩形框、实心圆形、圆形框）；
- 自带整型数显示功能，直接输入整型数显示，而无需作变换；
- 带有背光控制指令，只需一条指令便可控制背光亮度等级（0~127）。

2.2 主要功能与基本参数

MzLH08 模块的基本参数如下表：

显示模式	STN 液晶	蓝底白点
显示格式	128×64 点阵地图形液晶显示	
输入数据	串行 SPI 接口	非标准 SPI
模块尺寸	50（长）×37（宽）×10（高）mm	不包含直插针高度
视屏尺寸	40.92（长）×24.28（宽）mm	
点大小	0.32（宽）×0.38（长）mm	
像素尺寸	0.28（宽）×0.34（长）	
背光	白色 LED	
供电	3.3V	可选择 5V 供电的模块

极限电器特性（针对 3.3V 模块）：

参数	符号	最小	最大	单位

供电电压	VDD	-0.3	3.6	V
端口输出电平	Vout	-0.3	Vdd+0.3	V
端口输入电平	Vin	-0.3	5	V
操作温度范围	Topr	-10	70	℃
贮存温度	Tstr	-20	80	℃

电器特性（针对 3.3V 模块）：

参数		符号	条件	最小	典型	最大	单位
工作电压		Vdd	—	2.95	3.3	3.6	V
输入电平	High Level	Vih	—	0.8Vdd	Vdd	5	V
	Low Level	Vil		Vss		0.2Vdd	

MzLH08 模块可以按照用户的要求提供 5V 供电的模块，但是需要注意，这些 5V 的模块仅仅是在模块内部增加一个 5V 到 3.3V 的线性稳压模块电路，实际上模块的端口仍是 3.3V 的，只不过这些端口可以承受 5V 的输入。

模块工作电流：(3.3V 模块，工作温度为 25 摄氏度)

参数	符号	条件	最小	典型	最大	单位
模块正常工作电流	I_{S1}	全屏显示黑，背光=0	7.74	7.77	7.80	mA
	I_{S2}	全屏显示白，背光=0	7.79	7.81	7.83	
	I_{S3}	全屏显示白，背光=10	9.05	9.07	9.09	
	I_{S4}	全屏显示白，背光=20	10.15	10.17	10.19	
	I_{S5}	全屏显示白，背光=40	12.34	12.37	12.40	
	I_{S5}	全屏显示白，背光=60	14.57	14.59	14.61	
	I_{S6}	全屏显示白，背光=80	16.80	16.81	16.83	
	I_{S6}	全屏显示白，背光=100	19.03	19.05	19.07	
	I_{S6}	全屏显示白，背光=127	27.90	27.80	27.70	
低功耗模式工作电流	I_{ON1}	显示开，全屏显示黑，无背光	7400	745	750	uA
	I_{ON2}	显示开，全屏显示白，无背光	770	775	780	
	I_{OFF}	屏幕关闭显示	700	705	710	

注：以上测试均为模块控制引脚悬空的条件下，屏上如有显示，则保留静态画面，MCU 与模块连接断开，保留电源连接。

2.3 引脚示意图

下面的图为 MzLH08-12864 模块的实物图。

图 1.1 模块引脚分布示意图

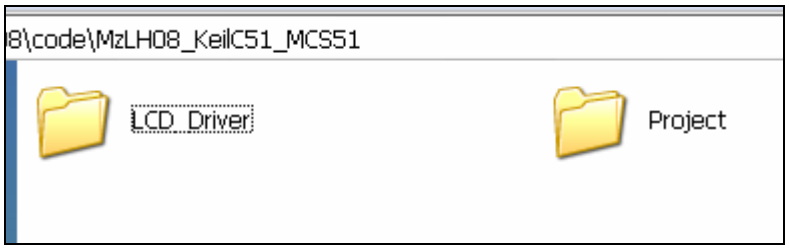
表 1.1 模块接口引脚说明

序号	接口引脚名	说明
1	VDD	LCD 供电
2	CS	片选
3	SDA	数据输入线
4	BY	LCD 忙信号输出线（BUSY 线）
5	SCK	SPI 时钟线(<4.5 MHz 的速度)

6	RST	模块复位线(低电平复位)
7	GND	LCD 接地

2.4 范例程序结构说明

MzLH08 模块所配的 MCS51 驱动演示例程分布有如下几个文件夹：

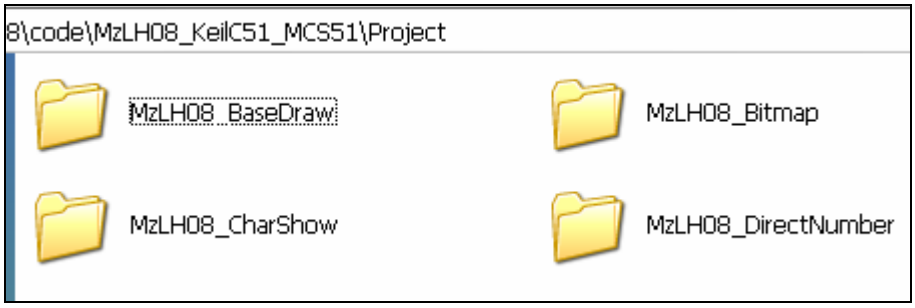


LCD_Driver 文件夹中包含有 MzLH08 模块的 MCS51 系列的驱动程序，本例程的工程对像为 MCS51 芯片，如果用户使其它芯片时，可能需要按照所使用的 MCU 作一些修改，当然这个需要按照实际的情况来看了。本例程共包含有四个工程，它们都使用同样的 LCD 驱动程序。

Project 文件夹中为四个工程的分类文件夹，以及一份说明文档；所有的工程都将共用前面近述的 LCD_Driver 文件夹中的 LCD 驱动程序。

2.4.1 Project 文件夹

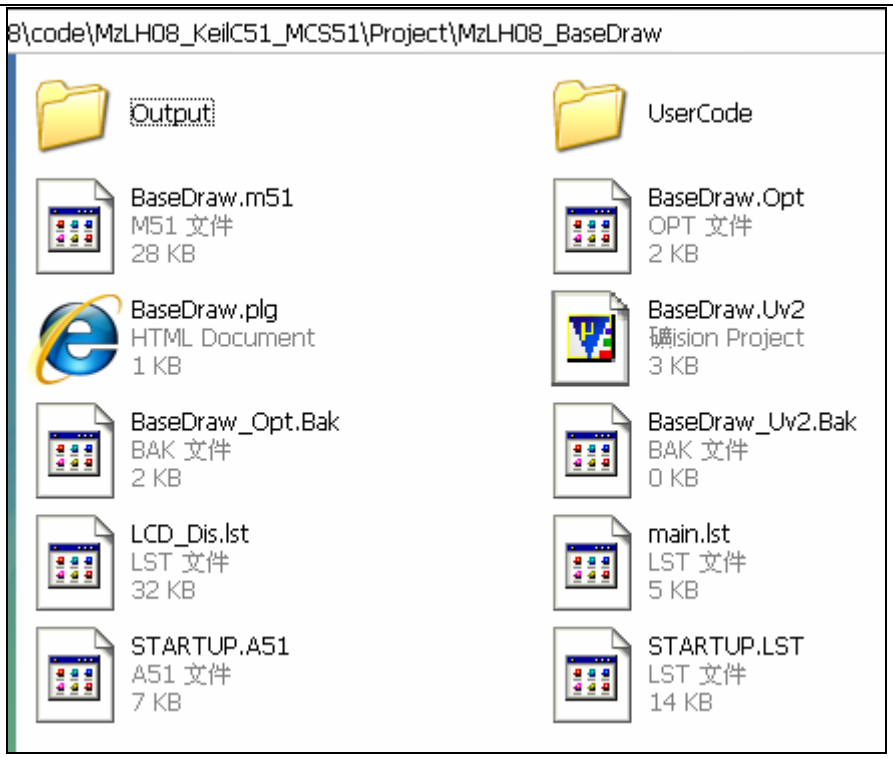
本例的四个工程的工程文件夹都放置在 Project 文件夹当中，如下图：



四个文件夹分别对应四个演示工程，分别是：

- MzLH08_BaseDraw：基本绘图显示功能演示；
- MzLH08_Bitmap：位图显示功能演示；
- MzLH08_CharShow：中英文字符显示功能演示；
- MzLH08_DirectNumber：直接数字显示功能演示。

这四个工程文件夹里面的结构是一样的，在此只看看一下其中一个便可：



Output 文件夹为工程的一些中间和最终文件的输出文件夹；UserCode 文件夹中为本工程所使用的一些用户文件，因为每个工程所演示的功能不同，所以在此都分开了；BaseDraw.Uv2 为工程文件，可以直接打开它来打开整个工程。

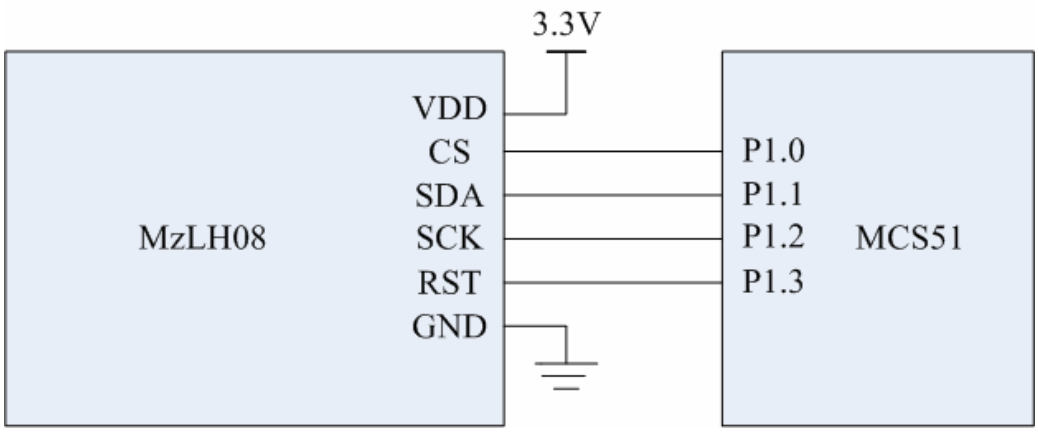
3 基本绘图显示功能演示例程

3.1 MzLH08_BaseDraw 基本绘图显示例程描述

MzLH08_BaseDraw 的例程是用于演示 MzLH08 模块的基本显示功能，比如一些绘点、直线矩形等。

3.2 例程硬件连接简图

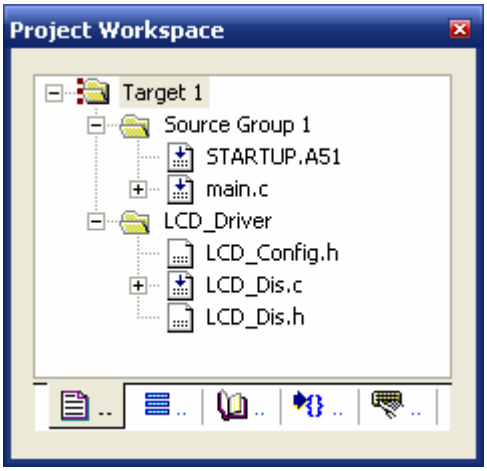
本工程范例使用了 MzLH08 模块的基本绘图显示功能，本例的硬件连接如下图所示：



本例程使用 GPIO 来模拟 SPI 时序，如用户使用硬件 SPI 端口，请参考 MzLH08 模块的端口进行连接和编程。上图的模块为 3.3V 的模块，有关 MCS51 的其它电路不在图中画出，请用户具体参考其它的开发板文档进行了解。

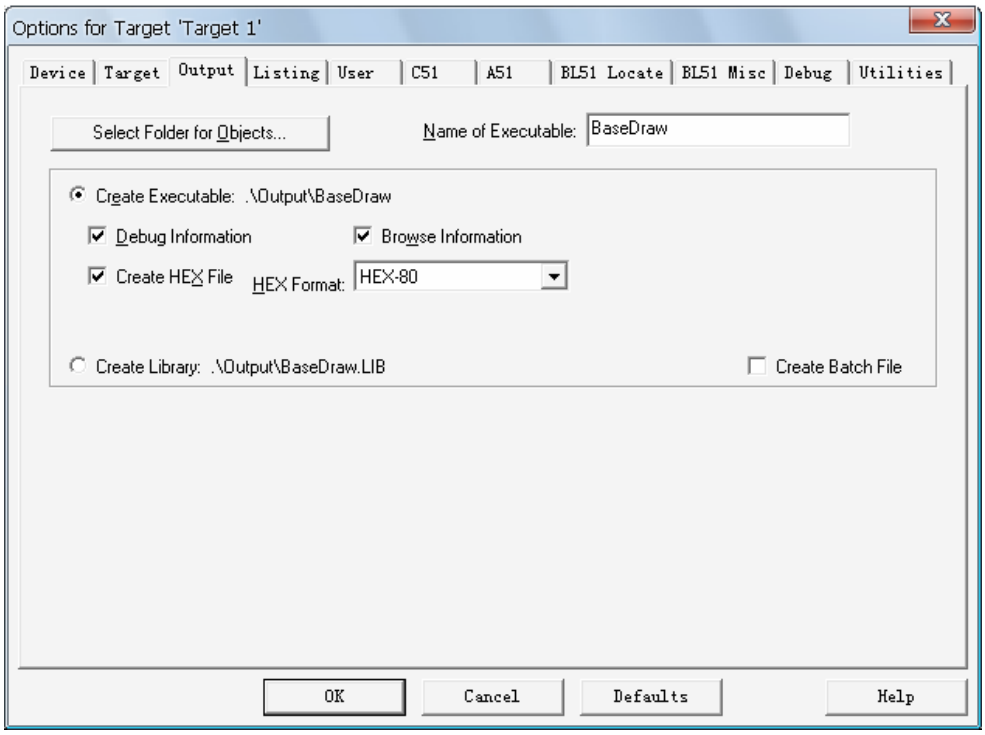
3.3 例程软件设置相关说明

本例程在打开后，工程组织如下图所示：

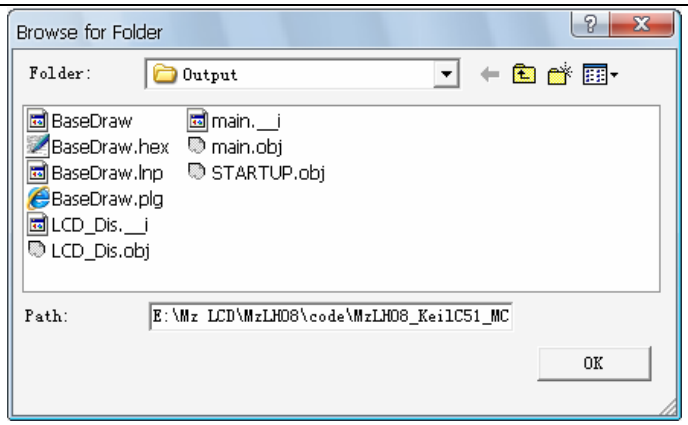


由于 LCD 驱动程序是共用了上层目录中的源文件，所以本例程针对 Keil 的开发环境进行了一些设置，主要是输出文件路径指定和头文件包含路径设置。

打开”Target”的 Option 设置对话框，可以在第三项”Output”选项卡中对输出文件路径进行设置，如下图：

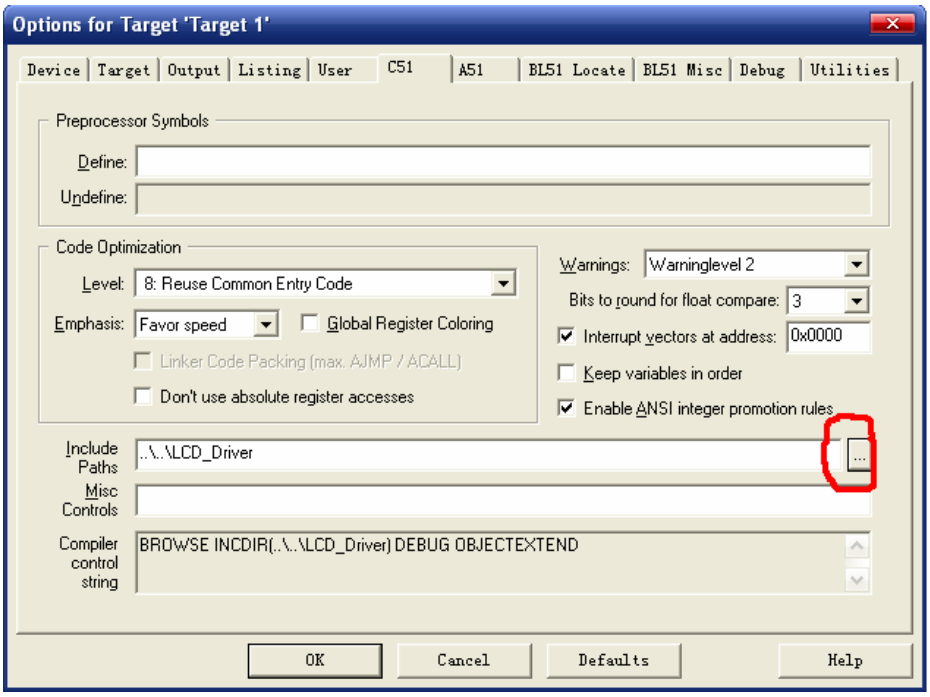


打开“Select Folder for Objects…”可以对路径进行指定，如下图所示：

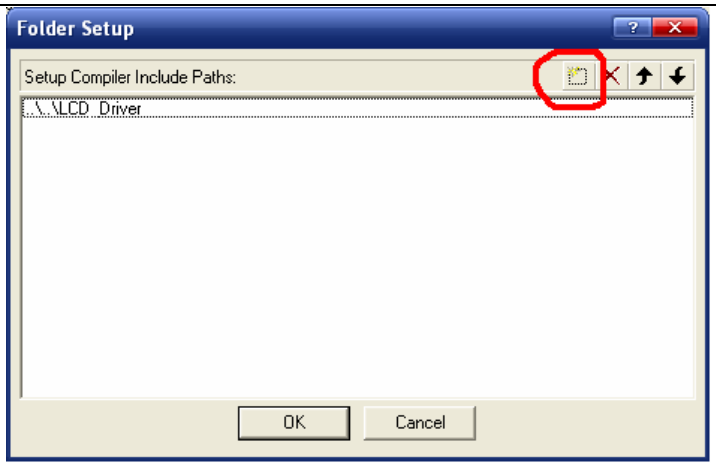


本工程范例指定了工程文件夹中的 OutPut 文件夹，用于保存这些生成的中间文件和目标文件。

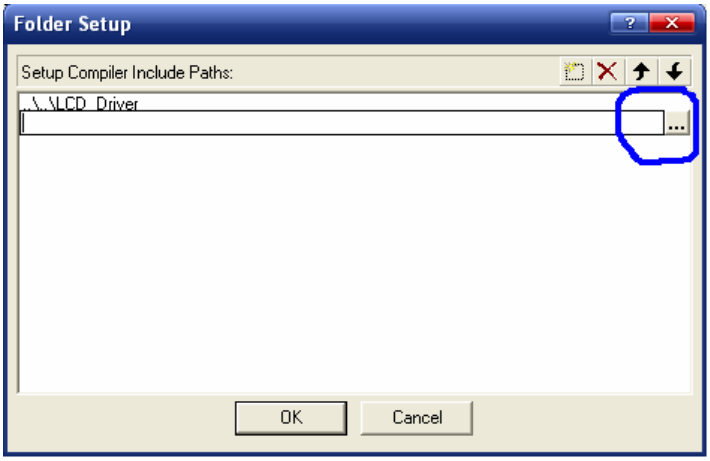
而设置头文件包含的选项卡在“Target”的 Option 对话框中的“C51”选项卡中设置，如下图：



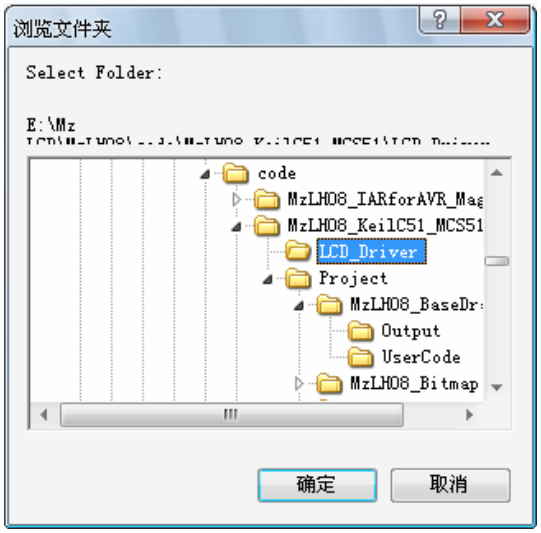
点击红圈中的 Button，可以在打开的对话框中添加默认的头文件路径，如下图：



点击上图中红外圈中的 Button 可以新建一个默认路径，如下图：

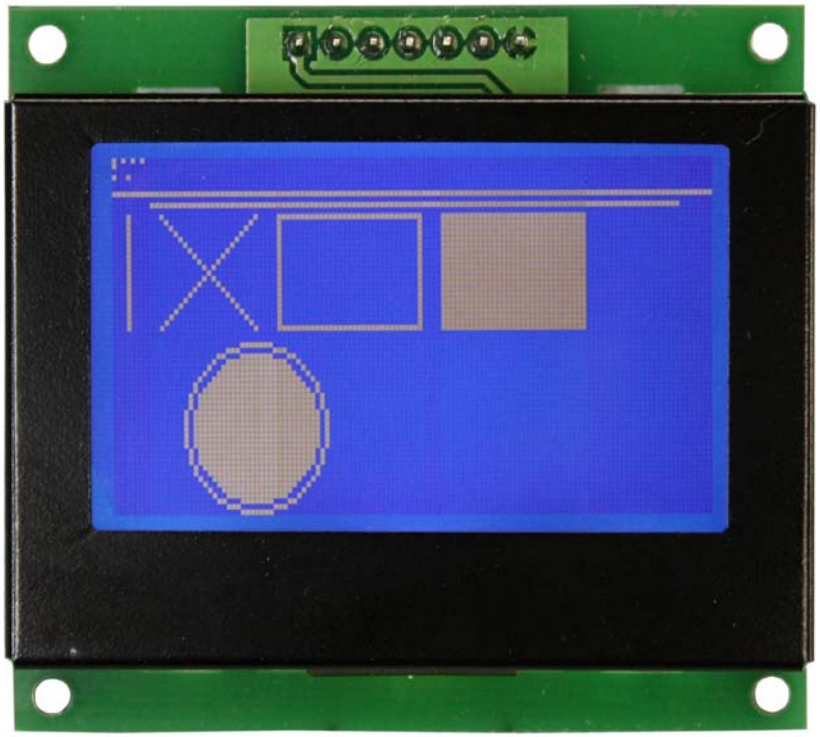


再点击蓝色圈中的按钮，就可以打开路径设置的界面了，将要包含的头文件的路径打开即可，这样在编程时，就不必在包含头文件时再将该头文件的相对或绝对路径写上了；如下图：



3.4 操作演示说明

MzLH08 模块的基本功能范例演示了模块上的基本的显示功能，包括绘制点、直线、矩形、ASCII 字符显示（多种显示效果及多种字号演示）、中文显示以及特别的直接数字显示，下图为显示的拍摄效果图：



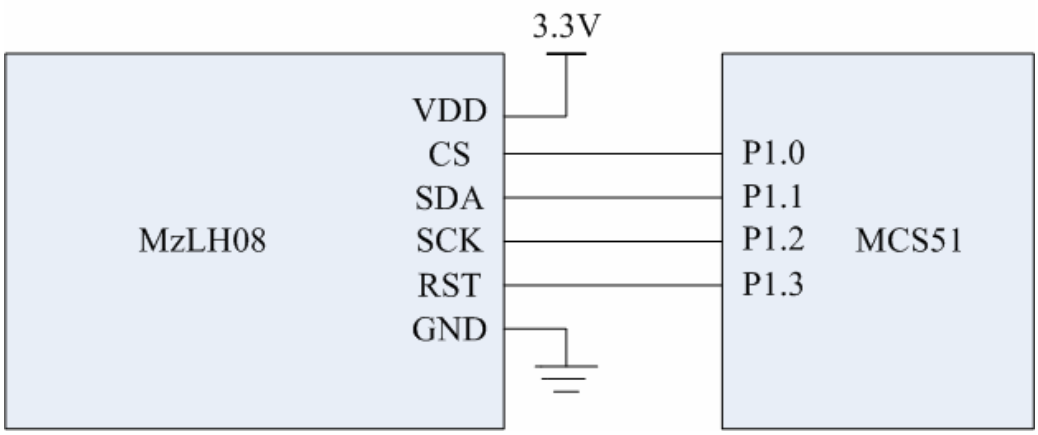
4 位图显示功能演示程序

4.1 MzLH08_Bitmap 基本功能描述

本工程范例演示了 MzLH08 模块显示单色位图的功能，如用户需要显示其它的图片，请依照 MzLH08 模块位图显示指令对位图字模的要求，利用字模提取工具进行字模提取，本例仅供参考。

4.2 例程硬件连接简图

本工程范例使用了 MzLH08 模块的基本绘图显示功能，本例的硬件连接如下图所示：

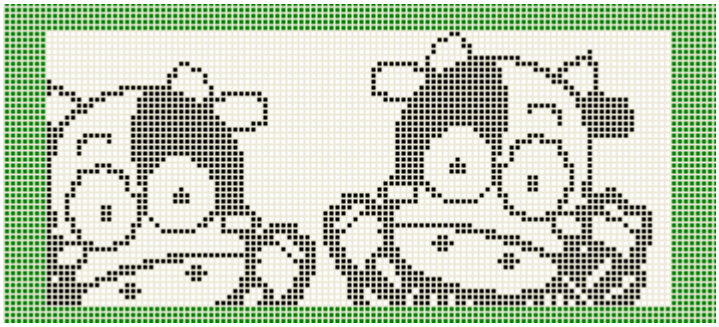


本例程使用 GPIO 来模拟 SPI 时序，如用户使用硬件 SPI 端口，请参考 MzLH08 模块的端口进行连接和编程。上图的模块为 3.3V 的模块，有关 MCS51 的其它电路不在图中画出，请用户具体参考其它的开发板文档进行了解。

4.3 例程软件设置相关说明

有关 Keil 的工程设置内容在上一章已经有述，这里与前面的类似，就不再重述，而着重介绍利用 MzLH08 模块的位图显示指令时，如何提取位图的字模，并将字模数据添加至工程当中。

按照模块的规定，用户在利用字模工具提取位图的字模数据时，要按照横向取模的方式取模，顺序是从左到右，自上到下；当位图的横向的点数不为 8 的整数倍时，会在最后一个字节的数据进行低位补零，使得每一行的图像字模占用整数个字节的空间。如下面的位图：



图中淡绿色的外框中的图像为一个宽度为 101 点（图中已经补齐，变成 104 点了），高度为 46 点的单色位图；选择取字模的方式为横向取模，下面是提取出来的第一行的字模数据，如下：

```
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x00,0x00,0x00,0x00,
```

接下来是第二行的数据：

```
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x64,0x00,0x00,0x00,0x00,
```

感兴趣的用户可以将字模数据与图中的点进行一下对比，这样思路更清晰一些。

下面是整个图像的字模数据：

```
/*--  宽度 x 高度=101x46  --*/
/*--  宽度不是 8 的倍数，现调整为：宽度 x 高度=104x46  --*/
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x64,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x86,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x82,
0x00,0x00,0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x03,0xFE,0x00,0xE0,
0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x01,0xFD,0x0F,0xE3,0xC3,0xB0,0x00,0x00,0x00,
0x04,0xC0,0x00,0x00,0x03,0x07,0xBF,0xF0,0x76,0x10,0x00,0x00,0x00,0x0C,0x20,0x00,
0x00,0x02,0x01,0xFF,0xF0,0x1C,0x10,0x00,0x80,0x00,0x08,0x20,0x00,0x00,0x02,0x01,
0xFF,0xF0,0x0E,0x10,0x00,0xE0,0x0F,0xF8,0x10,0x00,0x00,0x02,0x03,0xFF,0xF8,0x03,
0x10,0x00,0xB8,0x78,0xFE,0x17,0xF0,0x00,0x03,0x87,0xFF,0xF8,0x01,0xB8,0x00,0x0D,
0xC1,0xFF,0xBC,0x18,0x00,0x00,0xFF,0xFF,0xF8,0x00,0xFF,0x80,0x07,0x01,0xFF,0xF0,
0x08,0x00,0x00,0x0F,0xFF,0xF8,0xF8,0xFF,0x80,0x0E,0x01,0xFF,0xF0,0x08,0x00,0x00,
0x1F,0xFF,0xF8,0x8C,0x7F,0xC0,0x18,0x03,0xFF,0xF8,0x08,0x00,0x00,0x1F,0xFF,0xF8,
0x04,0x7F,0xC0,0xB0,0x03,0xFF,0xFC,0x38,0x00,0x00,0x3F,0xFF,0xF8,0x04,0x7F,0xC0,
0xE0,0x03,0xFF,0xFF,0xE0,0x00,0x00,0x3F,0xE1,0xF8,0x00,0x3F,0xC0,0xE3,0xE3,0xFF,
0xFE,0x00,0x00,0x00,0x3F,0xC0,0xF0,0xE0,0x3F,0x80,0xC6,0x23,0xFF,0xFF,0x00,0x00,
0x00,0x3F,0x00,0x73,0xBC,0x20,0x00,0xC4,0x03,0xFF,0xFF,0x00,0x00,0x00,0x3F,0x00,
0x36,0x04,0x20,0x00,0xC4,0x03,0xFF,0xFF,0x80,0x00,0x00,0x3E,0x00,0x34,0x06,0x20,
```

```
0x00,0x80,0x03,0xF0,0xFF,0x80,0x00,0x00,0x3E,0x08,0x24,0x02,0x20,0x00,0x80,0xE1,
0xE0,0x7F,0x80,0x00,0x00,0x3E,0x1C,0x38,0x03,0x20,0x00,0x87,0xB9,0xC0,0x1F,0x80,
0x00,0x00,0x3E,0x1C,0x30,0x01,0x20,0x00,0x84,0x0D,0x80,0x1F,0x80,0x00,0x00,0x7E,
0x00,0x30,0xC1,0x20,0x00,0x8C,0x05,0x80,0x0F,0x80,0x00,0x00,0xFF,0x00,0x30,0xC1,
0x60,0x00,0x88,0x04,0x82,0x0F,0x80,0x00,0x01,0x85,0x80,0x50,0xC3,0x66,0x00,0x98,
0x03,0x87,0x0F,0x80,0x00,0x7D,0x84,0xC0,0x98,0x02,0x7F,0x00,0x90,0x01,0x87,0x0F,
0x80,0x00,0x7E,0xFC,0x3F,0x08,0x06,0x3F,0x80,0x90,0x61,0x80,0x0F,0xC0,0x01,0xE7,
0xE0,0x00,0x0C,0x04,0xF9,0xF0,0xD0,0x61,0x80,0x1F,0xE0,0x03,0xCF,0xC0,0x00,0x07,
0x1F,0xFC,0xF8,0xD8,0x61,0x40,0x34,0x30,0x03,0x91,0xC0,0x00,0x01,0xF8,0xE7,0x18,
0xC8,0x03,0x20,0x64,0x37,0xC3,0x20,0xFF,0xFE,0x00,0x08,0x60,0x98,0x8C,0x02,0x1F,
0x87,0xEF,0xC3,0xC1,0xC0,0x0F,0xE0,0x0C,0x60,0x78,0xE4,0x06,0x00,0x00,0xFC,0xF3,
0x80,0xC0,0x60,0x7E,0x07,0xE0,0x38,0xFF,0x1C,0x00,0x00,0x7E,0x7B,0x80,0xC0,0xF0,
0x0F,0x83,0xF8,0x38,0xE3,0xF0,0x00,0x00,0x71,0x39,0xC1,0xE0,0x60,0x01,0xE3,0xBE,
0x70,0xC2,0x00,0x0F,0xFF,0xE0,0x99,0xC3,0xE0,0x00,0x0C,0x3F,0xFF,0x70,0xC6,0x00,
0xFE,0x00,0x70,0x78,0xE6,0x70,0x00,0x1E,0x0F,0xFB,0xF0,0xFC,0x0F,0xC0,0xC0,0x60,
0x38,0xF6,0x38,0x00,0x0C,0x07,0xF3,0xE0,0xF8,0x3E,0x01,0xE0,0x60,0x38,0x7F,0x3E,
0x00,0x00,0x07,0xE7,0xF0,0xB8,0xF0,0x00,0xC0,0xF0,0x70,0x3F,0x9F,0xC0,0x00,0x0D,
0xCF,0x30,0xFF,0x86,0x00,0x00,0xF8,0x70,0x3F,0xDF,0xF8,0x00,0x1E,0xDF,0x70,0xFE,
0x0F,0x00,0x01,0xCC,0xE0,0x3B,0xFB,0xFF,0xE1,0xFF,0x7C,0xE0,0xFC,0x06,0x00,0x03,
0x8D,0xE0,0x39,0xF7,0x6F,0xFF,0xEB,0xBB,0xE0,0xFC,0x00,0x00,0x0F,0x9F,0xC0,0x1C,
0x77,0x6D,0xFF,0x6A,0xDF,0xC0
```

工程中，将图像的字模数据保存于一个数组当中，以 code unsigned char 进行定义，这样可以将数组保存于 code 空间，放置在 bitmap.c 的文件里，而在 bitmap.h 的头文件当中对其进行外部声明，如下：

```
02
03  code unsigned char Bitmap01[] =
04  {
05  /*--  调入了一幅图像：E:\Mz LCD\MzLH06\code\LPC22XX_MzLH06_Demo\Project\Prj_MzLH
06  /*--  宽度x高度=101x46  --*/
07  /*--  宽度不是8的倍数，现调整为：宽度x高度=104x46  --*/
08  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
09  0x00, 0x00, 0x00, 0x00, 0x00, 0x64, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
10  0x00, 0x00, 0x86, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x82,
11  0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0xFE, 0x00, 0xE0,
12  0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x01, 0xFD, 0x0F, 0xE3, 0xC3, 0xB0, 0x00, 0x00, 0x00,
13  0x04, 0xC0, 0x00, 0x00, 0x03, 0x07, 0xBF, 0xF0, 0x76, 0x10, 0x00, 0x00, 0x00, 0x0C, 0x20, 0x00,
14  0x00, 0x02, 0x01, 0xFF, 0xF0, 0x1C, 0x10, 0x00, 0x80, 0x00, 0x08, 0x20, 0x00, 0x00, 0x02, 0x01,
15  0xFF, 0xF0, 0x0E, 0x10, 0x00, 0xE0, 0x0F, 0xF8, 0x10, 0x00, 0x00, 0x02, 0x03, 0xFF, 0xF8, 0x03,
16  0x10, 0x00, 0xB8, 0x78, 0xFE, 0x17, 0xF0, 0x00, 0x03, 0x87, 0xFF, 0xF8, 0x01, 0xB8, 0x00, 0x0D,
17  0xC1, 0xFF, 0xBC, 0x18, 0x00, 0x00, 0xFF, 0xFF, 0xF8, 0x00, 0xFF, 0x80, 0x07, 0x01, 0xFF, 0xF0,
```

而在调用位图显示函数时，还需要注意一下，将实际图像的大小传递给函数，并将对应的位图字模数组的首地址传送过去，如下便是调用函数显示该位图的代码：

```
53      PutBitmap(10,10,101,46,Bitmap01);          //在坐标点10,10为左上角的位置显示尺寸大小为101X46点的位图
```

而需要注意的是 PutBitmap 这个函数，如果用户将本驱动程序至别的 MCU 上，或者使用与本例不一样的 CPU 时钟频率时，要适当修改一下该函数：

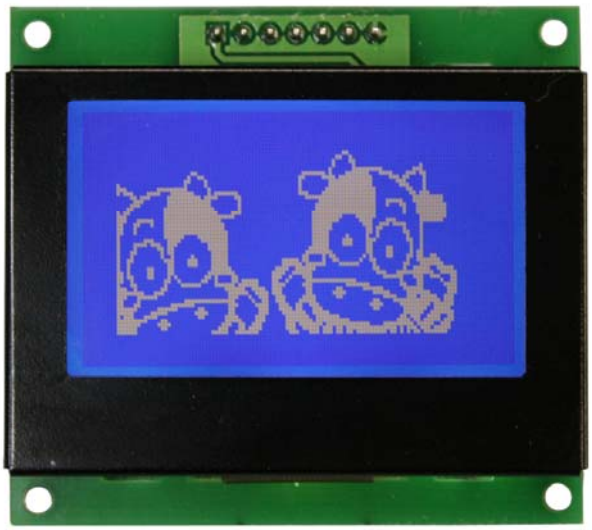
```
void PutBitmap(unsigned char x,unsigned char y,unsigned char width,unsigned char high,unsigned char *p)
{
    unsigned short Dat_Num;
    //unsigned short i;
    unsigned char ucTemp=0;
    SPI_SSSet(0);                       //SS置低电平
    SPI_Send(0x0e);
    SPI_Send(x);
    SPI_Send(y);
    SPI_Send(width);
    SPI_Send(high);

    width = width+0x07;
    Dat_Num = (width>>3)*high;
    while(Dat_Num--)
    {
        ucTemp++;
        SPI_Send(*p);
        if(ucTemp>250)                  //如果改换别的MCU时,这里可能需要适当的调整
        {
            TimeDelay(28);              //以及此处,以保证前面传送到LCD模块中的数据已被处理完
            ucTemp = 0;
        }
        p++;
    }
    SPI_SSSet(1);                       //完成操作置SS高电平
}
```

上图程序中每传输 250 个字节的数据时，适当延时一下，以待 LCD 模块将已经传送到模块上的数据处理完，然后再将后续的数据传送至 LCD 模块；这个用户可以参考一下 MzLH08 模块的手册，上面有说明该模块内部开了一个缓冲区以便于接收 MCU 传送来的指令以及数据；但一般情况下，LCD 模块对指令以及数据的处理相对较慢，所以为了防止缓冲区被填满，在发送大量数据时需要适当作一下延时。

4.4 操作演示说明

工程显示结果用相机拍摄出来如下图所示：



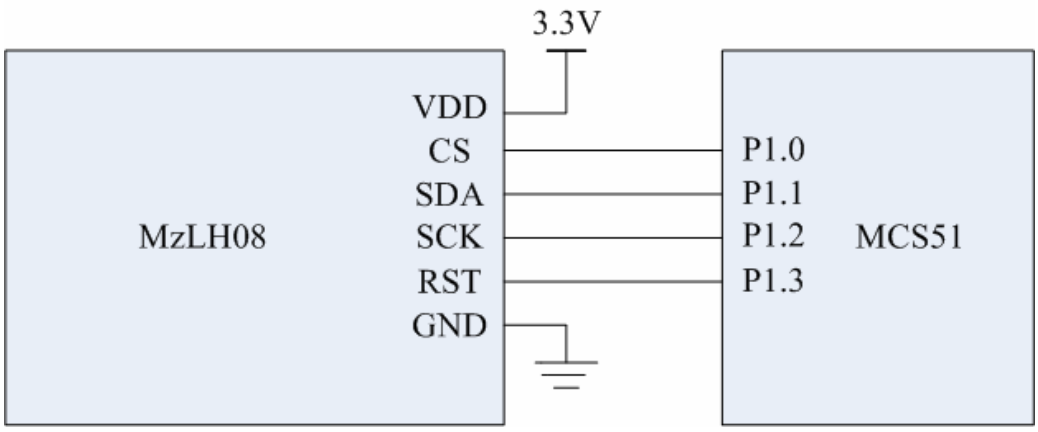
5 中英文字符显示功能演示例程

5.1 MzLH08_CharShow 基本功能描述

本工程范例演示了 MzLH08 模块自带的中英文符库的显示功能，并演示各种显示的效果。

5.2 例程硬件连接简图

本工程范例使用了 MzLH08 模块的基本绘图显示功能，本例的硬件连接如下图所示：



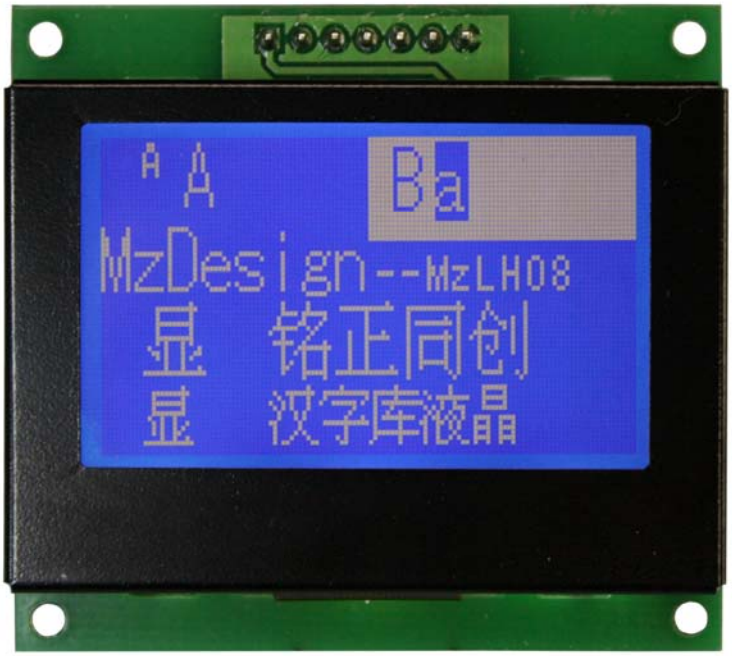
本例程使用 GPIO 来模拟 SPI 时序，如用户使用硬件 SPI 端口，请参考 MzLH08 模块的端口进行连接和编程。上图的模块为 3.3V 的模块，有关 MCS51 的其它电路不在图中画出，请用户具体参考其它的开发板文档进行了解。

5.3 例程软件设置相关说明

有关 Keil 的工程设置内容在上一章已经有述，这里与前面的类似。

5.4 操作演示说明

工程运行后，如连接无误，以及硬件工作都正常，将会出现如下图所示的显示：


A A
Ba
MzDesign--MzLH08
显 铭正同创
显 汉字库液晶

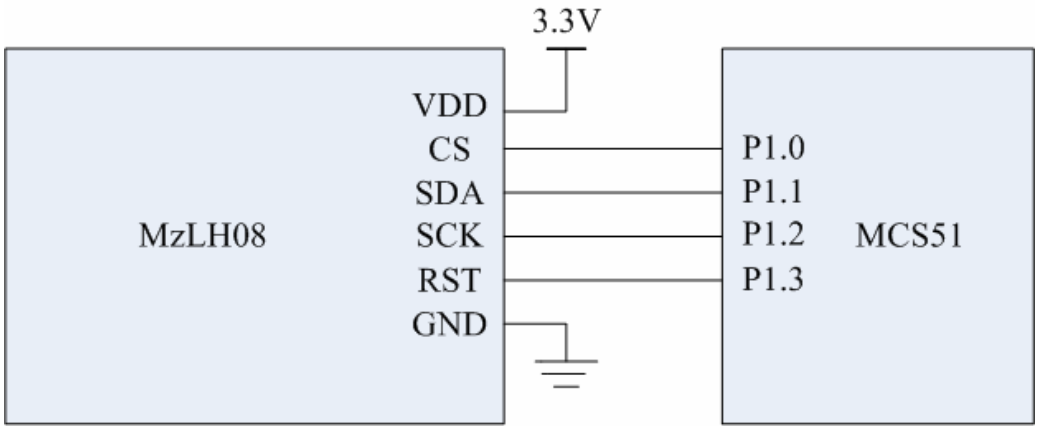
6 直接数字显示功能演示例程

6.1 MzLH08_DirectNumber 基本功能描述

MzLH08 模块提供了一个非常方便于用户的功能：直接数字显示功能，即用户可通过直接数字显示指令，将要显示的数据传送给模块，模块就可以直接将数据显示在指定的位置上，并且提供了几种显示的样式供用户选择使用。

6.2 例程硬件连接简图

本工程范例使用了 MzLH08 模块的基本绘图显示功能，本例的硬件连接如下图所示：



本例程使用 GPIO 来模拟 SPI 时序，如用户使用硬件 SPI 端口，请参考 MzLH08 模块的端口进行连接和编程。上图的模块为 3.3V 的模块，有关 MCS51 的其它电路不在图中画出，请用户具体参考其它的开发板文档进行了解。

6.3 例程软件设置相关说明

有关 Keil 的工程设置内容在前面章已经有述，这里与前面的类似。

MzLH08 模块的直接数字显示指令提供了三种显示样式，如下代码，分别演示了三种不同样式的显示效果调用：

```
45      FontSet(0,1);
46      ShowChar(10,0,ucTemp,0);              //在指定位区域显示一个无符字符型变量
47      ShowShort(50,0,usTemp,0);             //在指定位区域显示一个无符短整型变量
48
49      ShowChar(10,10,ucTemp,1);             //在指定位区域显示一个无符字符型变量
50      ShowShort(50,10,usTemp,1);            //在指定位区域显示一个无符短整型变量
51
52      ShowChar(10,20,ucTemp,2);             //在指定位区域显示一个无符字符型变量
53      ShowShort(50,20,usTemp,2);            //在指定位区域显示一个无符短整型变量
```

6.4 操作演示说明

工程运行后，如连接无误，以及硬件工作都正常，将会出现如下图所示的显示：

7 MzLH08 模块基本驱动程序介绍

铭正同创为 MzLH08 模块编写了一套基本的驱动程序，主要包含以下三个文件：

LCD_Config.h：主要是定义 LCD 驱动程序的端口使用；

LCD_Dis.c：包含底层的 SPI 读、写子程序以及各种功能接口函数；

LCD_Dis.h：对 LCD_Dis.c 中的函数进行外部声明。

7.1 端口定义

在 MzLH08 模块的例程中，将驱动该模块的端口定义放置在 LCD_Config.h 中进行了定义，这样当用户再将驱动程序移植到别的 MCU 上时，只需要修改这些配置即可。如下所示：

```
#include “REG52.h”

#define Dis_X_MAX          128-1
#define Dis_Y_MAX          64-1

//Define the MCU Register

sbit SPI_CS = P1^0;                  //定义 CS 接在 P1.0 端口
sbit SPI_SDA = P1^1;             //定义 SDA 接在 P1.1 端口
sbit SPI_SCK = P1^2;             //定义 SCK 接在 P1.2 端口
sbit SPI_RES = P1^3;             //定义 RESET 接 P1.3 端口
```

7.2 底层驱动函数

LCD_Dis.c 中有一个 LCD_Init 函数，主要完成对 MCU 与 LCD 模块相关的端口/硬件模块的初始化，如下代码：

```
//=========================================================================
// 函数: void LCD_Init(void)
// 描述: LCD 初始化程序，主要在里面完成端口初始化以及 LCD 模块的复位
// 参数: 无
// 返回: 无
//=========================================================================
void TimeDelay(unsigned int Timers)
```

```
{
        unsigned int I;
        while(Timers)
        {
            Timers--;
            for(i=0;i<100;i++) ;
        }
}
void LCD_Init(void)
{
        //SS 和 SCK 预先设置为高电平
        SPI_SCK = 1;
        SPI_CS = 1;

        //复位 LCD 模块
        SPI_RES = 0;
        TimeDelay(50);
        //保持低电平大概 2ms 左右
        SPI_RES = 1;
        TimeDelay(80);                    //延时大概 10ms 左右
}
```

在 LCD_Init()函数中，对 51 的端口进行了初始化，而代码中有一部分屏蔽的，这是对 LCD 模块进行复位的代码，但在本例中并没了对 LCD 模块复位进行控制，所以都将其屏蔽；此外 TimerDelay()函数并没有作一个准确的时间延时，仅仅是一个大概的延时处理，这点请用户根据实际情况调整。

LCD 片选信号设置子程序如下所示：

```
void SPI_SSSet(unsigned char Status)
{
        if(Status)                    //判断是要置 SS 为低还是高电平？//SS 置高电平
            SPI_CS = 1;
        else    //SS 置低电平
            SPI_CS = 0;
}
```

SPI 字节发送程序在本范例当中利用 GPIO 模拟了 SPI 的时序，如下代码所示：

```
void SPI_Send(unsigned char Data)
{
        unsigned char i=0;
        for(i=0;i<8;i++)
        {
            //SCK 置低
            SPI_SCK = 0;
            if(Data&0x0080)
                SPI_SDA = 1;
            else SPI_SDA = 0;//
            //SCK 上升沿触发串行数据采样
            SPI_SCK = 1;
            Data = Data<<1;                          //数据左移一位
        }
}
```

数据位通过 SDA 端口传送给 LCD 模块时，将会在 SCK 的上升沿将该位载入模块当中，这点请注意，如果用户使用硬件的 SPI 端口，请依照模块的接口时序来设置好。

7.3 MzLH08 驱动程序说明

功能	函数介绍	
清屏	函数名	void ClrScreen(void)
	功能介绍	全屏清屏
设置背光亮度等级	函数名	void SetBackLight(unsigned char Grade)
	功能介绍	Grade　0~127 的等级
字符覆盖模式设置	函数名	void FontMode(unsigned char Mode,unsigned char Color)
	功能介绍	Mode　字符覆盖模式设置，0 或 1 Color　覆盖模式为 1 时字符显示时的背景覆盖色
ASCII 字符字体设置	函数名	void FontSet(unsigned char Font_NUM,unsigned char Color)
	功能介绍	Font_NUM　字体选择,以驱动所带的字库为准 Color　文本颜色,仅作用于 ASCII 字库
汉字库字符字体设置	函数名	void FontSet_cn(unsigned char Font_NUM,unsigned char Color)

功能	函数介绍	
	功能介绍	Font_NUM 字体选择,以驱动所带的字库为准 Color 文本颜色,仅作用于汉字库
写入一个 ASCII 字符	函数名	void PutChar(unsigned char x,unsigned char y,unsigned char a)
	功能介绍	x X 轴坐标 y Y 轴坐标 a 要显示的字符的 ASCII 码
ASCII 字符串显示	函数名	void PutString(unsigned char x,unsigned char y,char *p)
	功能介绍	x X 轴坐标 y Y 轴坐标 p 要显示的字符串
写入一个汉字字符	函数名	void PutChar_cn(unsigned char x,unsigned char y,char * GB)
	功能介绍	x X 轴坐标 y Y 轴坐标 GB 指向一个 GB 码的指针
汉字字符串显示	函数名	void PutString_cn(unsigned char x,unsigned char y,char *p)
	功能介绍	x X 轴坐标 y Y 轴坐标 p 要显示的字符串
绘图色设置	函数名	void SetPaintMode(unsigned char Mode,unsigned char Color)
	功能介绍	Mode 绘图模式（保留，没有使用到） Color 像素点的颜色,相当于前景色
绘点	函数名	void PutPixel(unsigned char x,unsigned char y)
	功能介绍	x X 轴坐标 y Y 轴坐标
绘制直线	函数名	void Line(unsigned char s_x,unsigned char s_y,unsigned char e_x,unsigned char e_y)
	功能介绍	s_x、s_y 为起始坐标 e_x、e_y 为结束坐标
绘制矩形	函数名	void Rectangle(unsigned char left, unsigned char top, unsigned char right, unsigned char bottom, unsigned char mode)
	功能介绍	left – 矩形的左上角横坐标，范围 0 到 126 top – 矩形的左上角纵坐标，范围 0 到 62 right – 矩形的右下角横坐标，范围 1 到 127 bottom – 矩形的右下角纵坐标，范围 1 到 63

功能	函数介绍	
		Mode – 绘制模式，可以是下列数值之一： 0: 矩形框（空心矩形） 1: 矩形面（实心矩形）
绘制圆形	函数名	void Circle(unsigned char x,unsigned char y,unsigned char r,unsigned char mode)
	功能介绍	以 x,y 为圆心 R 为半径画一个圆(mode = 0) or 圆面(mode = 1)
无符字符型数显示	函数名	void ShowChar(unsigned char x,unsigned char y,unsigned char Num,unsigned char type)
	功能介绍	x X 轴坐标 y Y 轴坐标 Num 要显示的 8 位宽度的数据 type 显示特性(0,1,2)
无符短整型数显示	函数名	void ShowShort(unsigned char x,unsigned char y,unsigned short Num,unsigned char type)
	功能介绍	x X 轴坐标 y Y 轴坐标 Num 要显示的 16 位宽度的数据 type 显示特性(0,1,2)
位图显示	函数名	void PutBitmap(unsigned char x,unsigned char y,unsigned char width, unsigned char high,unsigned char *p)
	功能介绍	x X 轴坐标 y Y 轴坐标 width 位图的宽度 high 位图的高度 p 要显示的位图的字模首地址
设置整屏显示灰度	函数名	void SetLCDGra(unsigned char Dat)
	功能介绍	LCD 整屏显示灰度调节 Dat 要调成的灰度值,0~0x3f 调节后的灰度将会保持到下一次 LCD 复位之前,或者下一次重新调整灰度之前。
设置模块进入省电模式	函数名	void SetLCDSleep(unsigned char Dat)
	功能介绍	设置模块进入低功耗模式 Dat 0xaa 进入低功耗模式后,屏上保留显示 0x55 进入低功耗模式后,屏上关闭显示

功能	函数介绍	
退出省电模式	函数名	void SetLCDWakeup(void)
	功能介绍	设置模块退出低功耗模式