MzLH04-12864 模组

产品说明书

V1.0 – 2009.02

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

1 模块简介

1.1 特点

MzLH04-12864 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。

1. 128 × 64 点阵 FSTN ；
2. 串行 SPI 接口方式（仅写入）；
3. 自带 12×12 点和 16×16 点汉字库（包含一级和二级汉字库）；
4. 自带 6×10、8×16 点 ASCII 码西文字库（96 个字符）；
5. 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）；
6. 自带整型数显示功能，直接输入整型数显示，而无需作变换；
7. 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）。

1.2 主要功能与基本参数

MzLH04 模块的基本参数如下表：

显示模式	FSTN 液晶	灰模
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）×8（高）mm	不包含直插针高度
视屏尺寸	33.26（长）×20.46（宽）mm	
点大小	0.23（宽）×0.29（长）mm	
像素尺寸	0.26（宽）×0.32（长）	
背光	蓝色 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	—	2.4	3.3	3.6	V
输入电平	High Level	Vih	—	0.8Vdd	Vdd	5	V
	Low Level	Vil		Vss		0.2Vdd	
模块正常工作电流		Ilcd	—	4.5	5.5	5.8	mA
背光 LED 电流		Iled	—	0	—	10	mA

MzLH04 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。

1.3 结构及引脚示意

图 1.1 为 MzLH04-12864 模块中的结构尺寸示意图。

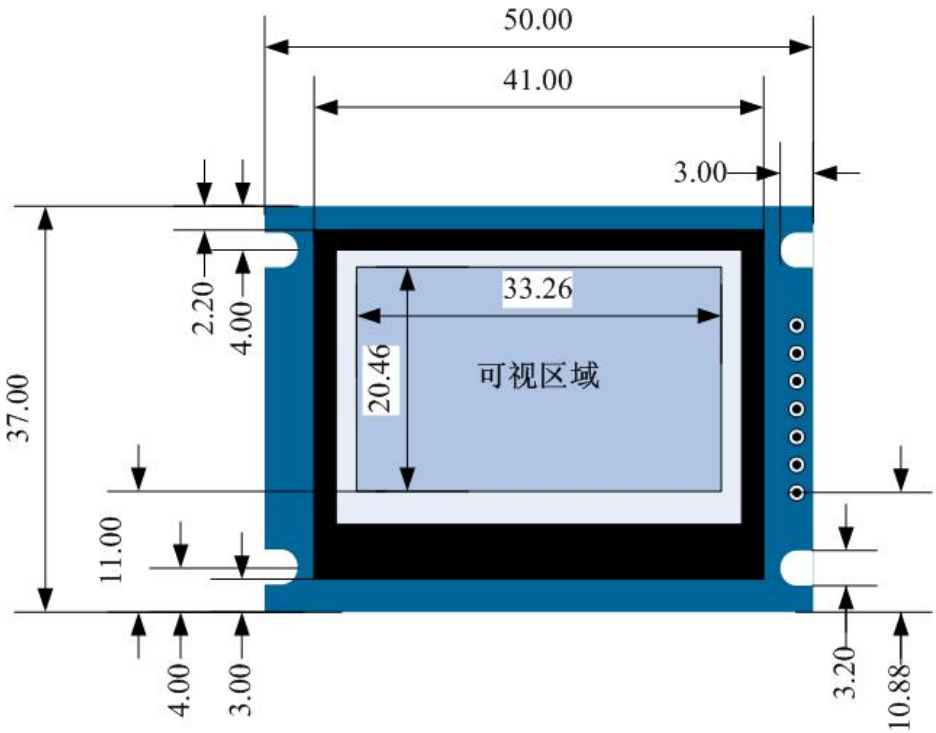


图 1.1 LCD 结构尺寸

MzLH04-12864 其正面和背面的照片如以下两图所示：

图 1.2 模块实物效果图

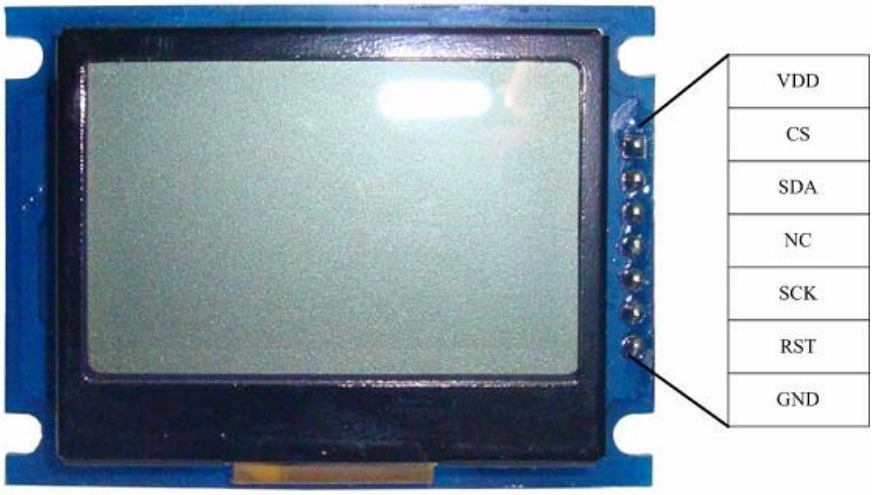


图 1.3 模块引脚分布示意图

表 1.1 模块接口引脚说明

序号	接口引脚名	说明
1	VCC	LCD 供电
2	CS	片选
3	SDA	数据输入线
4	NC	NC
5	SCK	SPI 时钟线(<2MHz 的速度)
6	RST	模块复位线(低电平复位)
7	GND	LCD 接地

1.4 模块显示效果

下面提供一些参考的显示效果图（照片中的效果图均打开了背光，请以实物为准，照片仅供参考）：

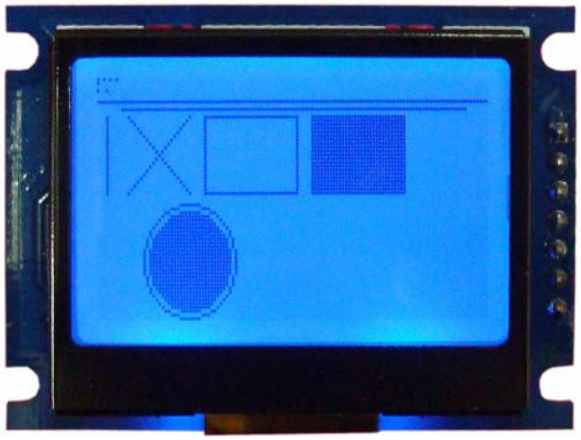


图 1.4 模块基本功能展示

图 1.5 字符显示效果

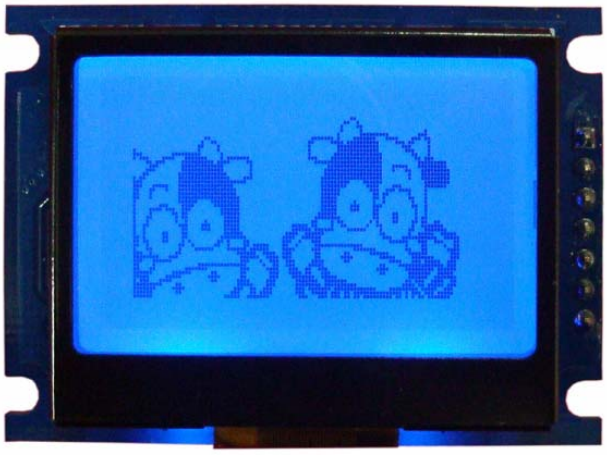


图 1.6 位图显示效果

2 液晶显示器介绍

2.1 显示屏坐标点映射关系

MzLH04 模块的显示面板上，共分布着 128×64 个单色像素点，每个像素点均与模块中的显示控制器中的显存有着对应关系，控制器中共有 1K byte 的显存映射着整个屏幕的显示区域；不过与一般的单色液晶模块不同，MzLH04 模块内部的显示控制器是不需要用户来直接控制这些显存的数据来改变显示的画面的，而只需要通过控制器所提供的各种接口指令就可以完成各种显示了。

MzLH04 将显示区域性以 X 和 Y 轴进行二维的坐标划分，将横向和纵向以 X 轴地址（X Address）和 Y 轴地址（Y Address）表示，分别可以寻址的范围为 X Address=0~127，Y Address = 0~63，X Address 和 Y Address 交叉对应着一个像素点。

MzLH04 模块的像素点地址关系如图 2.1所示：

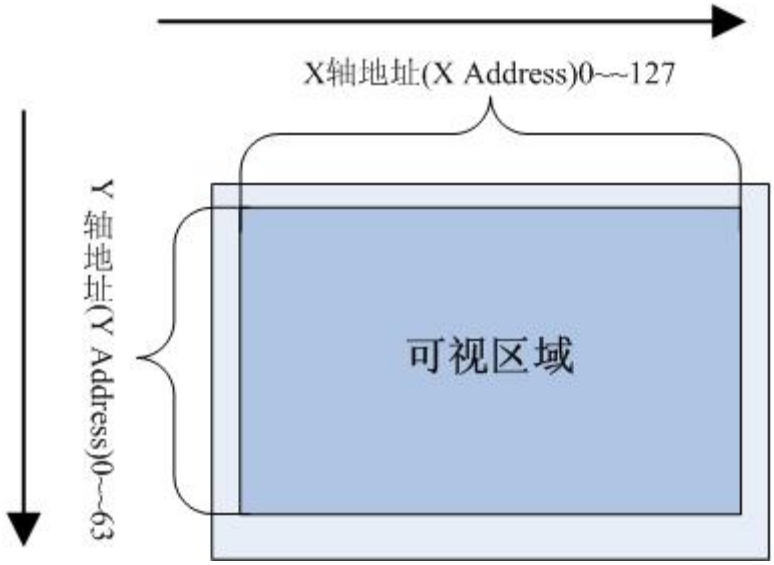


图 2.1 显示区域像素点排列情况

如上图所示 MzLH04 模块的 X、Y 轴原点位置在正视屏幕时的左上角。

3 模块控制方法

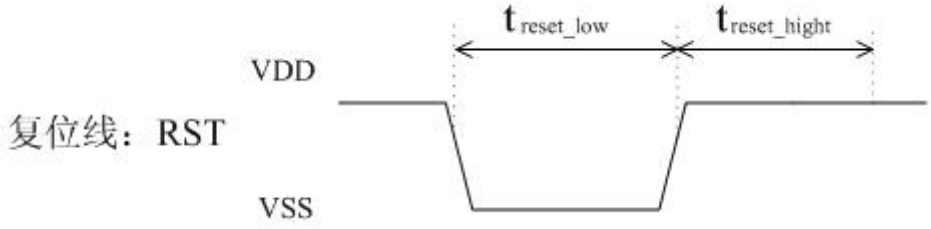
MzLH04 模块当中有一个 LCD 显示控制器，以及存放字库的存储器，用户在使用该模组时，一般只需要使用串行 SPI 接口对控制器写入规定的指令以及指令的数据，即可完成绘制像素点、画直线、实心矩形、矩形框或者是实心圆、圆框，以及在屏幕上的任意位置显示西文（ASCII 字符）、一二级汉字库中的汉字。当然，如果用户需要显示自己提取的汉字字模或者是位图图像，也可以通过对控制器写入位图指令及相关的尺寸数据后，跟着连续写入要显示的字模的数据即可。

下面介绍 MzLH04 模块的控制时序，以及相关的指令以及详细控制方法。

3.1 操作时序

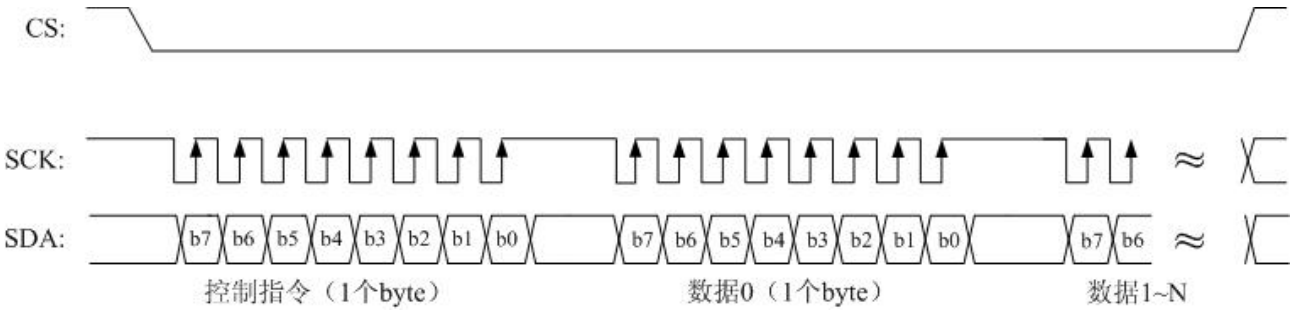
3.1.1 模块复位时序

MzLH04 模块有一个复位引脚，可以对该引脚输入一个低电平的脉冲使模组复位，复位需要低电平输入持续至少 2ms，在恢复高电平后需要等待 10ms 后方可对模组进行显示的控制操作（即通过串行接口输入指令和数据）。模组复位不正常时，将无法正常工作。



3.1.2 串行指令/数据写入时序

MzLH04 模块串行 SPI 接口，时钟频率必须**低于 2MHz**，指令以及数据的写入时序相同，时序图如下：



在通过串行 SPI 对模组进行控制时，CS 为从机选择线；CS 为低电平时，模组准备接收串行通讯的控制指令或数据，模块对 SDA 的采样在每个时钟线 SCK 的上升沿，当 CS 变为高电平后传输是无效的。

用户在使用时，需要注意 MzLH04 模块是没有忙信号（Busy）输出的，而为了抵消 LCD 模块上的显示控制器处理显示指令及数据与串行端口上接收到的数据之间的速度差，控制器内部开辟了 400 个字节的缓冲区；这样就可以保证用户通过串行端口给模块传输绘图或字符显示的指令及数据时，不需要等待模块上的显示控制器处理这些指令及数据，可以连续的将指令及数据传输给模块。但是毕竟缓冲区是有限的，并且模块对一些显示的操作相对较慢，所以用户在连续给模块传输指令及数据时需要注意一下，尽量不要不停的给模块送指令及数据，否则将有可能会产生指令/数据丢失，从而造成不可预测的结果。不过在通常情况之下，在用户连续写入控制指令/数据的个数不超过 400 个 byte 的情况下，模块都可以正常的显示及

操作。

而当需要连续的传输大量显示数据时（如位图显示），尽量在程序中控制好传输的时间。

3.2 控制指令简介

MzLH04 模组提供了两大类控制指令，主要分类如下：

- 液晶设置指令：比如一些清屏、字符选择、背光控制等指令；
- 基本显示功能控制指令：完成基本的绘图、字符显示功能，如绘制直线、矩形，显示字符等；

所有的用户操控 MzLH04 模块来完成显示功能都是通过给模块发送指令+数据的方式完成的，控制数据包指的是指令码+数据（不定长度，由具体的指令而定）。

下面，介绍 MzLH04 模块的指令。

液晶设置指令列表：

控制指令	功能	数据个数	数据解释（均为写入）	备注
0x80	清屏	0	无	
0x81	设置 ASCII 字符类型	1	Data1 高四位：ASCII 字符的字号 ● Data1_hight=0 时选择 6X10 字体； ● Data1_hight=1 时选择 8X16 字体； Data1 低四位：字符的前景色 ● Data1_low=0 字符为白色； ● Data1_low=1 字符为黑色。	
0x82	设置汉字库类型	1	Data1 高四位：汉字字符的字号 ● Data1_hight =0 时选择 12X12 字体； ● Data1_hight =1 时选择 16X16 字体； Data1 低四位：字符的前景色 ● Data1_low =0 字符为白色； ● Data1_low =1 字符为黑色。	
0x83	设置绘图色	1	Data1：为零时表示绘图将以白色进行绘制，为 1 即为黑色	仅作用于点、直线、矩形、圆的绘制操作
0x89	设置字符覆盖模式	1	Data1 高四位：字符覆盖模式 ● Data1_hight=0 关闭字符覆盖模式 ● Data1_hight=1 打开字符覆盖模式（默认） Data1 低四位：字符覆盖模式下背景填充色（0 或 1）	该指令仅对 ASCII 以及汉字字符显示起作用。

控制指令	功能	数据个数	数据解释（均为写入）	备注
0x8A	设置背光显示的亮度	1	Data1：0~127 的数值，表示要设置的背光亮度，0 为关闭背光，127 为最亮	

基本显示功能控制指令：

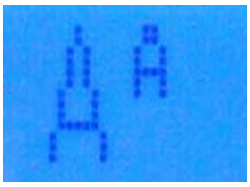
控制指令	功能	数据个数	数据解释（无特殊说明的均为写入）	备注
0x01	绘点	2	Data1：要绘制的点 X 轴坐标 Data2：要绘制的点 Y 轴坐标	以绘图色为前景色进行绘制
0x02	画直线	4	Data1：直线起点 X 轴坐标 Data2：直线起点 Y 轴坐标 Data3：直线终点 X 轴坐标 Data4：直线终点 Y 轴坐标	以绘图色为前景色进行绘制
0x03	画矩形框	4	Data1：左上角 X 轴坐标 Data2：左上角 Y 轴坐标 Data3：右下角 X 轴坐标 Data4：右下角 Y 轴坐标	以绘图色为前景色进行绘制
0x04	画矩形	4	Data1：左上角 X 轴坐标 Data2：左上角 Y 轴坐标 Data3：右下角 X 轴坐标 Data4：右下角 Y 轴坐标	以绘图色为前景色进行绘制
0x05	画圆框	3	Data1：圆心 X 轴坐标 Data2：圆心 Y 轴坐标 Data3：圆半径	以绘图色为前景色进行绘制
0x06	画圆	3	Data1：圆心 X 轴坐标 Data2：圆心 Y 轴坐标 Data3：圆半径	以绘图色为前景色进行绘制
0x07	显示 ASCII 字符	3	Data1：字符左上角的 X 轴坐标 Data2：字符左上角的 Y 轴坐标 Data3：ASCII 码	ASCII 码中，0~31 无效
0x08	显示汉字	4	Data1：字符左上角的 X 轴坐标 Data2：字符左上角的 Y 轴坐标 Data3：一、二级汉字库中 GB 码的高八位 Data4：一、二级汉字库中 GB 码的低八位	
0x0B	在指定位置显示	4	Data1：左上角 X 轴坐标	

控制指令	功能	数据个数	数据解释（无特殊说明的均为写入）	备注
	unsigned char 数据		Data2：左上角 Y 轴坐标 Data3：要显示的数据(unsigned char 型) Data4：显示属性（0、1 或 2） 该指令显示的数据占用点数受 ASCII 码字号选择控制，要占用 3 个 ASCII 字符宽度；同时也受字符覆盖模式控制。	
0x0C	在指定位置显示 unsigned short 数据	5	Data1：左上角 X 轴坐标 Data2：左上角 Y 轴坐标 Data3：要显示的数据高八位 Data4：要显示数据的低八位 Data5：显示属性（0、1 或 2） 该指令显示的数据占用点数受 ASCII 码字号选择控制，要占用 5 个 ASCII 字符宽度；同时也受字符覆盖模式控制。	
0x0E	位图显示	>4	Data1：显示位图的起始位置 X 轴坐标 Data2：显示位图的起始位置 X 轴坐标 Data3：位图的横向点数大小 Data4：位图的纵向点数大小 Data5~N：位图的字模数据 位图数据为字模工具对位图取模获得的，都有宽度和高度的点数信息，该指令后面的数据个数是以指令数居中的 Data3 和 Data4 来决定的，取模方式为横向取模，字节顺序。	以绘图色为前景色进行绘制

3.3 模块控制指令详述

3.3.1 字符型号设置

MzLH04 模块自带有 2 种字号的 ASCII 字符，以及 2 种字号的汉字字符，这些都可以通过相对应的控制指令完成设置；如下图就展示了模块 2 种字号的 ASCII 字符效果：



0x81 指令为设置 ASCII 码显示时的字号，以及字符色，上图展示的分别为字号 0、1 的 ASCII 字符‘A’；字符色为 1 的显示效果。

- 0 表示选择 6X10 字体；
- 1 表示选择 8X16 字体；

该指令设置后，影响指令编码为 0x07、0x0C、0x0B 的显示控制指令。

0x82 指令为设置汉字显示的字号，以及字符色（该字符色与 0x81 指令中的作用效果一样）；汉字字号选择可以为 0 或者 1。

- 0 表示选择 12X12 字体；
- 1 表示选择 16X16 字体；

该指令设置后，影响指令编码为 0x08 的显示控制指令。

3.3.2 字符覆盖模式设置

MzL04 模块提供了一个字符覆盖模式的设置，该指令提供了字符显示与原 LCD 显示的背景叠加的设置；指令需要有一个字节的数据，其高四位与低四位分别有不同的用途，高四位为覆盖模式，可以为 0 或者 1（默认为 1），低四位为覆盖模式下字符显示的背景擦除时使用的颜色（由于是单色液晶，所以仅为 0 或 1）。

字符覆盖模式为 0 时表示在显示字符时，对原字符显示位置区域的背景图像不作擦除，仅显示该字符的字符线条（字符线条的颜色由 0x81 或 0x82 指令设置）；而当字符覆盖模式设置为 1 时，在显示字符时将会使用指令设置的覆盖模式背景擦除色对该字符的显示区域进行擦除后才以当前设置的字符色来显示字符线条。

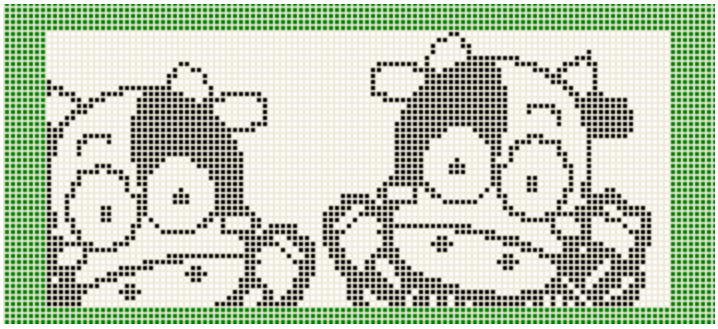
如下图所示，所显示的 ASCII 码字符’B’为选择字号为 1，字符色为 0 时，而字符覆盖模式关闭时的效果（在一片黑色的背景上显示）；而所显示的字符’a’，为 ASCII 字号为 1，字符色为 1，且字符覆盖模式使能（为 1）字符覆盖背景擦除色为 0 时的显示效果。

0x89 为字符覆盖模式指令码，该指令设置后影响 0x07、0x08、0x0B、0x0C、0x0E 的显示控制指令。

3.3.3 位图显示指令

MzLH04 模块提供了一个位图显示指令，指令要附带有 4 个 Byte 的必须的参数，分别设置了要显示的位图左上角起始位置，以及位图的宽度方向点数和高度方向的点数。这些必须的指令数据之后，跟随的是要显示的位图的字模数据，直至字模数据发送完毕后，该指令才执行完成。

该指令中，需要注意的是位图的宽度以及高度方向的点数定义，以及对应的位图的字模数据的提取；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：

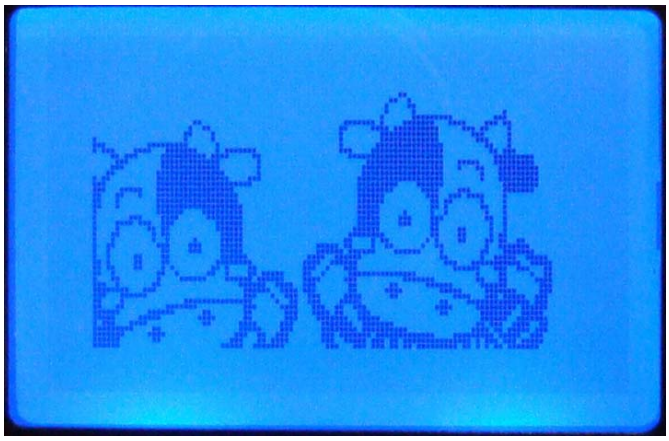


图中淡绿色的外框中的图像为一个宽度为 101 点，高度为 46 点的单色位图；选择取字模的方式为横向取模，可以得到的字模数据如下：

```
/*--  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,
0x00,0x00,0x00,0x00,0x00,0x64,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x86,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x82,
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
```

通过位图显示指令 0x0E，并设置好显示位图的左上角位置坐标，以及位图的宽度点数（该图为 101）、高度点数（46）后，显示的效果如下：



3.3.4 直接数字显示指令

大部分液晶模块的用户在使用液晶时，通常都会使用液晶来显示一些数据，为此，MzLH06 模块提供了两条指令 0x0B 和 0x0C 来实现 unsigned char(无符 8 位的字符数)和 unsigned short（无符短整数）的直接显示，用户仅需要通过这两条指令并且将数据传送给模块，便可以在指定的位置上显示出所输入的数据。

0x0B 的指令有 4 个字节参数，第一个字节为要显示的数字的左上角的起始 X 轴坐标，第二个节字为左上角的起始 Y 轴坐标，第三个字节为要显示的数据，最后一个字节数据为显示属性。该指令受前面的 ASCII 字符字号设置以及字符色的设置景响。

0x0B 指令显示的数据固定占用三个 ASCII 字符的位置，显示的属性可以设置为 0、1 或 2，如下所述：

- 属性 0：数字显示时从左到右将显示 unsigned char 型数的三个十进制位数据，比如用户显示 0 的数据，将在屏幕上显示 000，如显示 12，则显示出来 012；
- 属性 1：数字显示时，从左到右将显示 unsigned char 型数的有效十进制位数据，并且数据显示靠左基准开始显示（左对齐），比如用户要显示 0 数据时，则显示一个 0 在该指令所设定的起

始位置的显示区域，如要显示数据 12，则在屏幕上显示 12；

- 属性 2：数字显示时，从左到右将显示 unsigned char 型数的有效十进制位数据，并且数据显示告右基准（右对齐）。

而另一条指令 0x0C 与 0x0B 类似，只不过 **0x0C 指令**是实现显示一个 16 位宽度的无符短整型数功能而已，其指令所要求的参数比 0x0B 多一个字节，而其固定占了 5 个 ASCII 字符位置；它的属性设置与 0x0B 一样。

4 设计参考

下面将介绍一个简单的单片机对 MzLH04 进行操作的范例（以 MCS51 单片机的程序为例）。

4.1 与模块通讯的接口函数

MzLH04 模块的接口函数非常简单，如下：

```
#include "REG52.h"


//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
//=========================================================================
// 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 2ms 左右
}
//============================================================================
// 函数: void SPI_SSSet(unsigned char Status)
// 描述: 置 SS 线状态
// 参数: Status    =1 则置高电平，=0 则置低电平
// 返回: 无
//============================================================================
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;
}
//============================================================================
// 函数: void SPI_Send(unsigned char Data)
// 描述: 通过串行 SPI 口输送一个 byte 的数据置模组
// 参数: Data 要传送的数据
// 返回: 无
//============================================================================
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
        }
}
```

TimeDelay 函数是一个软件延时程序，在上面的代码中，并没有很精确的计算其延时的时长，本例仅作为一个大概的时间延时，用户在参考本例子程序时，请根据自己使用的 MCU 的情况来定夺。

4.2 操作示例

接下来是一些功能函数操作的简单示例：

```
//=============================================================================
// 函数: void ClrScreen(void)
// 描述: 清屏函数
// 参数:
// 返回: 无
//=============================================================================
void ClrScreen(void)
{
        //清屏操作
        SPI_SSSet(0);                   //SS 置低电平
        SPI_Send(0x80);                 //送指令 0x80
        SPI_SSSet(1);                   //完成操作置 SS 高电平
}
//=============================================================================
// 函数: void SetBackLight(unsigned char Deg)
// 描述: 设置背光亮度等级
// 参数: Grade   0~127 的等级
// 返回: 无
//=============================================================================
void SetBackLight(unsigned char Deg)
{
        //显示 ASCII 字符
        SPI_SSSet(0);                   //SS 置低电平
        SPI_Send(0x8a);                 //传送指令 0x8A
        SPI_Send(Deg);                  //背光亮度
        SPI_SSSet(1);                   //完成操作置 SS 高电平
}
```

更多的功能函数请参考例程，以及例程所附带的说明文档。

5 技术支持

铭正同创在 LCD 显示驱动方面有着丰富的经验，并通过积累总结出通用的成熟的 LCD 驱动程序架构，可方便的移植到不同的 LCD 应用上，或者移植到不同的 MCU 应用平台上。随着产品的推出，将会推出更多的完整的驱动程序，这些代码或应用开发软件用户可以放心的使用到产品当中。

而在人机界面编程方面，铭正同创也会给用户提供更多的参考，或者为用户订制专用的设计，使铭正同创的用户在得到质优价廉的产品的同时，也获得专业的技术支持。

铭正同创努力的目标是作为一个行业的**产品+技术的专业提供者**。

5.1 联系方式

- 北京铭正同创科技有限公司
- 联系电话：**010-82790200** （销售/技术）
- 公司邮箱：**mzdesign@mzdesign.com.cn**
- 公司网址：**http://www.mzdesign.com.cn**