诺基亚 5110 屏幕显示驱动（字符、汉字、图片）

对比一 5110 跟 12864 和 1602,5110 简直是太方便了，便宜好用，传送速度是 12864 和 1602 的几十倍，可高达 40Mbps, 低功耗，正常显示时电流在 200uA 一下，工作电压 3.3V，可以直接用 MSP430 来驱动，可以显示 15 个汉字，30 个字符，汉字可以显示 4 行。

5110 不带字库，所以要显示的字符或者汉字都需要提取字模才可以的，显示方式跟 64X32 点阵很像，这个等会再说。

右边是引脚图，5110 的引脚只有 8 个，除去电源输入、背光输入、GND、REST 引脚，只剩下四个引脚接到单片机就可以了。可以直接把 RST 置高，跟 12864 一样置高后复位功能不能使用。D/C 引脚是选择数据或命令的，这跟 12864 不太一样。BL 是背光灯，5110 都是从旧机器上拆下来的，商家通常会在后边的 PCB 板上加上两个 LED 灯 ,BL 一置高灯就可以亮了 ,不用灯的话 BL 引脚可以空着。VCC 接 3.3V，剩下的 CLK、DIN、DC、CE 引脚接 IO 口。

时序还是 SPI，跟 12864 一样的：

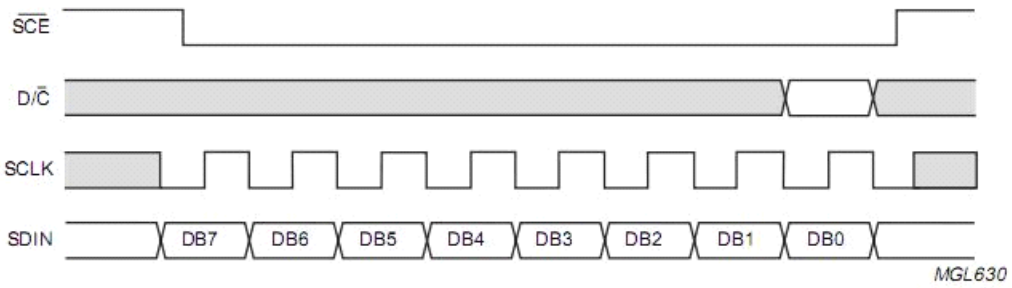


图10 串行总线协议——传送1个字节

程序：

```
#include <msp430g2553.h>

#define uint unsigned int
#define uchar unsigned char

#define LCD_CE BIT3
#define LCD_DC BIT4
#define SDIN BIT5
#define SCLK BIT6
#define REST BIT2

unsigned char const code[][6] =
{
    { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 },   // sp
    { 0x00, 0x00, 0x00, 0x2f, 0x00, 0x00 },   // !
    { 0x00, 0x00, 0x07, 0x00, 0x07, 0x00 },   // "
    { 0x00, 0x14, 0x7f, 0x14, 0x7f, 0x14 },   // #
    { 0x00, 0x24, 0x2a, 0x7f, 0x2a, 0x12 },   // $
    { 0x00, 0x62, 0x64, 0x08, 0x13, 0x23 },   // %
    { 0x00, 0x36, 0x49, 0x55, 0x22, 0x50 },   // &
    { 0x00, 0x00, 0x05, 0x03, 0x00, 0x00 },   // '
    { 0x00, 0x00, 0x1c, 0x22, 0x41, 0x00 },   // (
    { 0x00, 0x00, 0x41, 0x22, 0x1c, 0x00 },   // )
    { 0x00, 0x14, 0x08, 0x3E, 0x08, 0x14 },   // *
    { 0x00, 0x08, 0x08, 0x3E, 0x08, 0x08 },   // +
    { 0x00, 0x00, 0x00, 0xA0, 0x60, 0x00 },   // ,
    { 0x00, 0x08, 0x08, 0x08, 0x08, 0x08 },   // -
    { 0x00, 0x00, 0x60, 0x60, 0x00, 0x00 },   // .
    { 0x00, 0x20, 0x10, 0x08, 0x04, 0x02 },   // /
    { 0x00, 0x3E, 0x51, 0x49, 0x45, 0x3E },   // 0
    { 0x00, 0x00, 0x42, 0x7F, 0x40, 0x00 },   // 1
```

```
{ 0x00, 0x42, 0x61, 0x51, 0x49, 0x46 },   // 2
{ 0x00, 0x21, 0x41, 0x45, 0x4B, 0x31 },   // 3
{ 0x00, 0x18, 0x14, 0x12, 0x7F, 0x10 },   // 4
{ 0x00, 0x27, 0x45, 0x45, 0x45, 0x39 },   // 5
{ 0x00, 0x3C, 0x4A, 0x49, 0x49, 0x30 },   // 6
{ 0x00, 0x01, 0x71, 0x09, 0x05, 0x03 },   // 7
{ 0x00, 0x36, 0x49, 0x49, 0x49, 0x36 },   // 8
{ 0x00, 0x06, 0x49, 0x49, 0x29, 0x1E },   // 9
{ 0x00, 0x00, 0x36, 0x36, 0x00, 0x00 },   // :
{ 0x00, 0x00, 0x56, 0x36, 0x00, 0x00 },   // ;
{ 0x00, 0x08, 0x14, 0x22, 0x41, 0x00 },   // <
{ 0x00, 0x14, 0x14, 0x14, 0x14, 0x14 },   // =
{ 0x00, 0x00, 0x41, 0x22, 0x14, 0x08 },   // >
{ 0x00, 0x02, 0x01, 0x51, 0x09, 0x06 },   // ?
{ 0x00, 0x32, 0x49, 0x59, 0x51, 0x3E },   // @
{ 0x00, 0x7C, 0x12, 0x11, 0x12, 0x7C },   // A
{ 0x00, 0x7F, 0x49, 0x49, 0x49, 0x36 },   // B
{ 0x00, 0x3E, 0x41, 0x41, 0x41, 0x22 },   // C
{ 0x00, 0x7F, 0x41, 0x41, 0x22, 0x1C },   // D
{ 0x00, 0x7F, 0x49, 0x49, 0x49, 0x41 },   // E
{ 0x00, 0x7F, 0x09, 0x09, 0x09, 0x01 },   // F
{ 0x00, 0x3E, 0x41, 0x49, 0x49, 0x7A },   // G
{ 0x00, 0x7F, 0x08, 0x08, 0x08, 0x7F },   // H
{ 0x00, 0x00, 0x41, 0x7F, 0x41, 0x00 },   // I
{ 0x00, 0x20, 0x40, 0x41, 0x3F, 0x01 },   // J
{ 0x00, 0x7F, 0x08, 0x14, 0x22, 0x41 },   // K
{ 0x00, 0x7F, 0x40, 0x40, 0x40, 0x40 },   // L
{ 0x00, 0x7F, 0x02, 0x0C, 0x02, 0x7F },   // M
{ 0x00, 0x7F, 0x04, 0x08, 0x10, 0x7F },   // N
{ 0x00, 0x3E, 0x41, 0x41, 0x41, 0x3E },   // O
{ 0x00, 0x7F, 0x09, 0x09, 0x09, 0x06 },   // P
{ 0x00, 0x3E, 0x41, 0x51, 0x21, 0x5E },   // Q
{ 0x00, 0x7F, 0x09, 0x19, 0x29, 0x46 },   // R
{ 0x00, 0x46, 0x49, 0x49, 0x49, 0x31 },   // S
{ 0x00, 0x01, 0x01, 0x7F, 0x01, 0x01 },   // T
{ 0x00, 0x3F, 0x40, 0x40, 0x40, 0x3F },   // U
{ 0x00, 0x1F, 0x20, 0x40, 0x20, 0x1F },   // V
{ 0x00, 0x3F, 0x40, 0x38, 0x40, 0x3F },   // W
{ 0x00, 0x63, 0x14, 0x08, 0x14, 0x63 },   // X
{ 0x00, 0x07, 0x08, 0x70, 0x08, 0x07 },   // Y
{ 0x00, 0x61, 0x51, 0x49, 0x45, 0x43 },   // Z
{ 0x00, 0x00, 0x7F, 0x41, 0x41, 0x00 },   // [
{ 0x00, 0x55, 0x2A, 0x55, 0x2A, 0x55 },   // 55
{ 0x00, 0x00, 0x41, 0x41, 0x7F, 0x00 },   // ]
```

```
    { 0x00, 0x04, 0x02, 0x01, 0x02, 0x04 },   // ^
    { 0x00, 0x40, 0x40, 0x40, 0x40, 0x40 },   // _
    { 0x00, 0x00, 0x01, 0x02, 0x04, 0x00 },   // '
    { 0x00, 0x20, 0x54, 0x54, 0x54, 0x78 },   // a
    { 0x00, 0x7F, 0x48, 0x44, 0x44, 0x38 },   // b
    { 0x00, 0x38, 0x44, 0x44, 0x44, 0x20 },   // c
    { 0x00, 0x38, 0x44, 0x44, 0x48, 0x7F },   // d
    { 0x00, 0x38, 0x54, 0x54, 0x54, 0x18 },   // e
    { 0x00, 0x08, 0x7E, 0x09, 0x01, 0x02 },   // f
    { 0x00, 0x18, 0xA4, 0xA4, 0xA4, 0x7C },   // g
    { 0x00, 0x7F, 0x08, 0x04, 0x04, 0x78 },   // h
    { 0x00, 0x00, 0x44, 0x7D, 0x40, 0x00 },   // i
    { 0x00, 0x40, 0x80, 0x84, 0x7D, 0x00 },   // j
    { 0x00, 0x7F, 0x10, 0x28, 0x44, 0x00 },   // k
    { 0x00, 0x00, 0x41, 0x7F, 0x40, 0x00 },   // l
    { 0x00, 0x7C, 0x04, 0x18, 0x04, 0x78 },   // m
    { 0x00, 0x7C, 0x08, 0x04, 0x04, 0x78 },   // n
    { 0x00, 0x38, 0x44, 0x44, 0x44, 0x38 },   // o
    { 0x00, 0xFC, 0x24, 0x24, 0x24, 0x18 },   // p
    { 0x00, 0x18, 0x24, 0x24, 0x18, 0xFC },   // q
    { 0x00, 0x7C, 0x08, 0x04, 0x04, 0x08 },   // r
    { 0x00, 0x48, 0x54, 0x54, 0x54, 0x20 },   // s
    { 0x00, 0x04, 0x3F, 0x44, 0x40, 0x20 },   // t
    { 0x00, 0x3C, 0x40, 0x40, 0x20, 0x7C },   // u
    { 0x00, 0x1C, 0x20, 0x40, 0x20, 0x1C },   // v
    { 0x00, 0x3C, 0x40, 0x30, 0x40, 0x3C },   // w
    { 0x00, 0x44, 0x28, 0x10, 0x28, 0x44 },   // x
    { 0x00, 0x1C, 0xA0, 0xA0, 0xA0, 0x7C },   // y
    { 0x00, 0x44, 0x64, 0x54, 0x4C, 0x44 },   // z
    { 0x14, 0x14, 0x14, 0x14, 0x14, 0x14 }    // horiz lines
};

/*
 *                位 传 送
 */
void Write_Byte(unsigned char dat, unsigned char command)
{
    unsigned char i;
    P2OUT &= ~LCD_CE;              //LCD片选置低传送
    if (command == 0)
        P2OUT &= ~LCD_DC;          //选择传送命令
    else
        P2OUT |= LCD_DC;           //选择传送数据
    for(i=0; i<8; i++)
```

```
    {
        if (dat & 0x80)
            P2OUT |= SDIN;          //SDIN = 1;
        else
            P2OUT &= ~SDIN;             //SDIN = 0;
        P1OUT &= ~SCLK;                 //SCLK = 0;
        dat = dat << 1;
        P1OUT |= SCLK;              //SCLK = 1;
    }
    P2OUT |= LCD_CE;
}

/*
 *                  清 屏
 */

void LCD_Clear(void)
{
    unsigned int i;
    Write_Byte(0x0c, 0);            //正常显示模式，0x0c显示黑字，背景为白色，
0x0d显示白字，黑色背景。
    Write_Byte(0x80, 0);
    for (i=0; i<504; i++)
        Write_Byte(0, 1);
}

/*
 *              LCD 初 始 化 函 数
 */
void LCD_Init(void)
{
  // P2OUT &= BIT2;                   //产生一个让LCD复位的低电平脉冲
   _delay_cycles(100);              //延时大于100ns
  // P2OUT |= BIT2;

   P2OUT |= LCD_CE;                 //LCD_CE = 1;     // 使能LCD
   _delay_cycles(100);

    Write_Byte(0x21, 0);        // 使用扩展命令设置LCD模式
    Write_Byte(0xc8, 0);        // 设置液晶偏置电压
  //  LCD_write_byte(0x06, 0);     温度校正
    Write_Byte(0x13, 0);        // 设置混合率 即对比度 1:48
    Write_Byte(0x20, 0);        // 使用基本命令，V=0，水平寻址
    LCD_Clear();                // 清屏
```

```
    _delay_cycles(100);

    P2OUT &= ~LCD_CE;//LCD_CE = 0;      // 关闭LCD
}


/*
 *              坐 标 设 置
 */
void Set_XY(unsigned char X, unsigned char Y)
{
    Write_Byte(0x40 | Y, 0);        // column
    Write_Byte(0x80 | X, 0);         // row
}



/*
 *              显 示 单 个 字 符
 */
void Write_Char(unsigned char c)
{
    unsigned char i;
    c -= 32;
    for (i=0; i<6; i++)
        Write_Byte(code[c][i],1);

}

/*
 *               显 示 字 符 串
 */
void LCD_String(uchar x, uchar y,char *s)
{
    Set_XY(x,y);
    _delay_cycles(100);
    while(*s)
    {
        Write_Char(*s);
        s++;
    }
}

/*
```

```
 *            显 示 汉 字
 */
void LCD_Chinese(uchar x, uchar y)
{
    uchar i,j;
    Set_XY(x,y);
    for(i=0; i<8; i+=2)
        for(j=0; j<16; j++)
            Write_Byte(code1[i][j],1);
    Set_XY(x,y+1);
    for(i=1; i<8; i+=2)
        for(j=0; j<16; j++)
            Write_Byte(code1[i][j],1);
}

/*
 *            显 示 图 片
 */
void LCD_Picture(uchar x, uchar y)
{
    uchar i,j;
    Set_XY(x,y);
    for(i=0; i<40; i++)
        Write_Byte(code2[i],1);
    Set_XY(x,y++);
    for(j=40; j<80; j++)
        Write_Byte(code2[j],1);
    Set_XY(x,y++);
    for(i=80; i<120; i++)
        Write_Byte(code2[i],1);
    Set_XY(x,y++);
    for(j=120; j<160; j++)
        Write_Byte(code2[j],1);
    Set_XY(x,y++);
    for(j=160; j<200; j++)
        Write_Byte(code2[j],1);
}
/*
 *            主 函 数
 */
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD; //关闭看门狗
    P1DIR |= BIT6 + BIT7;
```

```
    P2DIR |= BIT3 + BIT4 + BIT5;
    P1OUT &= BIT7;
    LCD_Init();
    LCD_String(0,0,"Passage1");
    _delay_cycles(100);
    //LCD_Chinese(0,2);
    //LCD_Picture(0,0);
    //_delay_cycles(100);
}
```

单片机中有 RAM 和 ROM，一般程序运行时产生的数据都是存储在 RAM 中的，一掉电后数据就消失了，ROM 中存储的数据具有只读（不可修改）属性，一般 ROM 要比 RAM 大的多，其实 ROM 就相当于电脑硬盘，RAM 相当于内存条吧（个人理解。。。），MSPG2 系列单片机只有 512Bytes 的 RAM ,16K 的 flash 存储器 ,flash 存储器结合了 RAM 和 ROM 的优点，flash 正在逐渐代替 ROM。如果定义的变量太多的话都存在 ROM 中会占用很大的内存，可能会发生类似于溢出的现象。在定义字符数组的时候可以在数组名前面加 const 这时候定义的数组是储存在 flash 中的。开头定义数组比较大，把它存在 flash 中。字符数组的排序是按照 ASII 码来排序的。

下面的位传送函数跟12864的串行传送方式一样。接着是清屏函数， 5110的控制芯片是PCD8544，可以看一下它的数据手册，下面是它一些的寄存器：

表1 指令集

指令	D/C	命令字								描述
		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
(H = 0 or 1)										
NOP	0	0	0	0	0	0	0	0	0	空操作
功能设置	0	0	0	1	0	0	PD	V	H	掉电控制：进入模式； 扩展指令设置（H）
写数据	1	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	写数据到显示 RAM
(H = 0)										
保留	0	0	0	0	0	0	1	X	X	不可使用
显示控制	0	0	0	0	0	1	D	0	E	设置显示配置
保留	0	0	0	0	1	X	X	X	X	不可使用
设置RAM的Y地址	0	0	1	0	0	0	Y_2	Y_1	Y_0	设置RAM的Y地址 $0 \le Y \le 5$
设置RAM的X地址	0	1	X_6	X_5	X_4	X_3	X_2	X_1	X_0	设置RAM的X地址 $0 \le X \le 83$
(H = 1)										
保留	0	0	0	0	0	0	0	0	1	不可使用
	0	0	0	0	0	0	0	1	X	不可使用
温度控制	0	0	0	0	0	0	1	TC_1	TC_0	设置温度系数 (TC_X)
保留	0	0	0	0	0	1	X	X	X	不可使用
偏置系统	0	0	0	0	1	0	BS_2	BS_1	BS_0	设置偏置系统 (BS_X)
保留	0	0	1	X	X	X	X	X	X	不可使用
设置V_{OP}	0	1	V_{OP6}	V_{OP5}	V_{OP4}	V_{OP3}	V_{OP2}	V_{OP1}	V_{OP0}	写V_{OP}到寄存器

表2 表1中的符号说明

BIT	0	1
PD	芯片是活动的	芯片处于掉电模式
V	水平寻址	垂直寻址
H	使用基本指令集	使用扩展指令集
D and E 00 10 01 11	 显示空白 普通模式 开所有显示段 反转映象模式	
TC_1 and TC_0 00 01 10 11	 V_{LCD} 温度系数 0 V_{LCD} 温度系数 1 V_{LCD} 温度系数 2 V_{LCD} 温度系数 3	

要注意的是:

H=0 使用基本指令集命令，H=1 使用扩展指令集命令。

V=0 水平寻址，V=1 垂直寻址。

“D”和“E”是调整显示模式。

BS_2、BS_1、BS_0 是调整混合率的也就是对比度，可以通过改变程序来改变对比度，不要外接可调电阻就行。

5110 其实是 48*84 的点阵，传送数据的时候我们是把数据传送到它的 RAM 中去的，

寻址的的时候 x 是 0-84，y 是 0-5，y 中的一位数据其实是控制八行灯，0-5 六行控制 48 行灯，默认的传送模式就是先传送第一列的前八位数据，再返回来传送第二列的八位数据。。也就是说先传送 y 地址为 0，x 地址为 0 的八位数据，再回头传送 y=0，x=1 的八位数据。数据是低位在前，高位在上。

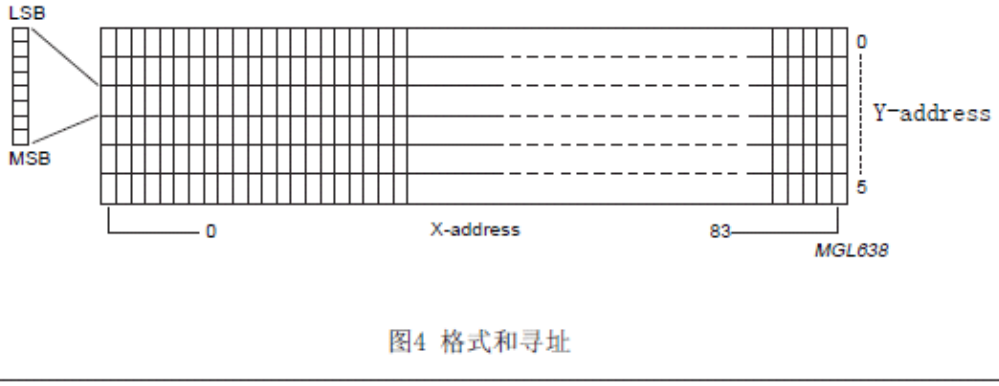


图4 格式和寻址

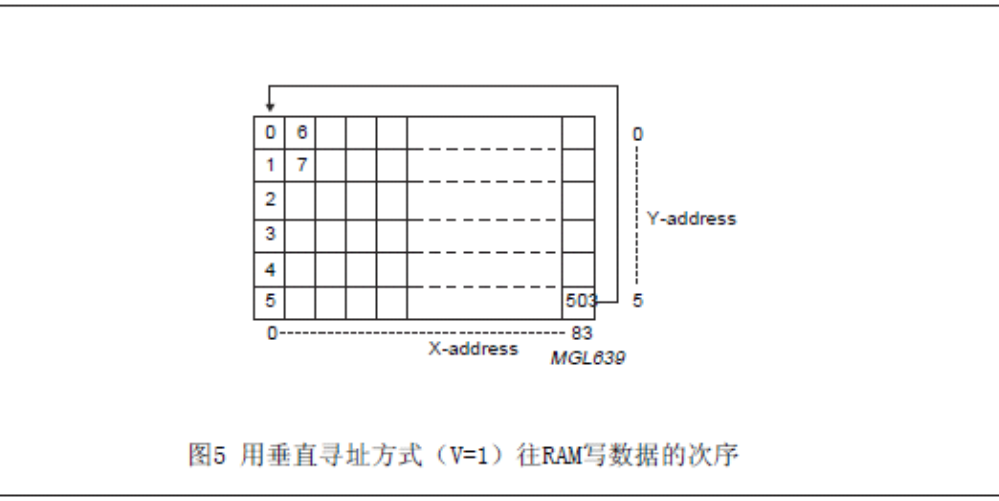
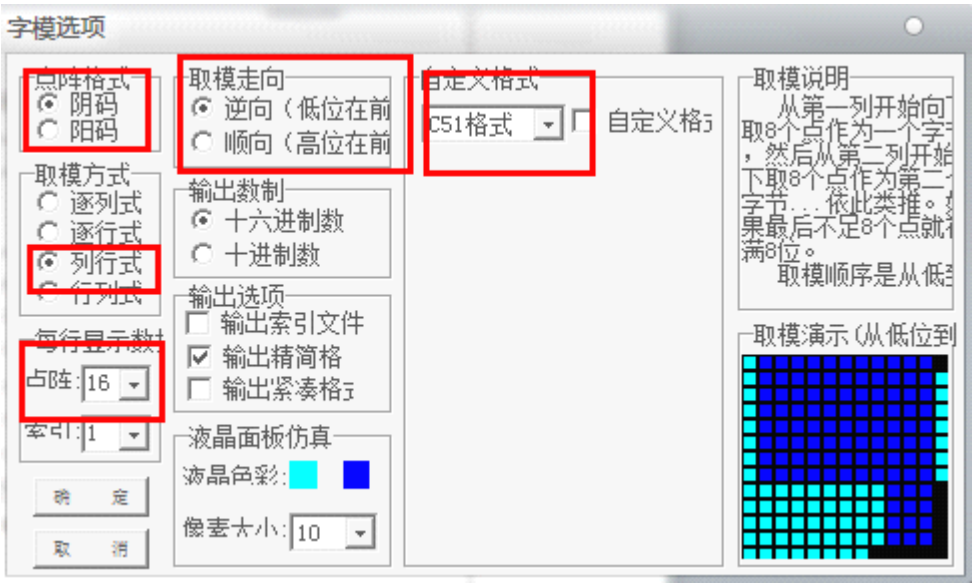


图5 用垂直寻址方式（V=1）往RAM写数据的次序

知道了数据传送模式，接下来就是取模了，可以用 PCtoLCD2002.exe 来取模，通过字宽字高来设置汉字在 LCD 上显示的大小，下面是取模软件的设置：



阴码就是亮点为 1，暗点为 0，取模走向是逆向（低位在前），方式是列行式，点阵就是生成的二位数组中一维数组元素的个数，点阵为 16 就是生成的二维数组的一维数组中元素个数是 16，当然也可以设置成 8，只不过函数不一样而已。

显示图片其实跟显示汉字是差不多的，扫描方式都是一样的，显示图片大小不能超过 84*48，要先转会为黑白色的。设置的时候点阵格式改为阳码，点阵根据图片大小来改。

Ps：此文仅为个人学习心得，难免有错漏之处，敬请谅解。

That's all

By 刘渠

2013. 4. 27 22 : 27