在上一篇文章中，我们详细介绍了如何驱动 LCD，对于在 LCD 屏上呈现各种简单的图形已经不是一件高不可攀的难事。但如何绘制字符呢？

其实每一字符就是一幅图像，字符的大小对应于图像的大小，字符的笔画对应于图像的内容。那么如何把字符转换为图像呢？简单的方法是使用“字模提取”之类的软件，它能够把任意的字符转换为一个字节型的数组，数组元素中的每一位代表 LCD 上的一个像素点，当为 1 时，表示该位置为字符的一个笔画，需要上色，而为 0 时，表示不是笔画，不需要上色。例如，一个字符想要在 16×16 的面积上显示，即该字符的宽和高各为 16 个像素，因为每一个像素用一位来表示，因此用字模提取软件生成的字节型数组，一共有 16×16÷8＝32 个字节。在字模提取的过程中，还要注意取模的顺序，顺序不同，得到的数组就不同，一般来说是从字符的左上角开始，从左向右，从上到下取模，这样程序编写上会方便一些。相同字体大小的中文字符和 ASCII 码字符的宽度还有所不同，一般 ASCII 码字符的宽度是中文字符宽度的一半，所以显示中文字符的程序和显示 ASCII 码字符的程序还略有不同。
当把一个字符取模变成一个数组后，只要对该数组中每个元素的每一位依次进行判断，对值为 1 的位和值为 0 的位进行不同的上色处理，即可完成一个字符的绘制。
下面的程序给出了一个简单的显示中文字符和 ASCII 字符的例子，我是用 PCtoLCD 这款软件来提取程序中想要显示的字符的。

```
#define U32 unsigned int

#define M5D(n)                                  ((n) & 0x1fffff)        // To get lower 21bits
#define rGPCCON       (*(volatile unsigned *)0x56000020)          //Port C control

#define rGPCDAT       (*(volatile unsigned *)0x56000024)           //Port C data

#define rGPCUP        (*(volatile unsigned *)0x56000028) //Pull-up control C
#define rGPDCON       (*(volatile unsigned *)0x56000030)          //Port D control
#define rGPDDAT       (*(volatile unsigned *)0x56000034)           //Port D data
#define rGPDUP        (*(volatile unsigned *)0x56000038)           //Pull-up control D
#define rGPGCON       (*(volatile unsigned *)0x56000060)          //Port G control
#define rGPGDAT       (*(volatile unsigned *)0x56000064)           //Port G data
#define rGPGUP        (*(volatile unsigned *)0x56000068)           //Pull-up control
#define rLCDCON1      (*(volatile unsigned *)0x4d000000)          //LCD control 1
#define rLCDCON2      (*(volatile unsigned *)0x4d000004)          //LCD control 2
#define rLCDCON3      (*(volatile unsigned *)0x4d000008)          //LCD control 3
#define rLCDCON4      (*(volatile unsigned *)0x4d00000c)          //LCD control 4
#define rLCDCON5      (*(volatile unsigned *)0x4d000010)          //LCD control 5
#define rLCDSADDR1    (*(volatile unsigned *)0x4d000014)        //STN/TFT Frame buffer start address 1
#define rLCDSADDR2    (*(volatile unsigned *)0x4d000018)        //STN/TFT Frame buffer start address 2
```

```
#define rLCDSADDR3     (*(volatile unsigned *)0x4d00001c)          //STN/TFT Virtual screen address set

#define rLCDINTMSK    (*(volatile unsigned *)0x4d00005c)       //LCD Interrupt mask

#define rTCONSEL         (*(volatile unsigned *)0x4d000060)      //LPC3600 Control --- edited by junon
#define LCD_WIDTH       320
#define LCD_HEIGHT    240

#define VSPW            (3-1)
#define VBPD            (15-1)
#define VFPD (12-1)
#define HSPW            (30-1)
#define HBPD            (38-1)
#define HFPD (20-1)
#define LINEVAL    (LCD_HEIGHT-1)
#define HOZVAL     (LCD_WIDTH-1)
//for LCDCON1
#define CLKVAL_TFT     6
#define MVAL_USED     0
#define PNRMODE_TFT          3
#define BPPMODE_TFT          13
//#define VIDEO_OUT    0
//for LCDCON5
#define BPP24BL     0
#define INVVCLK 0
#define INVVLINE            1
#define INVVFRAME     1
#define INVVD         0

#define INVVDEN 0
#define PWREN       1
#define BSWP            0
#define HWSWP       0
volatile U32 LCD_BUFFER[LCD_HEIGHT][LCD_WIDTH];



unsigned char zhao[]=                          //赵

{

0x08, 0x00, 0x08, 0x00, 0x08, 0x04, 0x7E, 0x84, 0x08, 0x48, 0x08, 0x28, 0xFF, 0x10, 0x08,
```

```
0x10,

0x28, 0x28, 0x2F, 0x28, 0x28, 0x44, 0x28, 0x84, 0x58, 0x00, 0x48, 0x00, 0x87, 0xFE, 0x00,
0x00

};

unsigned char chun[]=                    //春

{

0x01, 0x00, 0x01, 0x00, 0x3F, 0xFC, 0x01, 0x00, 0x1F, 0xF8, 0x02, 0x00, 0xFF, 0xFE, 0x04,
0x20,

0x08, 0x18, 0x3F, 0xEE, 0xC8, 0x24, 0x0F, 0xE0, 0x08, 0x20, 0x08, 0x20, 0x0F, 0xE0, 0x00,
0x00

};

unsigned char jiang[]=                   //江

{

0x20, 0x00, 0x10, 0x00, 0x13, 0xFC, 0x00, 0x40, 0x88, 0x40, 0x48, 0x40, 0x50, 0x40, 0x10,
0x40,

0x10, 0x40, 0x20, 0x40, 0xE0, 0x40, 0x20, 0x40, 0x20, 0x40, 0x2F, 0xFE, 0x20, 0x00, 0x00,
0x00,

};

unsigned char ASCII_A[]=                 //A

{

0x00, 0x00, 0x00, 0x10, 010, 0x18, 0x28, 0x28, 0x24, 0x3C, 0x44, 0x42, 0x42, 0xE7, 0x00, 0x00

};

unsigned char ASCII_R[]=                 //R

{

0x00, 0x00, 0x00, 0xFC, 0x42, 0x42, 0x42, 0x7C, 0x48, 0x48, 0x44, 0x44, 0x42, 0xE3, 0x00,
```

```
0x00

};

unsigned char ASCII_M[]=                    //M

{

0x00, 0x00, 0x00, 0xEE, 0x6C, 0x6C, 0x6C, 0x6C, 0x54, 0x54, 0x54, 0x54, 0x54, 0xD6, 0x00,
0x00

};



//绘制背景

void Brush_Background( U32 c)

{

        int x,y ;



    for( y = 0 ; y < LCD_HEIGHT ; y++ )

    {

            for( x = 0 ; x < LCD_WIDTH ; x++ )

            {

                    LCD_BUFFER[y][x] = c ;

            }

    }

}



//绘制像素点
```

```
void PutPixel(U32 x,U32 y, U32 c )
{
LCD_BUFFER[y][x] = c;
}

//绘制大小为 16×16 的中文字符
void Draw_Text16(U32 x,U32 y,U32 color,const unsigned char ch[])
{
    unsigned short int i,j;
    unsigned char mask,buffer;

    for(i=0;i<16;i++)
    {
        mask=0x80;                          //掩码
        buffer=ch[i*2];                     //提取一行的第一个字节
        for(j=0;j<8;j++)
        {
            if(buffer&mask)
            {
                PutPixel(x+j,y+i,color);            //为笔画上色
            }
            mask=mask>>1;
```

```
                }
                mask=0x80;                          //掩码
                buffer=ch[i*2+1];               //提取一行的第二个字节
                for(j=0;j<8;j++)
                {
                        if(buffer&mask)
                        {
                                PutPixel(x+j+8,y+i,color);                  //为笔画上色
                        }
                        mask=mask>>1;
                }
        }
}

//绘制大小为 8×16 的 ASCII 码
void Draw_ASCII(U32 x,U32 y,U32 color,const unsigned char ch[])
{
        unsigned short int i,j;
        unsigned char mask,buffer;

        for(i=0;i<16;i++)
        {
```

```
                mask=0x80;
                buffer=ch[i];
                for(j=0;j<8;j++)
                {
                        if(buffer&mask)
                        {
                                PutPixel(x+j,y+i,color);
                        }
                        mask=mask>>1;
                }
        }
}

//LCD 初始化
void LCD_Init()
{
        rGPCUP    = 0x00000000;
        rGPCCON = 0xaaaa02a9;

        rGPDUP    = 0x00000000;
        rGPDCON=0xaaaaaaaa; //Initialize VD[15:8]
rLCDCON1=(CLKVAL_TFT<<8)|(MVAL_USED<<7)|(PNRMODE_TFT<<5)|(BPPMODE_TF
```

```
T<<1)|0;

        rLCDCON2=(VBPD<<24)|(LINEVAL<<14)|(VFPD<<6)|(VSPW);

            rLCDCON3=(HBPD<<19)|(HOZVAL<<8)|(HFPD);

            rLCDCON4=(HSPW);

            rLCDCON5 = (BPP24BL<<12) | (INVVCLK<<10) | (INVVLINE<<9) |
(INVVFRAME<<8) | (0<<7) | (INVVDEN<<6) | (PWREN<<3)   |(BSWP<<1) | (HWSWP);



            rLCDSADDR1=(((U32)LCD_BUFFER>>22)<<21)|M5D((U32)LCD_BUFFER>>1);


rLCDSADDR2=M5D( ((U32)LCD_BUFFER+(LCD_WIDTH*LCD_HEIGHT*4))>>1 );

            rLCDSADDR3=LCD_WIDTH*4/2;

            rLCDINTMSK|=(3); // MASK LCD Sub Interrupt

            rTCONSEL = 0;          // Disable LPC3480



        rGPGUP=rGPGUP&(~(1<<4))|(1<<4); // Pull-up disable

            rGPGCON=rGPGCON&(~(3<<8))|(3<<8); //GPG4=LCD_PWREN

            rGPGDAT = rGPGDAT | (1<<4) ;

            rLCDCON5=rLCDCON5&(~(1<<3))|(1<<3);      // PWREN

            rLCDCON5=rLCDCON5&(~(1<<5))|(0<<5);      // INVPWREN

}



void Main(void)

{
```

```
LCD_Init();

rLCDCON1|=1;        //开启 LCD 显示

Brush_Background(0xFFFFFF);                //绘制白色背景

        //绘制黑色字符
Draw_Text16(50,100,0x0,zhao);
Draw_Text16(66,100,0x0,chun);
Draw_Text16(82,100,0x0,jiang);
Draw_ASCII(50,120,0x0,ASCII_A);
Draw_ASCII(58,120,0x0,ASCII_R);
Draw_ASCII(66,120,0x0,ASCII_M);
while(1)
{
;
}
}
```

看了上面的程序，有人可能会问，如果要在程序中显示大量的中文字符，是不是要把这些字符都取模啊？回答是肯定的，但前人已经为我们完成了这一步，做成了数据库，并

且进行了编码，只要按照编码规则调用该库文件，就可以检索到相要的字符。下面就来说说编码规则：每个汉字是由两个字节表示的，前一个字节表示的区号，后一个字节表示的位号，那么汉字在汉字库中的位置为：94×（区号－1）＋（位号－1）。94 表示的是每个区里一共有 94 个汉字，减 1 表示的是数组是从 0 开始，而区号和位号是从 1 开始的。具体到汉字在某一数据库中的位置，还需要乘以一个汉字字模所占的字节数，即[94×（区号－1）＋（位号－1）]×一个汉字字模所占字节数。如一个字模大小为 16×16 的宋体数据库，库里每个汉字所占的字节为 16×16÷8＝32，则每个汉字在该宋体数据库中的位置为：[94×（区号－1）＋（位号－1）]×32。ASCII 码的字符调用比汉字字符要简单，只要把它乘以字模所占字节数即可找到该字符所在字库的位置，如 8×16 的 ASCII 字库，ASCII 码在该字库的位置为 ASCII×16。如果中文字符和 ASCII 码混合在一样，如何区分它们呢？其实也很简单，ASCII 码的最高位是 0，而中文的最高位是 1，因此当读取到的一个字节的最高位是 0，则该字节为 ASCII 码，它的下一个字节与这个字节无关；当取得到的字节的最高位是 1，则表示的是中文字符，并且该字节与它的下一个字节组合在一起表示完整的一个汉字。

编码规则介绍完了，那么如何打开字库呢？我们可以利用前人已做好的字库，然后像访问一般文件一样打开它。另一种方法是把字库变换成一个超大的数组，那么我们就可以像操作数组一样读取字库了（在这里，我们使用的是这种方法）。

下面我们就给出具体的实例，它可以显示任意的中、英文字符串。这里只给出主程序，需要调用的子程序与上面的一样。

```
#include "font_libs.h"                    //内有两个数组__HZK[ ]和__ASCII[ ]

//分别表示中文字符和 ASCII 字符

……

……

void Main(void)

{

        unsigned char String[]="我的博客是：http://blog.csdn.net/zhaocj";

        int length = sizeof(String);

        int k,xx;
```

```
        unsigned char qh,wh;
        const unsigned char *mould;
LCD_Init();
rLCDCON1|=1;

Brush_Background(0xffffff);
for(k=0,xx=0;k<length-1;k++)
{
            if(String[k]&0x80)          //中文字符
        {
                qh=String[k]-0xa0;                  //区号
                wh=String[k+1]-0xa0;            //位号
                mould = & __HZK[ ( ( qh - 1 )*94 + wh- 1 )*32 ];
                Draw_Text16(4+xx,100,0x0f,mould);
                xx+=16;
                k++;
        }
            else                                //ASCII 码字符
            {
                mould = & __ASCII[String[k]*16];
                Draw_ASCII(4+xx,100,0x0,mould);
                xx+=8;
```

```
                }
}
while(1)
{
                ;
}
}
```