人机交互是嵌入式系统必须具有的功能。比较简单的人机交互有按键、LED、蜂鸣器，稍微复杂的有 7 段数码管和点阵。但如今这些都不能满足人们的需求了，所以又出现了 LCD 和触摸屏技术。s3c2440 具有 LCD 和触摸屏接口，可以很好的连接 LCD 和触摸屏。这篇文章主要介绍 TFT 型 LCD 的用法。

要想正确使用 LCD，必须注意两点：1、时序；2、显示缓存区。

1、时序

LCD 一般需要三个时序信号：VSYNC、HSYNC 和 VCLK。VSYNC 是垂直同步信号，在每进行一个帧（即一个屏）的扫描之前，该信号就有效一次，由该信号可以确定 LCD 的场频，即每秒屏幕刷新的次数（单位 Hz）。HSYNC 是水平同步信号，在每进行一行的扫描之前，该信号就有效一次，由该信号可以确定 LCD 的行频，即每秒屏幕从左到右扫描一行的次数（单位 Hz）。VCLK 是像素时钟信号。

s3c2440 处理 LCD 的时钟源是 HCLK，通过寄存器 LCDCON1 中的 CLKVAL 可以调整 VCLK 频率大小，它的公式为：

VCLK＝HCLK÷[(CLKVAL＋1)×2]

例如，HCLK 的频率为 100MHz，要想驱动像素时钟信号为 6.4MHz 的 LCD 屏，则通过上式计算 CLKVAL 值，结果 CLKVAL 为 6.8，取整后（值为 6）放入寄存器 LCDCON1 中相应的位置即可。由于 CLKVAL 进行了取整，因此我们把取整后的值代入上式，重新计算 VCLK，得到 VCLK＝7.1MHz。

按理说，对于一个已知尺寸（即水平显示尺寸 HOZVAL 和垂直显示尺寸 LINEVAL 已知）的 LCD 屏，只要确定了 VCLK 值，行频和场频就应该知道了。但这样还不行的，因为在每一帧时钟信号中，还会有一些与屏显示无关的时钟出现，这就给确定行频和场频带来了一定的复杂性。如在 HSYNC 信号先后会有水平同步信号前肩（HFPD）和水平同步信号后肩（HBPD）出现，在 VSYNC 信号先后会有垂直同步信号前肩（VFPD）和垂直同步信号后肩（VBPD）出现，在这些信号时序内，不会有有效像素信号出现，另外 HSYNC 和 VSYNC 信号有效时，其电平要保持一定的时间，它们分别叫做水平同步信号脉宽 HSPW 和垂直同步信号脉宽 VSPW，这段时间也不能有像素信号。因此计算行频和场频时，一定要包括这些信号。HBPD、HFPD 和 HSPW 的单位是一个 VCLK 的时间，而 VSPW、VFPD 和 VBPD 的单位是扫描一行所用的时间。在 s3c2440 中，所有的这些信号（VSPW、VFPD、VBPD、LINEVAL、HBPD、HFPD、HSPW 和 HOZVAL）都是实际值减 1 的结果。这些值是通过寄存器 LCDCON2、LCDCON3 和 LCDCON4 来配置，只要把这些值配置成与所要驱动的 LCD 中相关内容的数据一致即可。例如，我们所要显示的 LCD 屏大小为 320×240，因此 HOZVAL＝320－1，LINEVAL＝240－1。水平同步信号的脉宽、前肩和后肩分别为 30、20 和 38，则 HSPW＝30－1，HFPD＝20－1，HBPD＝38－1；垂直同步信号的脉宽、前肩和后肩分别为

3、12 和 15，则 VSPW＝3－1，VFPD＝12－1，VBPD＝15－1。

下面我们就具体计算一下行频（HSF）和场频（VSF）：

HSF＝VCLK÷[(HSPW＋1)＋(HSPD＋1)＋(HFPD＋1)＋(HOZVAL＋1)]

＝7.1÷408＝17.5kHz

VSF＝HSF÷[(VSPW＋1)＋(VBPD＋1)＋(VFPD＋1)＋(LINEVAL＋1)]

＝17.5÷270＝64.8Hz

在有些情况下，s3c2440 的 LCD 时钟信号的默认极性与所控制的 LCD 时钟信号的极性相反，这时可以通过寄存器 LCDCON5 的相关位来改变某些时钟信号的极性。

2、显示缓存区

只要把所要显示的数据放入显示缓存区内，就可以在屏幕上呈现内容。该缓存区是我们自己编程时开辟的一段内存区。一般我们是通过定义一个与屏幕尺寸大小相同的二维数组来开辟该空间的，这样控制屏幕内容会方便一些，如当屏幕的尺寸为 320×240 时，可以定义该缓存区为 LCD_BUFFER[240][320]。由于 s3c2440 支持 16 位和 24 位的非调色板真彩色的 TFT 型 LCD 模式，而 24 位颜色模式是用 32 位数据来表示的，所以前面定义的那个二维数据的数据类型应该是半字整型或全字整型的。例如，在 24 位颜色模式下，我们想要在尺寸大小为 320×240 屏幕的中心处设置为白色像素，则：LCD_BUFFER[120][160]＝0xffffffff。

在 s3c2440 中，寄存器 LCDSADDR1 和 LCDSADDR2 用于设置显示缓存区，即把我们定义的那个二维数组告诉 s3c2440。其中 LCDBANK 的 9 位数据指定 LCD 的 BANK，即显示缓存区的第 30 位到第 22 位地址；LCDBASEU 的 21 位数据指定了 LCD 的基址，即显示缓存区开始地址的第 21 位到第 1 位；LCDBASEL 的 21 位数据指定了 LCD 的尾址，即显示缓存区结束地址的第 21 位到第 1 位。例如，我们想要在尺寸为 320×240 的屏幕上显示 24 位颜色，定义的显示缓存区数组为 LCD_BUFFER[240][320]，则 LCDBANK 等于 LCD_BUFFER 的第 30 位到第 22 位数据值（因为 LCD_BUFFER 表示的就是数组的首地址），LCDBASEU 等于 LCD_BUFFER 的第 21 位到第 1 位数据值，由于是用 32 位数据表示 24 为颜色，因此每个像素值是 4 个字节，所以 LCDBASEL 等于(LCD_BUFFER＋(240×320×4))结果的第 21 位到第 1 位的数据值。另外寄存器 LCDSADDR3 有两个内容：OFFSIZE 和 PAGEWIDTH。OFFSIZE 用于虚拟屏幕的偏移长度，如果我们不使用虚拟屏幕，就把它置为 0；PAGEWIDTH 定义了视口的宽，单位是半字，如在上面的例子中，PAGEWIDTH 应该为 320×32÷16。

下面我们给出一段具体的 TFT 型 LCD 显示的实例，其中，屏幕的大小为 320×240，所设置

的颜色为 24 位真彩色模式。

```
#define U32 unsigned int
#define M5D(n)                  ((n) & 0x1fffff)        //用于设置显示缓存区时，取低 21 位地址

#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 G

#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                     //屏幕的宽

#define LCD_HEIGHT    240                       //屏幕的高


//垂直同步信号的脉宽、后肩和前肩

#define VSPW            (3-1)

#define VBPD                    (15-1)

#define VFPD            (12-1)

//水平同步信号的脉宽、后肩和前肩

#define HSPW            (30-1)

#define HBPD            (38-1)

#define HFPD (20-1)

//显示尺寸

#define LINEVAL    (LCD_HEIGHT-1)

#define HOZVAL     (LCD_WIDTH-1)
```

```
//for LCDCON1

#define CLKVAL_TFT          6           //设置时钟信号

#define MVAL_USED           0           //

#define PNRMODE_TFT         3           //TFT 型 LCD

#define BPPMODE_TFT         13          //24 位 TFT 型 LCD


//for LCDCON5

#define BPP24BL             0       //32 位数据表示 24 位颜色值时，低位数据有效，高 8 位
无效

#define INVVCLK             0       //像素值在 VCLK 下降沿有效

#define INVVLINE            1       //翻转 HSYNC 信号

#define INVVFRAME   1       //翻转 VSYNC 信号

#define INVVD               0       //正常 VD 信号极性

#define INVVDEN             0       //正常 VDEN 信号极性

#define PWREN               1       //使能 PWREN 信号

#define BSWP                        0       //颜色数据字节不交换

#define HWSWP               0       //颜色数据半字不交换

//定义显示缓存区

volatile U32 LCD_BUFFER[LCD_HEIGHT][LCD_WIDTH];


//延时程序

void delay(int a)
```

```
{
        int k;
        for(k=0;k<a;k++)
                ;
}

//绘制屏幕背景颜色，颜色为 c
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 ;
            }
    }
}

//画实心圆，颜色为 c。圆心在屏幕中心，半径为 80 个像素
void Draw_Circular(U32 c)
```

```
{
        int x,y ;
        int tempX,tempY;
        int radius = 80;
        int SquareOfR = radius*radius;

    for( y = 0 ; y < LCD_HEIGHT ; y++ )
    {
            for( x = 0 ; x < LCD_WIDTH ; x++ )
            {
                    if(y<=120 && x<=160)
                    {
                            tempY=120-y;
                            tempX=160-x;
                    }
                        else if(y<=120&& x>=160)
                        {
                                tempY=120-y;
                            tempX=x-160;
                        }
                        else if(y>=120&& x<=160)
                        {
```

```
                tempY=y-120;
            tempX=160-x;
        }
        else
        {
                tempY = y-120;
                tempX = x-160;
        }

        if ((tempY*tempY+tempX*tempX)<=SquareOfR)
                LCD_BUFFER[y][x] = c ;
    }
  }
}

void Main(void)
{
    //配置 LCD 相关引脚
rGPCUP   = 0x00000000;
    rGPCCON = 0xaaaa02a9;
    rGPDUP   = 0x00000000;
    rGPDCON=0xaaaaaaaa;
```

```
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*32/16;

        rLCDINTMSK|=(3);          //  屏蔽 LCD 中断

        rTCONSEL = 0;                   //无效 LPC3480


rGPGUP=rGPGUP&(~(1<<4))|(1<<4);          //GPG4 上拉电阻无效

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

        rGPGDAT = rGPGDAT | (1<<4) ;                    //GPG4 置 1


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

        rLCDCON5=rLCDCON5&(~(1<<5))|(0<<5);     //PWREN 信号极性不翻转
```

```
rLCDCON1|=1;                    //LCD 开启

while(1)
    {
        //黑色背景，白色实心圆
    Brush_Background(0x0);
        Draw_Circular(0xffffff);
        delay(5000000);

    //白色背景，黑色实心圆
    Brush_Background(0xffffff);
        Draw_Circular(0x0);
    delay(5000000);

        //蓝色背景，黄色实心圆
        Brush_Background(0xff);
        Draw_Circular(0xffff00);
    delay(5000000);

        //绿色背景，品色实心圆
    Brush_Background(0xff00);
    Draw_Circular(0xff00ff);
```

```
        delay(5000000);

                //红色背景，青色实心圆
        Brush_Background(0xff0000);
        Draw_Circular(0xffff);
        delay(5000000);

                //青色背景，红色实心圆
        Brush_Background(0xffff);
        Draw_Circular(0xff0000);
        delay(5000000);

                //品色背景，绿色实心圆
        Brush_Background(0xff00ff);
        Draw_Circular(0xff00);
        delay(5000000);

                //黄色背景，蓝色实心圆
        Brush_Background(0xffff00);
        Draw_Circular(0xff);
        delay(5000000);
}
```

}