S3C2410 下 LCD 驱动程序移植及 GUI 程序编写

Write by llg 著作权所有:刘利国
如转载请告知作者 laoliu@laoliu-soft.net 并注明出处 www.laoliu-soft.net

1. 为了不让大家觉枯燥，让朋友们更好的理解，我以一个实例来叙述 S3C2410 下一个驱动程序的编写（本文的初始化源码以华恒公司提供的 s3c2410fb.c 为基础）及简单的 GUI 程序的编写。
2. 拿到一块 LCD，首先要将 LCD 的各个控制线与 S3C2410 的 LCD 控制信号相接，当然，电源也一定要接入了，否则不亮可别找我。另外需要注意以下几点：
   1） 背光：对于大部分的彩色 LCD 一定要接背光，我们才能看到屏上的内容；
   2） 控制信号：不同的 LCD 厂商对于控制信号有不同的叫法，S3C2410 芯片手册也给出了一个信号的多个名称（图一），这就要看你们硬件工程师的功底了，

EXTERNAL INTERFACE SIGNAL

VFRAME/VSYNC/STV	: Frame synchronous signal (STN)/vertical synchronous signal (TFT)/SEC TFT signal
VLINE/HSYNC/CPV	: Line synchronous pulse signal (STN)/horizontal sync signal (TFT)/SEC TFT signal
VCLK/LCD_HCLK	: Pixel clock signal (STN/TFT)/SEC TFT signal
VD[23:0]	: LCD pixel data output ports (STN/TFT/SEC TFT)
VM/VDEN/TP	: AC bias signal for the LCD driver (STN)/data enable signal (TFT)/SEC TFT signal
LEND/STH	: Line end signal (TFT)/SEC TFT signal
LCD_PWREN	: LCD panel power enable control signal
LCDVF0	: SEC TFT Signal OE
LCDVF1	: SEC TFT Signal REV
LCDVF2	: SEC TFT Signal REVB

图一 S3C2410 手册上给出的控制信号的名称及解释

这里我做一个简单的介绍：

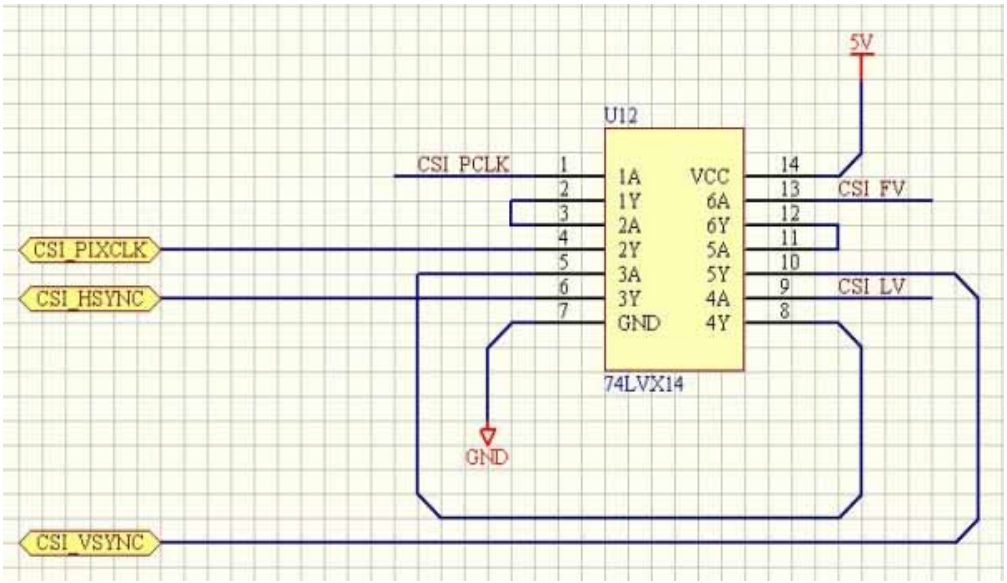
- VFRAME：LCD 控制器和 LCD 驱动器之间的帧同步信号。该信号告诉 LCD 屏的新的一帧开始了。LCD 控制器在一个完整帧显示完成后立即插入一个 VFRAME 信号，开始新一帧的显示；
- VLINE：LCD 控制器和 LCD 驱动器之间的线同步脉冲信号，该信号用于 LCD 驱动器将水平线（行）移位寄存器的内容传送给 LCD 屏显示。LCD 控制器在整个水平线（整行）数据移入 LCD 驱动器后，插入一个 VLINE 信号；
- VCLK：LCD 控制器和 LCD 驱动器之间的像素时钟信号，由 LCD 控制器送出的数据在 VCLK 的上升沿处送出，在 VCLK 的下降沿处被 LCD 驱动器采样；
- VM：LCD 驱动器的 AC 信号。VM 信号被 LCD 驱动器用于改变行和列的电压极性，从而控制像素点的显示或熄灭。VM 信号可以与每个帧同步，也可以与可变数量的 VLINE 信号同步。

3） 数据线：也就是我们说的 RGB 信号线，S3C2410 芯片手册上都有详细的说明，由于篇幅关系，在此不一一摘录，不过需要与硬件工程是配合的是他采用了哪种接线方法，24 位 16 位或其它。对于 16 位 TFT 屏又有两种方式，在写驱动前你要清楚

是 5：6：5 还是 5：5：5：I，这些与驱动的编写都有关系

4） 要注意一下 LCD 的电源电压，对于手持设备来说一般都为 5V 或 3.3V，或同时支持 5V 和 3.3V，如果 LCD 的需要的电源电压是 5V，那就要注意了，S3C2410 的逻辑输出电压只有 3.3V，此时一定要让你们的硬件工程师帮忙把 S3C2410 的逻辑输出电压提高到 5V，否则你可能能将屏点亮，但显示的图像要等到太阳从西边出来的那一天才能正常，呵呵，我可吃过苦头的哦！

5） 3.3V 逻辑电压转变成 5V 逻辑电压电路图（此图由华恒公司提供）



6） 最后还有一个问题，有些 LCD 屏还需要一颗伴侣芯片，就是 S3C2410 手册中的那颗 LPC3600。这可能在 LCD 的手册中都有论述吧，我没有遇到过这样的屏，所以也不是很清楚。那么是不是所有的屏与 S3C2410 相接都需要那个讨厌的家伙呢？这是好多人（包括我）在最开始都会有的疑问，不过现在的大部分 LCD 屏应该都不需要这个讨厌的家伙了，屏的控制信号直接与 S3C2410 的控制信号相接就可以了，至少我还没有遇到过。

7） 还得提醒大家一下，S3C2410 到 LCD 屏的连线千万千万别超过 **0.5** 米，否则会给你带来麻烦，我也是吃过苦头的，LCD 屏上面的部分显示任何信息都是正确的，而只有屏的底部会有时正确有时错误，折腾了好一阵，才知道是连线太长的缘故！

3. 好了，在硬件工程师的帮助下，硬件接好了，那就该我们做软件的干活了，编写驱动吧

1） 让我们首先看一下 RGB 数据结构的定义

在 s3c2410fb.c 中找到如下信息

```
static struct s3c2410fb_rgb xxx_tft_rgb_16 = {
    red:    {offset: 11,    length: 5, },
    green:  {offset: 5, length: 6, },
    blue:   {offset: 0, length: 5, },
    transp: {offset: 0, length: 0, },
};
```

这是对 16 位色的 RGB 颜色进行定义，R：G：B：I = 5：6：5：0，即我们常说的 565 显示方式。呵呵，为了让有些朋友更好的理解，我多罗嗦几句，我们随便写一个 16 位数据的颜色数据（为了分析的方便，我把它写成二进制）

RGB = 10101101 10111001

根据上面的结构定义我们来分析一下 RGB 各是多少（因为没有透明色，我们不去分析）

a) blue: {offset: 0, length: 5} 偏移量为 0，长度为 5，我们从那个 RGB 中提取出来便是“ 11001 ”

b) green:{offset: 5, length: 6} 偏移量为 5，长度为 6，我们从那个 RGB 中提取出来便是 101 101

c) red: {offset: 11, length: 5 } 偏移量为 11，长度为 5，我们从那个 RGB 中提取出来便是 10101

d) 我们得到了一个 RGB 值为 13：45：200，就是这个颜色 

e) 那么反过来，有了 RGB 的值我们该如何，因为 RGB 的有效位数都不足一个字节（8 位），那我们只能忍痛割爱了，舍弃掉低位数据，代码如下

```
r = (rDat & 0xF8);
g = (gDat & 0xFC);
b = (bDat & 0xF8);
hight = r | (g >> 5);
low = (g<<3) | (b >> 3);
color = (hight<<8) | low;
```

记住，这段代码在 GUI 程序中是有用的

2） 对于 8 位色（256 色）的数据结构定义

```
static struct s3c2410fb_rgb rgb_8 = {
    red:    {offset: 0, length: 4, },
    green:  {offset: 0, length: 4, },
    blue:   {offset: 0, length: 4, },
    transp: {offset: 0, length: 0, },
};
```

这是原程序中给出的定义，我感觉有些错误，我认为应该为 R：G：B = 3：3：2

```
//llg add
static struct s3c2410fb_rgb xxx_stn_rgb_8 = {
    red:    {offset: 5, length: 3, },
    green:  {offset: 2, length: 3, },
    blue:   {offset: 0, length: 2, },
    transp: {offset: 0, length: 0, },
};
```

因为没有亲自去调适，所以没有什么发言权，希望做过这方面的朋友给我一个答案。

3） 对于 CSTN 屏，一般都能达到 12 位色（4096 色）的，S3C2410 这颗芯片也是支持的，但是在软件方面要做的工作比较大，因为从原有的代码，我们找不到任何 12 位色显示的迹象，另外 Linux 本身好像也不支持 12 位色的，如果你要作的事情比较简单，那你就自己写代码吧。我在此给出 12 位色的数据结构定义

```
//llg add
static struct s3c2410fb_rgb xxx_stn_rgb_12 = {
    red:    {offset: 8, length: 4, },
    green:  {offset: 4,  length: 4, },
    blue:   {offset: 0,  length: 4, },
    transp: {offset: 0,  length: 0, },
};
```

但是要完成 12 位色 CSTN 屏驱动程序的编写还有一些工作要做，稍后我会适当的向大家介绍。

4） 接着看下面的代码，其中要修改的部分已经用绿色标出，下面分别进行介绍。

```
static struct s3c2410fb_mach_info xxx_stn_info __initdata = {
        pixclock:    174757,      bpp:          16,
#ifdef CONFIG_FB_S3C2410_EMUL
    xres:         96,
#else
    xres:         240,
#endif
    yres:         320,


    hsync_len    :  5,     vsync_len     :  1,
    left_margin  :  7,     upper_margin  :  1,
    right_margin:   3,     lower_margin  :  3,

    sync:         0,       cmap_static:     1,
    reg : {
        lcdcon1 : LCD1_BPP_16T | LCD1_PNR_TFT | LCD1_CLKVAL(7) ,
        lcdcon2 : LCD2_VBPD(4) | LCD2_VFPD(1) | LCD2_VSPW(1),
        lcdcon3 : LCD3_HBPD(6) | LCD3_HFPD(30),
        lcdcon4 : LCD4_HSPW(3) | LCD4_MVAL(13),
        lcdcon5 : LCD5_FRM565 | LCD5_HWSWP | LCD5_PWREN,
    },
};
```

a) 颜色位数

bpp：16

如果你的 LCD 屏是 TFT 的，那一般都可以达到 16 位色或 24 位色，这也要看硬件怎么连接了，根据情况进行设置即可；

如果你的 LCD 屏是 CSTN 的，按照常规 LCD 手册的介绍，一般都可以支持到 8 位色（256 色），而实际的 CSTN 屏的显示效果都可以达到 12 位色（4096 色），那可有很大的区别的，如果你要选择便宜的屏又要丰富的颜色，那就费点劲，完成 12 位色的驱动。

b) LCD 屏的宽度和高度

xres: 240

yres: 320

这个就不用多说了，你的屏的分辨率是多少就设置成多少呗。

c) 寄存器的设置，这些也不困难。下面就让我们一起一口一口的将 S3C2410 的 LCD 寄存器统统吃掉！

- 首先介绍一下我这块屏，这是日立的一块 TFT 屏，大小为 640X240，可以支持到 16 位色。
- 与驱动有关的一张表

ITEM		MIN.	TYP.	MAX.	UNIT	SYMBOL	REMARKS
DCLK	Cycle time	37.4	(47.8)	58.1	ns	t_{CLK}	
	Low level Width	15	-	-		t_{WCL}	
	High level Width	15	-	-		t_{WCH}	
	Rise time	-	-	25		t_{rCLK}	
	Fall time	-	-	25		t_{fCLK}	
	Duty	0.45	0.5	0.55	-	D	D= t_{CLKL}/ t_{CLK}
Hsync	Set up time	5	-	-	ns	t_{SH}	for DCLK
	Hold time	10	-	-		t_{HH}	
	Cycle	679	(709)	739	t_{CLK}	t_{HP}	
	Valid width	4	5	5		t_{WH}	
	Rise/Fall time	-	-	30	ns	T_{Hr},t_{Hf}	
Vsync	Set up	0	-	-	t_{CLK}	t_{SV}	for Hsync
	Hold	2	-	-		t_{HV}	
	Cycle	485	(491)	533	t_{HP}	t_{VP}	
	Valid width	2	2	2		t_{WV}	
	Rise/Fall time	-	-	50	ns	t_{Vr},t_{Vf}	
DTMG	Set up time	5	-	-	ns	t_{SI}	for DCLK
	Hold time	10	-	-		t_{HI}	
	Rise/Fall time	-	-	30	ns	T_{Ir},t_{If}	
	Horizontal back porch	24	(37)	50	t_{CLK}	t_{HBP}	
	Horizontal front porch	15	(32)	49		t_{HFP}	
	Vertical back porch	4	(7)	28	t_{HP}	t_{VBP}	
	Vertical front porch	1	(4)	25		t_{VFP}	
Data	Set up time	5	-	-	ns	t_{SD}	for DCLK
	Hold time	10	-	-		t_{HD}	
	Rise/Fall time	-	-	25	ns	T_{Dr},t_{Df}	

图二 LCD 屏资料

有了这些信息，让我们看一下 LCD 寄存器的设置。

- LCD 控制器 1

LCD Control 1 Register

Register	Address	R/W	Description	Reset Value
LCDCON1	0X4D000000	R/W	LCD control 1 register	0x00000000

LCDCON1	Bit	Description	Initial State
LINECNT (read only)	[27:18]	Provide the status of the line counter. Down count from LINEVAL to 0	0000000000
CLKVAL	[17:8]	Determine the rates of VCLK and CLKVAL[9:0]. **STN**: VCLK = HCLK / (CLKVAL × 2) (CLKVAL ≥ 2) **TFT**: VCLK = HCLK / [(CLKVAL+1) × 2] (CLKVAL ≥ 0)	0000000000
MMODE	[7]	Determine the toggle rate of the VM. 0 = Each Frame, 1 = The rate defined by the MVAL	0
PNRMODE	[6:5]	Select the display mode. 00 = 4-bit dual scan display mode (STN) 01 = 4-bit single scan display mode (STN) 10 = 8-bit single scan display mode (STN) 11 = TFT LCD panel	00
BPPMODE	[4:1]	Select the BPP (Bits Per Pixel) mode. 0000 = 1 bpp for STN, Monochrome mode 0001 = 2 bpp for STN, 4-level gray mode 0010 = 4 bpp for STN, 16-level gray mode 0011 = 8 bpp for STN, color mode 0100 = 12 bpp for STN, color mode 1000 = 1 bpp for TFT 1001 = 2 bpp for TFT 1010 = 4 bpp for TFT 1011 = 8 bpp for TFT 1100 = 16 bpp for TFT 1101 = 24 bpp for TFT	0000
ENVID	[0]	LCD video output and the logic enable/disable. 0 = Disable the video output and the LCD control signal. 1 = Enable the video output and the LCD control signal.	0

- ✧ LINECNT ---- 这是一个只读的数据，我们当然没有必要理它
- ✧ CLKVAL ---- 这可是一个很有用的参数，其实没必要管它后面的计算，我们可以通过实际的测试来得出一个有效的值，对于

320x240 的屏一般设置为 7 就可以了，而对于 640x480 的屏，该值可以小一点。对于后面的计算公式及注释(STN: CLKVAL >= 2，TFT: CLKVAL >= 0),我不知道该如何去理解，因为在实际的应用中我点了一块 640X240 的 CSTN 屏，当我的 CLKVAL = 1 时才达到了一个最佳的效果，这似乎与说明书相违背，我也解释不清为什么？！

- ✧ PNRMODE ---- 这个应该不用多做解释，大家一看都明白了，对于 TFT 屏，只能设置成 11,而对于 CSTN 屏，可能需要根据实际屏的信息去设置,我遇到的屏都设置成 10,即 8bit 单扫描模式。对于 4bit 单扫描、4bit 双扫描、8bit 单扫描的说明在 s3c2410 的手册中有详细的介绍，大家可以去参考一下。
- ✧ BPPMODE ---- 这个参数更不用多说了吧，就是设置屏的颜色位数喽。
- ✧ 这些参数的设置都很简单，我给出我这块屏的定义：

```
lcdcon1 : LCD1_BPP_16T | LCD1_PNR_TFT | LCD1_CLKVAL(1) ,
```

- ✧ 同时，我也给出一块 CSTN 屏的寄存器参数信息

```
lcdcon1 : LCD1_BPP_12S |  LCD1_PNR_8S | LCD1_CLKVAL(9),
```

➢ LCD 控制器 2

LCD Control 2 Register

Register	Address	R/W	Description	Reset Value
LCDCON2	0X4D000004	R/W	LCD control 2 register	0x00000000

LCDCON2	Bit	Description	Initial State
VBPD	[31:24]	**TFT**: Vertical back porch is the number of inactive lines at the start of a frame, after vertical synchronization period. **STN**: These bits should be set to zero on STN LCD.	0x00
LINEVAL	[23:14]	**TFT/STN**: These bits determine the vertical size of LCD panel.	0000000000
VFPD	[13:6]	**TFT**: Vertical front porch is the number of inactive lines at the end of a frame, before vertical synchronization period. **STN**: These bits should be set to zero on STN LCD.	00000000
VSPW	[5:0]	**TFT**: Vertical sync pulse width determines the VSYNC pulse's high level width by counting the number of inactive lines. **STN**: These bits should be set to zero on STN LCD.	000000

对于 TFT 屏必须要填，至于什么意思怎么翻译，相信大家都比我的水平强，自己翻译吧。我只说明从 LCD 中如何将这个值“扣”出来。

很容易，看一下**图二 LCD 屏资料，**对比一下得出如下信息：

- ✧ LCD2_VBPD:

  Vertical vack proch 典型值为 7
- ✧ LCD2_VFPD：

  Vertical front porch 典型值为 4
- ✧ LCD2_VSPW：

  Vsync Valid width 典型值为 2
- ✧ 关于 LINEVAL 在程序的后面将会提到，此处不必理会。
- ✧ 经过分析，我们知道了如何设置 LCD2：

```
lcdcon2 : LCD2_VBPD(7) | LCD2_VFPD(4) | LCD2_VSPW(2),
```

- ✧ 对于 STN（CSTN）屏，这个寄存器的设置最简单，将 VBPD、VFPD、VSPW 都设置成 Zero 就可以了。即

```
lcdcon2 : LCD2_VBPD(0) | LCD2_VFPD(0) | LCD2_VSPW(0),
```

- LCD 控制器 3

LCD Control 3 Register

Register	Address	R/W	Description	Reset Value
LCDCON3	0X4D000008	R/W	LCD control 3 register	0x00000000

LCDCON3	Bit	Description	Initial state
HBPD (TFT)	[25:19]	**TFT**: Horizontal back porch is the number of VCLK periods between the falling edge of HSYNC and the start of active data.	0000000
WDLY (STN)		**STN**: WDLY[1:0] bits determine the delay between VLINE and VCLK by counting the number of the HCLK. WDLY[7:2] are reserved. 00 = 16 HCLK, 01 = 32 HCLK, 10 = 48 HCLK, 11 = 64 HCLK	
HOZVAL	[18:8]	**TFT/STN**: These bits determine the horizontal size of LCD panel. HOZVAL has to be determined to meet the condition that total bytes of 1 line are 4n bytes. If the x size of LCD is 120 dot in mono mode, x=120 cannot be supported because 1 line consists of 15 bytes. Instead, x=128 in mono mode can be supported because 1 line is composed of 16 bytes (2n). LCD panel driver will discard the additional 8 dot.	00000000000
HFPD (TFT)	[7:0]	**TFT**: Horizontal front porch is the number of VCLK periods between the end of active data and the rising edge of HSYNC.	0X00
LINEBLANK (STN)		**STN**: These bits indicate the blank time in one horizontal line duration time. These bits adjust the rate of the VLINE finely. The unit of LINEBLANK is HCLK X 8. Ex) If the value of LINEBLANK is 10, the blank time is inserted to VCLK during 80 HCLK.	

对于 TFT 屏，很容易将 HBPD 和 HFPD 找出来，如下

- LCD3_HBPD:

  Horizontal back porch 典型值为 37
- LCD3_HFBD:

  Horizontal back porch 典型值为 32
- 对于 HOZVAL 同样会在后面提到，此处暂时不管
- 经过分析，我们知道了如何设置 LCD3：

```
lcdcon3 : LCD3_HBPD(37) | LCD3_HFPD(32)  ,
```

- 对于（STN）CSTN 屏，我没有很好的理解 WDLY 和 LINEBLANK 的真正涵义，通过改变这两个参数的值，我也没有得到特别明显的差异，我一般设置为：

```
lcdcon3 : LCD3_WDLY_16 | 0x10 ,//(0x10为LINEBLANK)
```

- LCD 控制器 4

LCD Control 4 Register

Register	Address	R/W	Description	Reset Value
LCDCON4	0X4D00000C	R/W	LCD control 4 register	0x00000000

LCDCON4	Bit	Description	Initial state
MVAL	[15:8]	**STN**: These bit define the rate at which the VM signal will toggle if the MMODE bit is set to logic '1'.	0X00
HSPW(TFT)	[7:0]	**TFT**: Horizontal sync pulse width determines the HSYNC pulse's high level width by counting the number of the VCLK.	0X00
WLH(STN)		**STN**: WLH[1:0] bits determine the VLINE pulse's high level width by counting the number of the HCLK. WLH[7:2] are reserved. 00 = 16 HCLK, 01 = 32 HCLK, 10 = 48 HCLK, 11 = 64 HCLK	

- 对于 TFT 屏，需要设置 HSPW 的值，这个在 LCD 手册上也很容易得到

  LCD4_HSPW：

  Hsync Valid width 典型值为 5
- 至于 MVAL，我不知道是什么意思，有什么作用，我从来不动它，

只取它最初的那个值 13

- ✧ 经过分析，我们知道了如何设置 LCD4：

```
lcdcon4 : LCD4_HSPW(5) | LCD4_MVAL(13),
```

- ✧ 对于 STN（CSTN）屏，像 WDLY 一样，我通常不改变，因为改变了没有发现有什么作用，这是我驱动中的代码，好几块屏都一样的：

```
lcdcon4 : LCD4_WLH(0) | LCD4_MVAL(13),
```

- ➢ LCD 控制器 5

LCD Control 5 Register

Register	Address	R/W	Description	Reset Value
LCDCON5	0X4D000010	R/W	LCD control 5 register	0x00000000

LCDCON5	Bit	Description	Initial state
Reserved	[31:17]	This bit is reserved and the value should be '0'.	0
VSTATUS	[16:15]	**TFT**: Vertical Status (read only). 00 = VSYNC 01 = BACK Porch 10 = ACTIVE 11 = FRONT Porch	00
HSTATUS	[14:13]	**TFT**: Horizontal Status (read only). 00 = HSYNC 01 = BACK Porch 10 = ACTIVE 11 = FRONT Porch	00
BPP24BL	[12]	**TFT**: This bit determines the order of 24 bpp video memory. 0 = LSB valid 1 = MSB Valid	0
FRM565	[11]	**TFT**: This bit selects the format of 16 bpp output video data. 0 = 5:5:5:1 Format 1 = 5:6:5 Format	0
INVVCLK	[10]	**STN/TFT**: This bit controls the polarity of the VCLK active edge. 0 = The video data is fetched at VCLK falling edge 1 = The video data is fetched at VCLK rising edge	0
INVVLINE	[9]	**STN/TFT**: This bit indicates the VLINE/HSYNC pulse polarity. 0 = Normal 1 = Inverted	0
INVVFRAME	[8]	**STN/TFT**: This bit indicates the VFRAME/VSYNC pulse polarity. 0 = Normal 1 = Inverted	0
INVVD	[7]	**STN/TFT**: This bit indicates the VD (video data) pulse polarity. 0 = Normal 1 = VD is inverted.	0

LCD Control 5 Register (Continued)

LCDCON5	Bit	Description	Initial state
INVVDEN	[6]	**TFT**: This bit indicates the VDEN signal polarity. 0 = Normal 1 = Inverted	0
INVPWREN	[5]	**STN/TFT**: This bit indicates the PWREN signal polarity. 0 = Normal 1 = Inverted	0
INVLEND	[4]	**TFT**: This bit indicates the LEND signal polarity. 0 = Normal 1 = Inverted	0
PWREN	[3]	**STN/TFT**: LCD_PWREN output signal enable/disable. 0 = Disable PWREN signal 1 = Enable PWREN signal	0
ENLEND	[2]	**TFT**: LEND output signal enable/disable. 0 = Disable LEND signal 1 = Enable LEND signal	0
BSWP	[1]	**STN/TFT**: Byte swap control bit. 0 = Swap Disable 1 = Swap Enable	0
HWSWP	[0]	**STN/TFT**: Half-Word swap control bit. 0 = Swap Disable 1 = Swap Enable	0

这个寄存器的看起来比较复杂，但是无外乎这几类：

- ✧ 只读信息：VSTATUS 和 HSTATUS

  只读的东东，设置它也没用，不必理会。

- ✧ TFT 屏的颜色信息：BPP24BL、FRM565

  TFT 屏的颜色信息，这个我们在 LCD 的硬件连接时已经提到了，根据

具体的接线方式，设置信息。

- ✧ 控制信号的极性
  TFT/STN 屏控制信号的极性：INVVCLK、INVVLINE、INVVFRAME、INVVD、INVPWREN、PWREN
  TFT 屏特有的控制信号的极性：INVVDEN、INVLEND、ENLEND
  这些信息主要是使S3C2410的信号输出极性与LCD屏的输入极性的问题，需要根据具体的硬件进行设置，较为常见的是vline/hsync 、VFRAME/VSYNC脉冲的极性。
- ✧ 颜色信息的字节交换控制位：BSWP、HWSWP
  这两位用来控制字节交换和半字交换，主要用来大小头的问题，如果输出到屏上的汉字左右互换了，或者输出到屏上的图花屏了，可以更改这个选项。具体涵义在 S3C2410 芯片手册上有详细的说明。
- ✧ 我的这块 TFT 的信息设置如下：

```
lcdcon5 : LCD5_FRM565 | LCD5_HWSWP | LCD5_PWREN ,
```

- ✧ 一块 CSTN 屏的信息

```
lcdcon5 : LCD5_BSWP | LCD5_PWREN ,
```

- ➢ FrameBuffer 起始寄存器 1

FRAME BUFFER START ADDRESS 1 REGISTER

Register	Address	R/W	Description	Reset Value
LCDSADDR1	0X4D000014	R/W	**STN/TFT**: Frame buffer start address 1 register	0x00000000

LCDSADDR1	Bit	Description	Initial State
LCDBANK	[29:21]	These bits indicate A[30:22] of the bank location for the video buffer in the system memory. LCDBANK value cannot be changed even when moving the view port. LCD frame buffer should be within aligned 4MB region, which ensures that LCDBANK value will not be changed when moving the view port. So, care should be taken to use the malloc() function.	0x00
LCDBASEU	[20:0]	For dual-scan LCD: These bits indicate A[21:1] of the start address of the upper address counter, which is for the upper frame memory of dual scan LCD or the frame memory of single scan LCD. For single-scan LCD: These bits indicate A[21:1] of the start address of the LCD frame buffer.	0x000000

这个寄存器的设置没有必要去修改（TFT/STN），都使用默认的代码即可：

```
new_regs.lcdsaddr1 =
    LCDADDR_BANK(((unsigned long)VideoPhysicalTemp >> 22))
    | LCDADDR_BASEU(((unsigned long)VideoPhysicalTemp >> 1));
```

- ➢ FrameBuffer 起始寄存器 2 和 FrameBuffer 起始寄存器 3

FRAME Buffer Start Address 2 Register

Register	Address	R/W	Description	Reset Value
LCDSADDR2	0X4D000018	R/W	**STN/TFT**: Frame buffer start address 2 register	0x00000000

LCDSADDR2	Bit	Description	Initial State
LCDBASEL	[20:0]	For dual-scan LCD: These bits indicate A[21:1] of the start address of the lower address counter, which is used for the lower frame memory of dual scan LCD. For single scan LCD: These bits indicate A[21:1] of the end address of the LCD frame buffer. LCDBASEL = ((the fame end address) >>1) + 1 = LCDBASEU + (PAGEWIDTH+OFFSIZE)x(LINEVAL+1)	0x0000

FRAME Buffer Start Address 3 Register

Register	Address	R/W	Description	Reset Value
LCDSADDR3	0X4D00001C	R/W	**STN/TFT**: Virtual screen address set	0x00000000

LCDSADDR3	Bit	Description	Initial State
OFFSIZE	[21:11]	Virtual screen offset size (the number of half words). This value defines the difference between the address of the last half word displayed on the previous LCD line and the address of the first half word to be displayed in the new LCD line.	00000000000
PAGEWIDTH	[10:0]	Virtual screen page width (the number of half words). This value defines the width of the view port in the frame.	000000000

这两个寄存器的设置比较重要，在此我给出 12 位色 CSTN 屏和 16 位色 TFT 的设置代码：

```
#if MODE_CSTN_12BIT
    //llg 12bpp
    new_regs.lcdsaddr2 = LCDADDR_BASEL(
        ((unsigned long)VideoPhysicalTemp + (var->xres * var->yres * 3/2))
        >> 1);
    new_regs.lcdsaddr3 = LCDADDR_OFFSET(0) | (LCDADDR_PAGE(var->xres*3/4) /*>> 1*/);
#endif

#if MODE_TFT_16BIT
    /* 16bpp */
    new_regs.lcdsaddr2 = LCDADDR_BASEL(
        ((unsigned long)VideoPhysicalTemp + (var->xres * 2 * (var->yres/*-1*/)))
        >> 1);
    new_regs.lcdsaddr3 = LCDADDR_OFFSET(0) | (LCDADDR_PAGE(var->xres) /*>> 1*/);
#endif
```

- RGB Loopup Table Register

RED Lookup Table Register

Register	Address	R/W	Description	Reset Value
REDLUT	0X4D000020	R/W	**STN**: Red lookup table register	0x00000000

REDLUT	Bit	Description	Initial State
REDVAL	[31:0]	These bits define which of the 16 shades will be chosen by each of the 8 possible red combinations. 000 = REDVAL[3:0], 001 = REDVAL[7:4] 010 = REDVAL[11:8], 011 = REDVAL[15:12] 100 = REDVAL[19:16], 101 = REDVAL[23:20] 110 = REDVAL[27:24], 111 = REDVAL[31:28]	0x00000000

GREEN Lookup Table Register

Register	Address	R/W	Description	Reset Value
GREENLUT	0X4D000024	R/W	**STN**: Green lookup table register	0x00000000

GREENLUT	Bit	Description	Initial State
GREENVAL	[31:0]	These bits define which of the 16 shades will be chosen by each of the 8 possible green combinations. 000 = GREENVAL[3:0], 001 = GREENVAL[7:4] 010 = GREENVAL[11:8], 011 = GREENVAL[15:12] 100 = GREENVAL[19:16], 101 = GREENVAL[23:20] 110 = GREENVAL[27:24], 111 = GREENVAL[31:28]	0x00000000

BLUE Lookup Table Register

Register	Address	R/W	Description	Reset Value
BLUELUT	0X4D000028	R/W	**STN**: Blue lookup table register	0x0000

BULELUT	Bit	Description	Initial State
BLUEVAL	[15:0]	These bits define which of the 16 shades will be chosen by each of the 4 possible blue combinations. 00 = BLUEVAL[3:0], 01 = BLUEVAL[7:4] 10 = BLUEVAL[11:8], 11 = BLUEVAL[15:12]	0x0000

- 这三个寄存器的在驱动 256 色 CSTN 屏的时候需要使用，我在别的芯片上使用过，因为这颗芯片支持 12 位色，所以没有去调试，我给

出两组可能的值：

- S3C44B0 上的

```
rREDLUT = 0xFCA86420;
rGREENLUT = 0xFCA86420;
rBLUELUT = 0xFFFFFA50;
```

- Jupiter 上的

```
rREDLUT = 0xFEC85310
rGREENLUT = 0xFEC85310
rBLUELUT = 0xFB40
```

5) 好了，各个寄存器的设置完成了，最后在驱动 CSTN 屏的时候需要提醒大家一句，CSTN 的信号引脚中有一个叫 VM/DISP 的信号线,这个信号线的作用就是打开 LCD 的显示开关，让其进行显示，它可以接到任何一个 GPIO 口上。S3C2410 中提供了一个 VM 信号，可以将 LCD 的这个信号与 S3C2410 的 VM 信号相接即可，然后在驱动中一定要加上如下语句（蓝色选中部分）：

```
void s3c2410_lcd_init(void)
{
    GPDCON = 0xaaaaaaaa;

#ifdef CONFIG_S3C2410_SMDK
    GPCCON = 0xaaaaaaaa;
    set_gpio_ctrl(GPIO_G4 | GPIO_PULLUP_EN | GPIO_MODE_LCD_PWRDN);
#endif

#if defined(CONFIG_MIZI) && defined(CONFIG_PM)
    if (mz_pm_ops.blank_helper != NULL)
        (*(mz_pm_ops.blank_helper))(MZ_BLANK_ON);
#endif
    //llg
    GPCUP=GPCUP|(1<<4); // Pull-up disable
    GPCDAT=GPCDAT&(~(1<<4))|(1<<4);
    GPCCON=GPCCON&(~(3<<8))|(1<<8);
}
```

否则你的 LCD 可能没有任何显示哦（对于 TFT 屏不需要这个语句）

6) 关于 12 位色的 CSTN 屏的驱动还需要做一些工作，我在这里简单介绍一下：

a) 首先要完成一个 fbcon-cfb12.c 和 fbcon-cfb12.h 的编写，这两个文件很简单，在 armLinux 中不是提供了 fbcon-cfb16.c 和 fbcon-cfb12.h 吗？简单修改一下就可以了；

b) 将 fbcon-cfb12.c 的编译加入 Config.in 中（不会的话去 google 搜一下，或者看一下我的另一篇文章《JFFS2 在 HHARM2410 上的实现》，里边有一些说明），并定义一个 FBCON_HAS_CFB12 参数（模仿 FBCON_HAS_CFB16 呗）；

c) 另外，需要在 s3c2410fb.c 中的相应部分加上对 12 位色的支持即可。呵，说起来简单，但实际做起来可能会有一些问题，给大家一个窍门：在程序中找到 #ifdef FBCON_HAS_CFB16 之类的代码，简单理解一下加上对 12 位色的支持；

d) 我只给出函数 s3c2410fb_set_var 中的改动，其他的应该都不是很困难，相信朋友们都能搞定。

```
//llg add
#ifdef FBCON_HAS_CFB12
    case 12:
        display->visual = FB_VISUAL_PSEUDOCOLOR; //显示变量
        display->line_length    = var->xres*3/2; //行的宽度
        display->dispsw     = &fbcon_cfb12;
        display->dispsw_data    = fbi->fb.pseudo_palette; //
        rgbidx          = RGB_12;
        break;
#endif
```

e) 不要跟我要源码哦，否则老板会不高兴哦  。

4. 驱动写好了，重新 Make，下载就可以了。如果一切顺利，在 TFT 屏或 256 色的 CSTN 屏上会有一个漂亮的小蜻蜓（应该是蜻蜓吧）出现。注意，并不是蜻蜓出现了就代表你的驱动 OK 了，还要用 GUI 程序做进一步的测试，因为某一个或几个参数虽然不正确，但是仍然能够看到小蜻蜓的，但显示图形的时候就有问题了。另外，在驱动 CSTN 到 12 位色的时候，我们在屏上看不到小蜻蜓（我的 N 块 CSTN 屏上都没见到小蜻蜓），我想，可能是 armLinux 本身不支持 12 位色显示，或者我们某些地方没搞对的原因吧，但这不代表你的驱动有问题，用 GUI 程序写 FrameBuffer，看看能否的到正确的结果。
5. GUI 程序的编写

   FrameBuffer 驱动写好了，那么怎么去使用，怎么在 LCD 上显示图像呢？这就是 GUI 程序的任务了，其实要在 LCD 上显示图像，说白了就是把数据(包含颜色)写到 FrameBuffer 中对应的位置就可以了。如果你使用如 Microwinow、MiniGui、Qt 之类的 GUI，则没有必要关心 FrameBuffer 与 LCD 屏上的点如何进行映射了，但如果你在使用了 CSTN 屏，并且要显示效果好的照片，选择了 CSTN 的 12 位色(4096 色)，那你就要自己写 GUI 程序了，因为好像 armLinux(Linux)本身都不支持 12 位色的，听说 MiniGui 支持 12 位色，但我在工作中的要求只是显示图形而已，没有去深入研究 MiniGui，所以自己写了。

   另外请朋友们见谅的是我不能给出全部的源代码，因为我毕竟受雇于人，有些东东是可以 GPL 的，而有些东东暂时是不可以 GPL 的。

   下面给出我的程序的部分代码，希望对朋友们有所帮助。

   1） 全局变量的定义：

```
int fb;
U16 * fbp;
struct fb_var_screeninfo vinfo;
struct fb_fix_screeninfo finfo;
long int screensize = 0;
```

   定义几个全局变量，用起来方便。

   2） 初始化图形显示引擎，将 fb0 与 GUI 的 buffer 做个映射

```
int initgraph()
{
    FILE *fp;
    fb = open("/dev/fb0",O_RDWR);
    if(fb < 0) {
        printf("open fb0 faild\n");
        return -1;
    }
    if(ioctl(fb, FBIOGET_FSCREENINFO, &finfo) == -1 ||
        ioctl(fb, FBIOGET_VSCREENINFO, &vinfo) == -1) {
            printf("Error reading screen info \n");
    }
    printf("xres * yres * bits_per_pixel %d ,%d ,% d\n",vinfo.xres ,
    vinfo.yres , vinfo.bits_per_pixel);
    screensize = vinfo.xres * vinfo.yres * vinfo.bits_per_pixel / 8;
    printf("screensize = % d\n",screensize);
    fbp = (U16 *) mmap(0,screensize,PROT_READ|PROT_WRITE,MAP_SHARED,fb,0);
    return 0;
}
```

- 用 mmap 函数使用户空间的一段地址关联到设备内存（FrameBuffer）上。无论何时，只要程序在分配的地址范围内进行读取或者写入，实际上就是对设备的访问，使用 mmap 可以既快速又简单地访问显示卡的内存。对于象这样的性能要求比较严格的应用来说，直接访问能给我们提供很大不同。
- 不过我曾将帮一个网友调试了一个 S3C44B0 上的 GUI 程序，在他的 GUI 中 mmap 函数总会出错，因为没有拿到他的硬件和驱动源码，没有分析出其中的原因，所以只得用 write 函数，直接向 fb0 写入数据，奇怪的是只写入一部分数据好像都不起任何作用，只得整屏数据写入才搞定了。这可就比较痛苦了，不过好在他只是写入的黑白数据，数据量还不是很大，要是彩色的那可真的痛苦了  。
- 另外，我还想多啰嗦两句，FrameBuffer 的像素点与 LCD 屏上的像素点的对应关系 ，深入了解一下对程序的理解可能会更清楚一点。我们知道黑白（2 色）颜色用 0 和 1 就可以表示了，也就是 1 位数据就可以了，那 1 个字节就可以表示 8 位数据，假如这个字节是 10101010，FrameBuffer 的偏移地址为 0，则在 LCD 屏上便会显示出 4 个黑点，黑点中间会有 4 个白点出现（假如 1 是黑色）；对于 4 色则用 00、01、10、11 就可以表示出四种颜色，即用两位数据可以表示一位数据，那同样是 10101010，则对应于 LCD 屏上则显示的是颜色值为 10，长度为 4（8/2）的一条直线；同理，对于 8 位色（256 色），则 8 位数据才能表示出一个点的颜色值，10101010 在 LCD 屏上就只能显示为颜色值为 10101010 的点了。
- 有了上面的基础我们就可以很好的理解这个语句了：

  ```
  screensize = vinfo.xres * vinfo.yres * vinfo.bits_per_pixel / 8;
  ```

  即 FrameBuffer 的大小=LCD 屏的宽度 * LCD 屏的高度 * 每像素的位数 / 每字节的位数

  例如，一个 320*240 的黑白平，FrameBuffer 的大小为

  320 * 240 * 1 / 8 = 9600 （字节）

  而一个 320 * 240 的 16 位色 LCD 的 FrameBuffer 的大小则为

  320 * 240 * 16 / 8 = 153600（字节）

3） TFT 屏 16 位色的画点函数

```
void PutTft16Bit(U32 x,U32 y,U32 c)
{
    int iLineLength = vinfo.xres;
    if(x<vinfo.xres *2 && y<vinfo.yres*2)
        fbp[y*iLineLength + x/2]=( fbp[y*iLineLength + x/2]
        & ~(0xffff0000>>((x)%2)*16) ) | ( (c&0x0000ffff)<<((2-1-((x)%2))*16) );

}
```

有了画点函数，你还愁什么？图形汉字都可以搞定了吧！

4） CSTN 屏 12 位色的画点函数

```
int PutCstn12Bit(U32 x,U32 y,U32 c)
{
    U32 z;
    int iLineLength = vinfo.xres * 3 / 2;
    z=((x)%8);
    if(x<vinfo.xres*4 && y<vinfo.yres*2)
    {
        if((z%3)!=2)
        {
            fbp[y*iLineLength+x*3/8]=
            ( fbp[y*iLineLength+x*3/8]
            & ~(0xfff00000>>(((z/3)*4)+((z)%3)*12) )
            | ( (c&0xfff)<<(20-(((z/3)*4)+((z)%3)*12))) );
        }
        else
        {
            if(z==2)
            {
                fbp[y*iLineLength+x*3/8]=( (fbp[y*iLineLength+x*3/8]
            & ~(0xff)) | ((c&0xff0)>>4)  );
                fbp[y*iLineLength+x*3/8+1]=( (fbp[y*iLineLength+x*3/8+1]
            & ~(0xf0000000)) | ((c&0xf)<<28)  );
            }
            else if(z==5)
            {
                fbp[y*iLineLength+x*3/8]=( (fbp[y*iLineLength+x*3/8]
                & ~(0xf)) | ((c&0xf00)>>8)  );
                fbp[y*iLineLength+x*3/8+1]=( (fbp[y*iLineLength+x*3/8+1]
                & ~(0xff000000)) | ((c&0xff)<<24)  );
            }
        }
    }
    return 0;
}
```

注意，为了更便于代码书写，我在这个函数中将 fbp 定义为 static char * fbp，而在 TFT 屏 16 位色的画点函数中 fbp 的定义为 U16 * fbp，你可以根据需要进行修改。

5） TFT 屏 16 位色下显示 24 色位图函数

```
int showbmp24_bit()
{
    unsigned char *pdata_buf;
    int x,y;
    unsigned char r,g,b,i,j;
    unsigned int low,hight;
    unsigned int color;//颜色值
    int dot_offset;
    int iIndex;
    U32 lDataLength;
    //对于24位色位图,没有调色板信息,3个字节代表一个像素颜色
    printf("header.offset %d\n",header.offset);
    lDataLength = header.bmp_size - header.offset; //数据长度
    pdata_buf = (char *)malloc(lDataLength);
    //读取数据信息
    fseek(fp_bmp,header.offset,SEEK_SET);
    fread(pdata_buf,1,lDataLength,fp_bmp);
    for(y=0;y<info_header.width_y;y++)
    {
        dot_offset = 0;
        for(x=0;x<info_header.width_x;x++)
        {
            iIndex = (info_header.width_x *
                      info_header.width_y -
                      (info_header.width_x-x)
                      -y*info_header.width_x) * 3;
            r = (pdata_buf[iIndex+2] & 0xF8);
            g = (pdata_buf[iIndex+1] & 0xFC);
            b = (pdata_buf[iIndex] & 0xF8);
            hight = r | (g >> 5);
            low = (g<<3) | (b >> 3);
            color = (hight<<8) | low;
            PutTft16Bit(x+dot_offset,y,color);
            PutTft16Bit(x+dot_offset+1,y,color);
            dot_offset += 1;
        }
    }
    free(pdata_buf);
}
```

Bmp文件的格式可以参考网上的一些资料，另外在我的网站www.laoliu-soft.net上也有相关的资料可以参考，如果需要也可以直接找我要。

6） CSTN 屏 12 位色下显示 24 色位图函数

```
int showbmp24_bit()
{
    unsigned char *pdata_buf;
    int x,y,r,g,b,i;
    unsigned int color;//颜色值
    int dot_offset;
    int iIndex;
    U32 lDataLength;
    //对于24位色位图,没有调色板信息,3个字节代表一个像素颜色
    printf("header.offset %d\n",header.offset);
    lDataLength = header.bmp_size - header.offset; //数据长度
    pdata_buf = (char *)malloc(lDataLength);
    //读取数据信息
    fseek(fp_bmp,header.offset,SEEK_SET);
    fread(pdata_buf,1,lDataLength,fp_bmp);
    for(y=0;y<info_header.width_y;y++)
    {
        dot_offset = 0;
        for(x=0;x<info_header.width_x;x++)
        {
            iIndex = (info_header.width_x *
                      info_header.width_y -
                      (info_header.width_x-x)
                      -y*info_header.width_x) * 3;
            r = (pdata_buf[iIndex+2] & 0xF0)<< 4;
            g = (pdata_buf[iIndex+1] & 0xF0);
            b = (pdata_buf[iIndex] & 0xF0)>>4;
            color = r|g|b;
            PutCstn12Bit(x+dot_offset,y,color);
            PutCstn12Bit(x+dot_offset+1,y,color);
            PutCstn12Bit(x+dot_offset+2,y,color);
            PutCstn12Bit(x+dot_offset+3,y,color);
            dot_offset += 3;
        }
    }
    free(pdata_buf);
}
```

7） 呵呵，别忘了关闭设备哦

```
void closegraph()
{
    munmap(fbp,screensize);
    close(fb);
}
```

8） 另外关于如何显示汉字的 C 语言代码，在我的网站上有现成的源码《点阵字库西那时程序》，在 TC2.0 下都是可以跑的，她可以支持 12X12 点阵、14X14 点阵、16X16 点阵汉字及 ASC 码的显示，将画点函数和与设备相关的函数简单替换就可以了，移植起来应该不困难。对于有更高需求的朋友，如果 GB 库还不够，我也提供了 GBK 支持 2 万多个汉字的显示，只可惜只有 16X16 点阵的字库，没有小字体。显示方法我也提供了源码哦（因为网上没有找到相关资料，再加上老刘我比较笨，我研究了好长时间才成功，就让需要的朋友一起分享吧！）

6. 糊里糊涂写了好长时间，从 2004 年 12 月就开始了，一直写到了今天（2005 年 1 月 18 日），希望我的辛苦不会白费，希望这些东东对朋友们有所帮助，如果你发现错误或有好的意见，别忘了通知我啊laoliu@laoliu-soft.net，我及读到这片文章的朋友们会感谢你的哦！阿门！