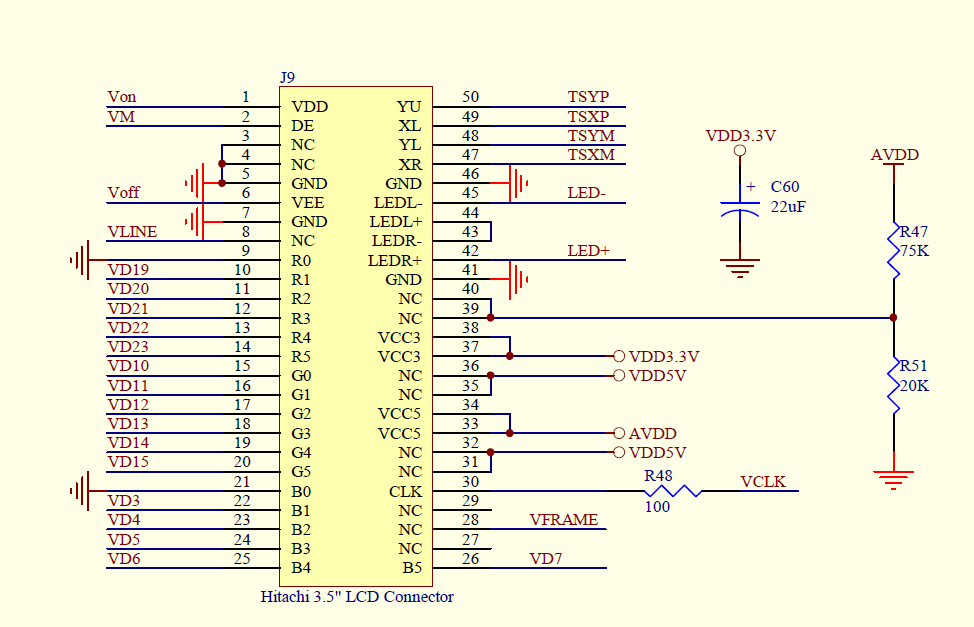
## lcd硬件原理



电子枪移动像素单位，打出不同颜色的光

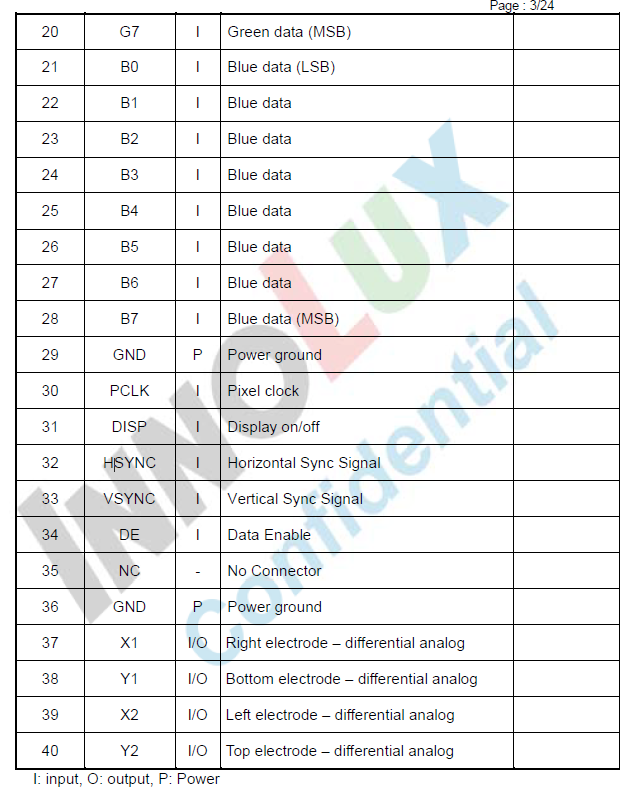
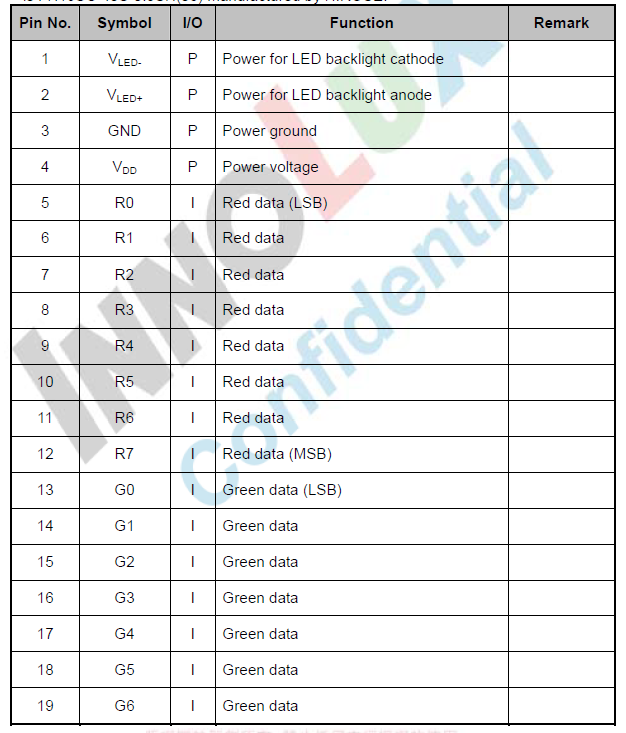
clk引脚每发出一个clk，电子枪移动一个像素

颜色有R G B引脚决定，内存里分配一块显存（frambuffer），lcd控制器从里面将数据读出来，通过RGB线传给电子枪，打到屏幕上

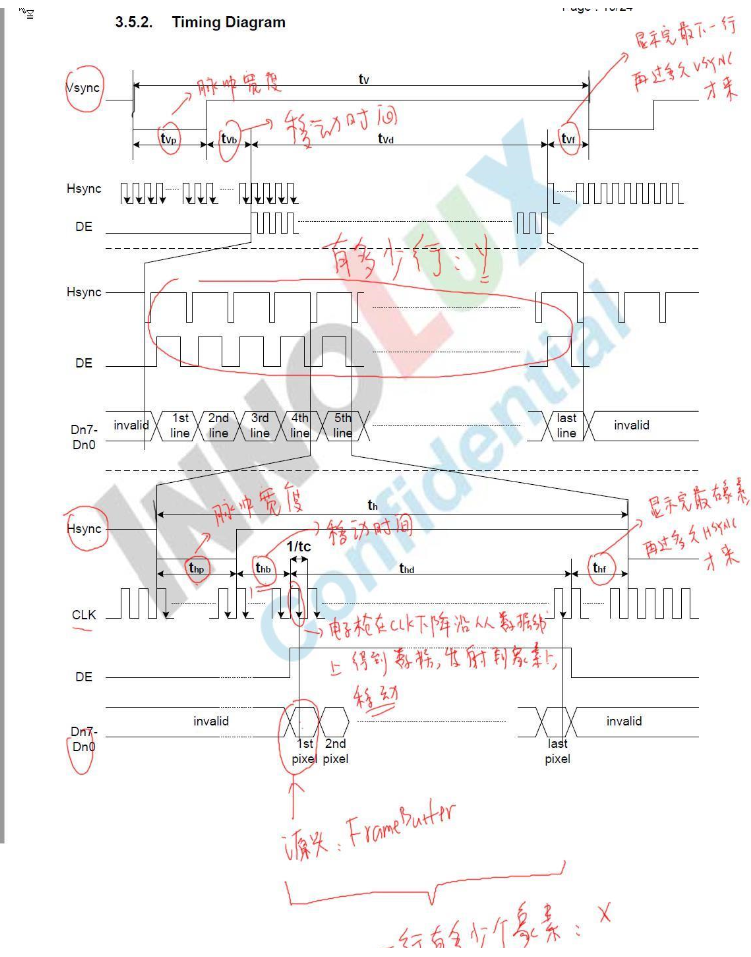
HSYNC引脚发出脉冲，电子枪跳到下一行

CSYNC引脚发出脉冲，电子枪跳会原点

所有的引脚信号由lcd控制器出



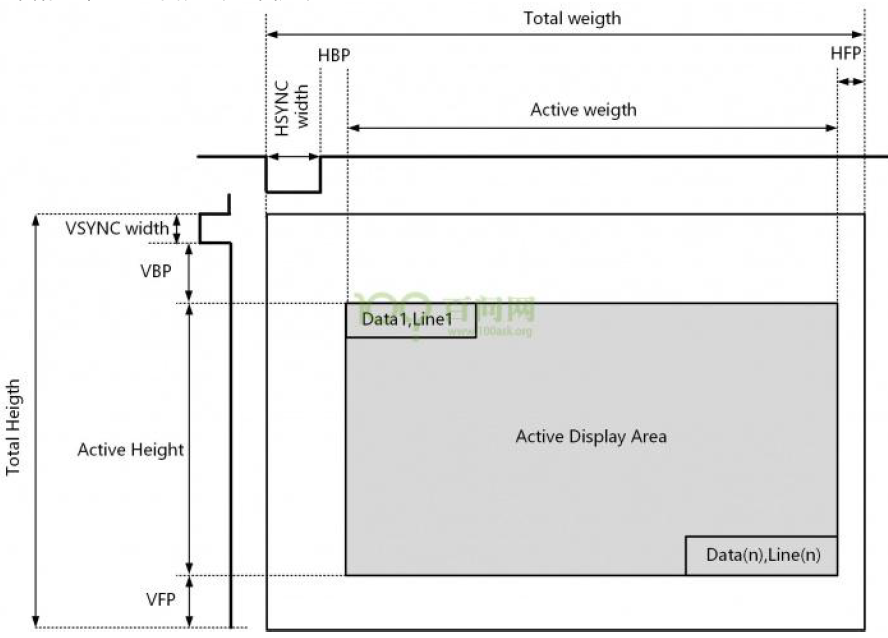
时序图

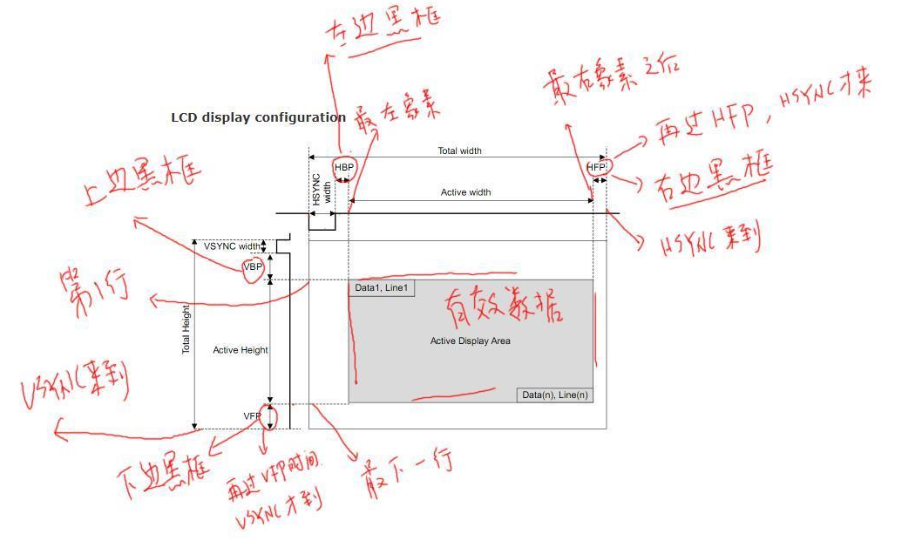


从像素开始分析，电子枪在下降沿从数据Dn0-7上得到数据，数据来源就是framebuff，电子枪从一行的最左边移动到最右，

打印完一行的最后一个数据后，收到Hsync行同步信号，一个Hsync大致分为五部分：看图

Vsync垂直同步信号同理





每个像素在framebuffer中占多少位bpp（bits per pixels），硬件上是3\*8 = 24 ，开发板使用的是16bpp,5 + 6 + 6 = 16，每个像素占16位数据

写程序的思路：

1. 查看lcd芯片手册，查看相关的时间参数、分辨率、引脚极性
2. 根据以上信息设置lcd控制寄存器，让其发出正确信号
3. 在内存里分配一个framebuff，在里面用若干位表示一个像素，再把首地址告诉lcd控制器

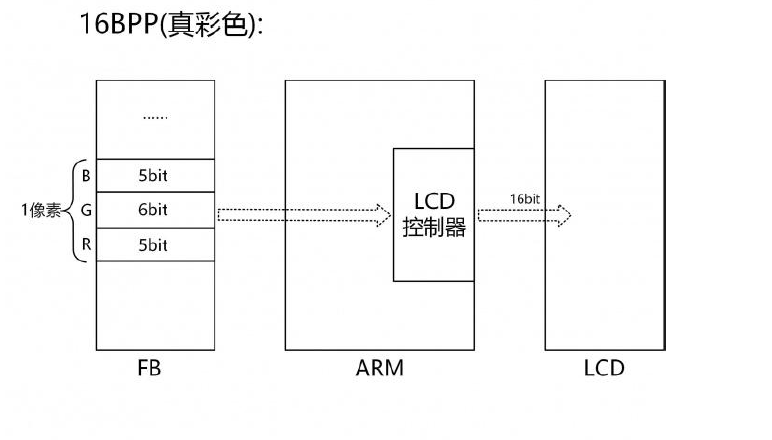
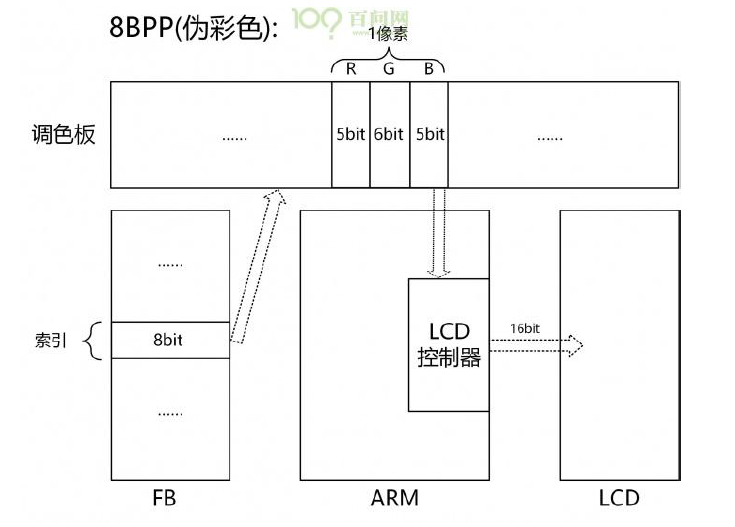
之后lcd控制器就能周二复制取出framebuff里的像素数据，配合其他控制信号，发送给电子枪，在lcd上显示出来，因此只需要在framebuff里填入相应的数据即可，硬件会自动完成显示操作

## s3c2440\_lcd\_controller

lcd控制器主要功能和需要的设置；

1. 取：从内存frambuff里取出某个像素数据，之后需要把framebuff地址，bpp，分辨率告诉lcd控制器
2. 发：配合其他信号把framebuff数据发送给lcd；设置lcd控制器时序、设置引脚极性

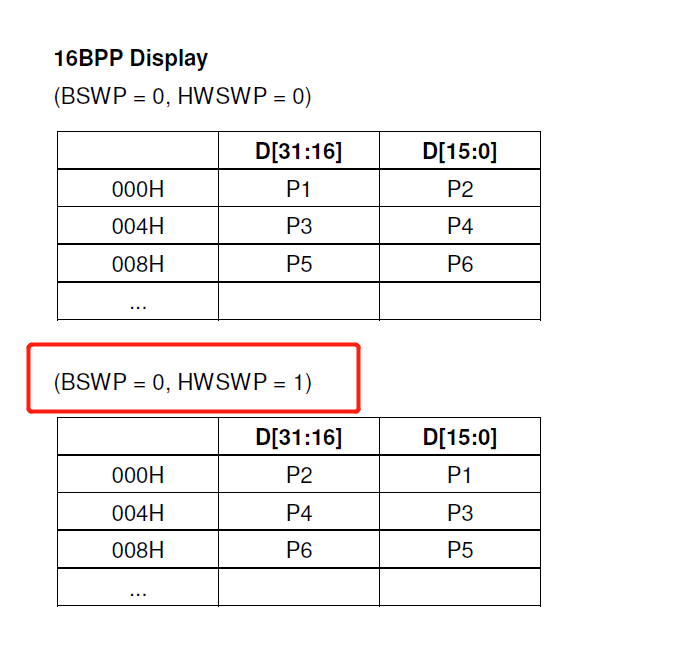
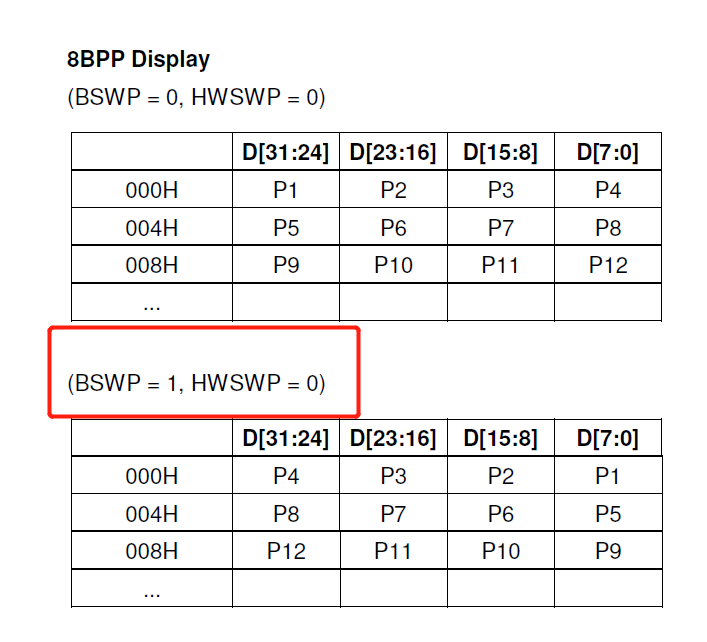
难点在于如何配合其他信号，需要阅读lcd芯片手册，

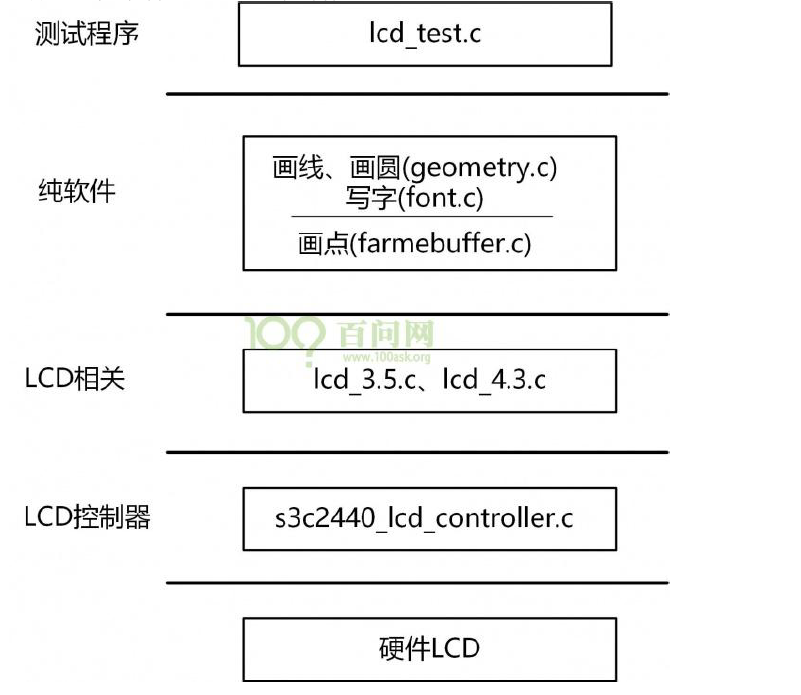
如果想节约内存，对颜色要求不高，那么就可以采用调色板的方式，调色板存放256个16bit的数据，framebuff值存放每个像素的索引，根据索引去调色板找到对应的数据传给lcd控制器，电子枪再显示

如果只显示同种颜色

16/24bppframebuff填充同值，8bpp可以修改framebuff也可以设置调色板为同种颜色，对于s3c2440有个临时调色板特性，一旦使用了临时调色板，不管framebuff里面是什么数据，都只能用临时调色板的数据

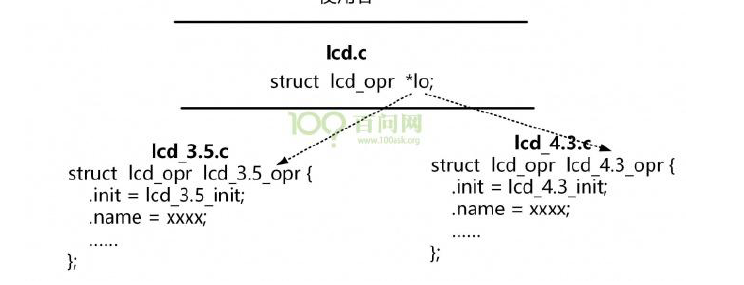
 

## 编程框架准备



如果有两种尺寸大小的lcd，如何快速切换

首先抽象出lcd3.5 lcd4.3的共同点，比如都有初始化函数init(),可以新建一个lcd.c，然后定义一个结构体



只需要在lcd.c里通过指针访问对应的结构体函数，也就调用了不同的init(),

## 抽象出结构体和lcd控制器

新建font.c frambuffer.c geometry.c lcd.c lcd\_4.3.c lcd\_controller.c s3c2440\_lcd\_controller.c lcd\_test.c

lcd的参数有引脚的极性、时序、bpp、分辨率等，结构体封装在lcd.h

对于有多个lcd的情况，在定义一个结构体，包含指针初始化函数和使能函数，放在lcd\_controller.h里面

lcd\_controller.c传入lcd参数，通过指针函数初始化对应的lcd控制器

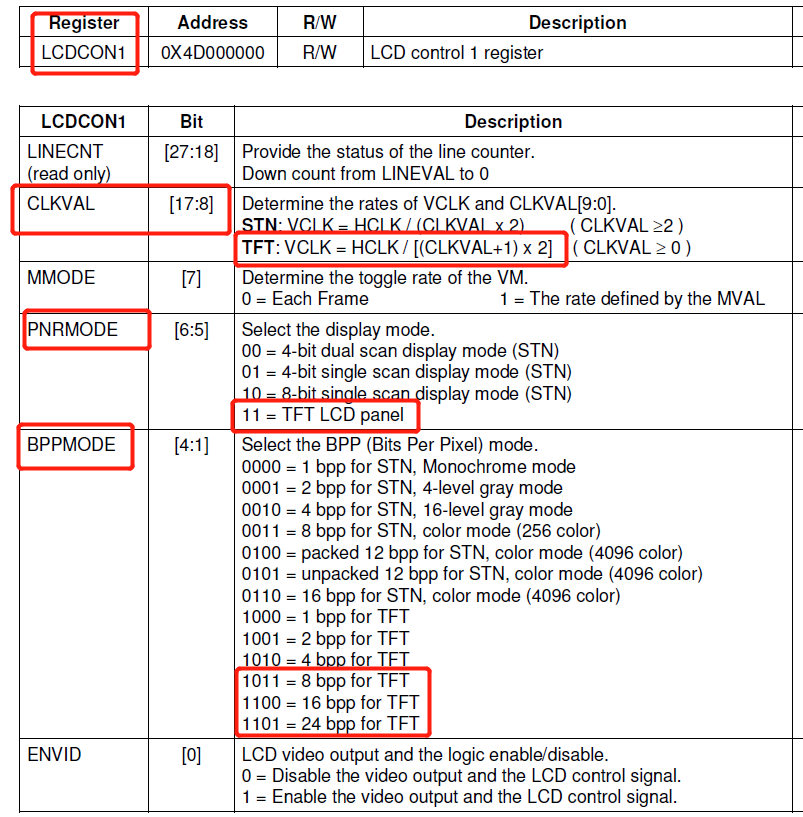
在s3c2440\_lcd\_controller.c还需要构造一个当前soc的lcd控制器结构体

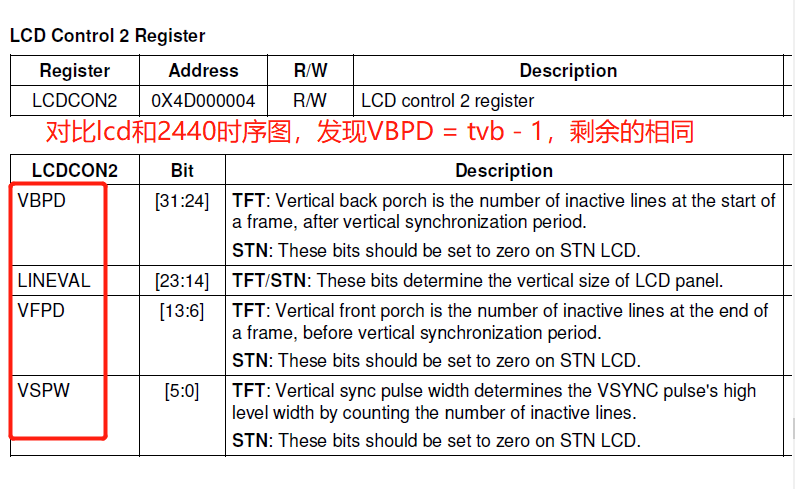
然后看手册逐个实现Init函数 使能函数

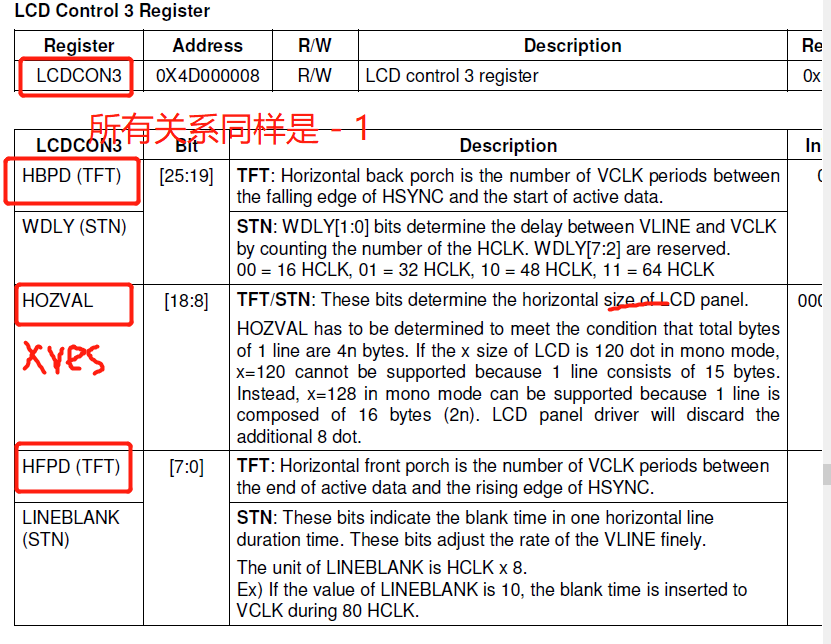
### 再这之前还要初始化引脚

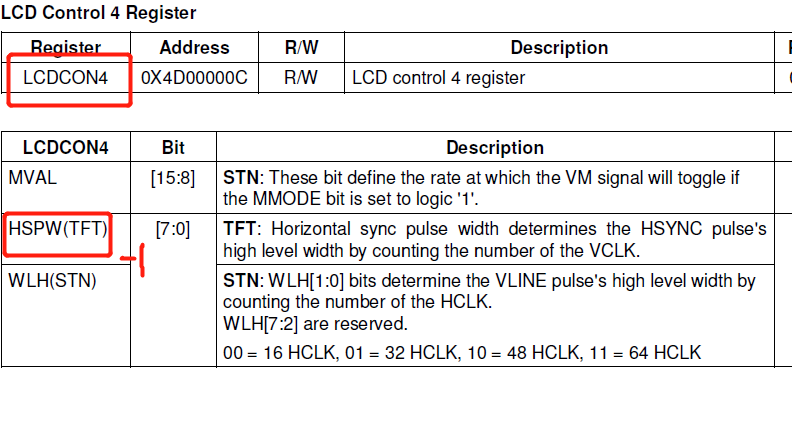
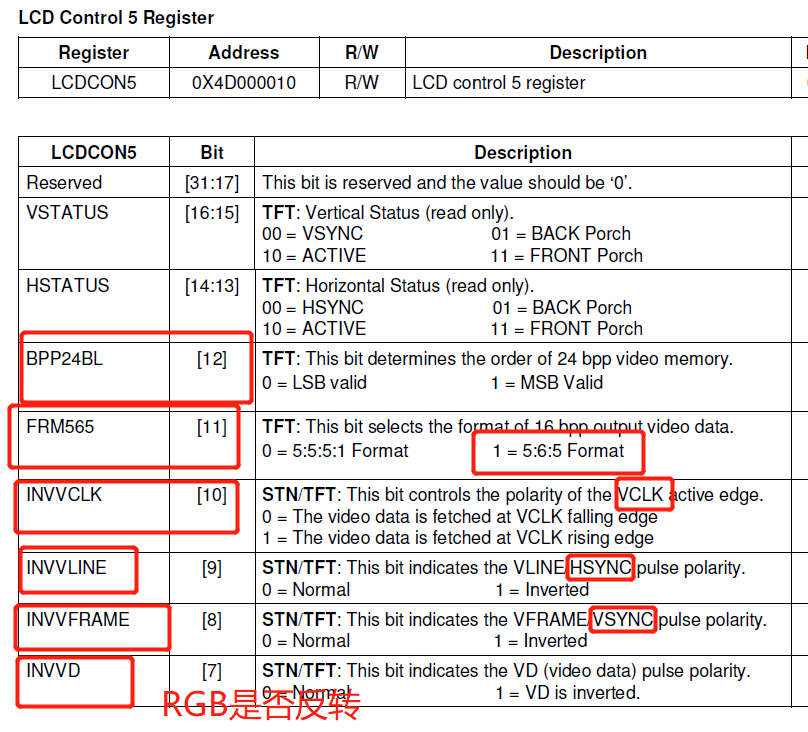


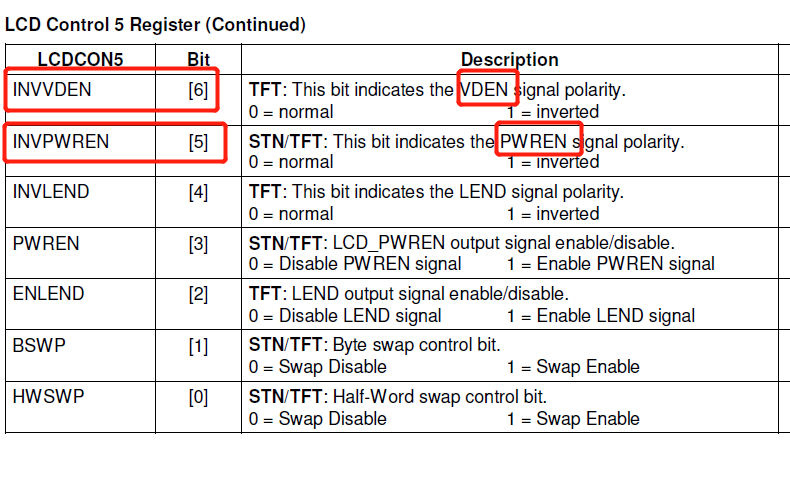
### 根据lcd传入的参数设置lcd控制器

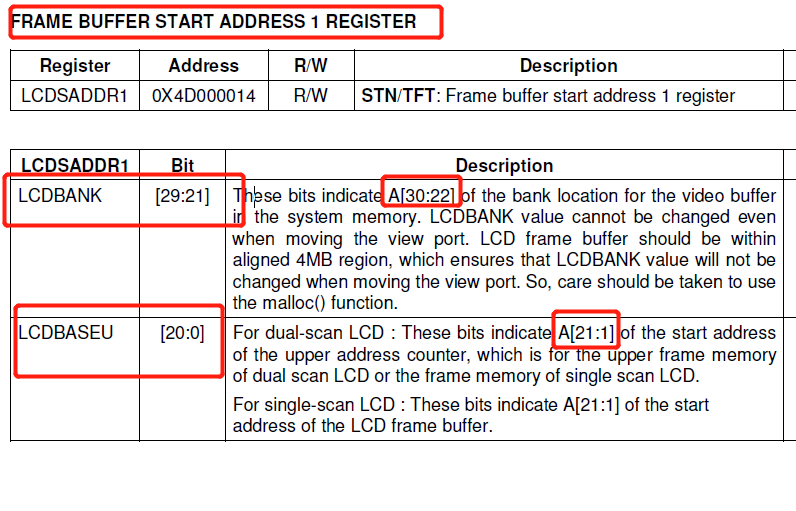


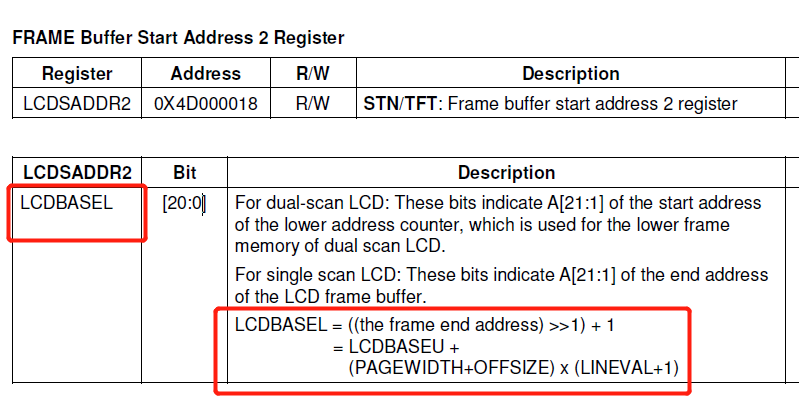




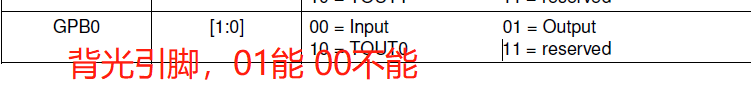
 

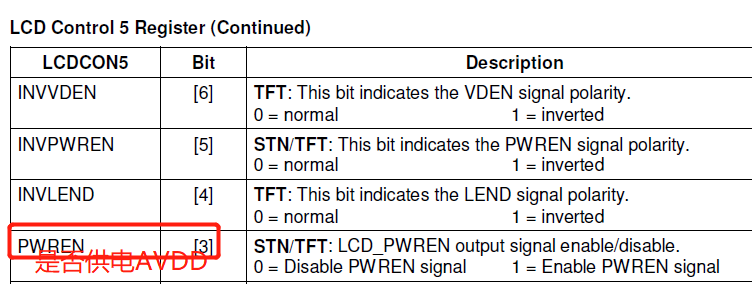


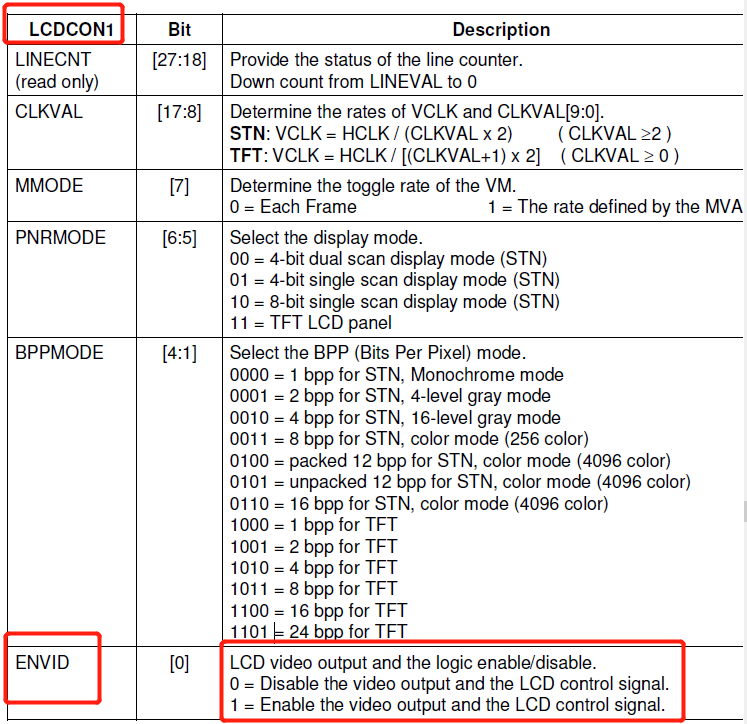




### 接下来设置使能

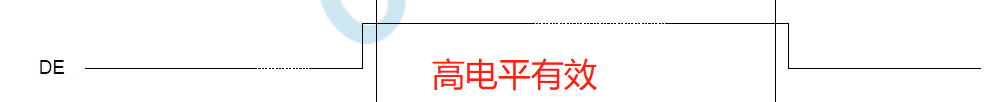




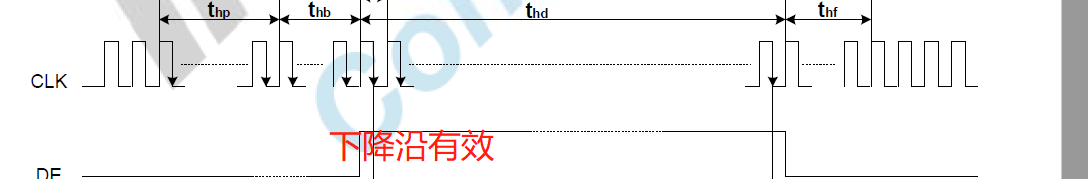


## lcd\_4\_3编程

### lcd\_4\_3 极性 看时序图



pwren看原理图，高电平有效



rgb 高电平

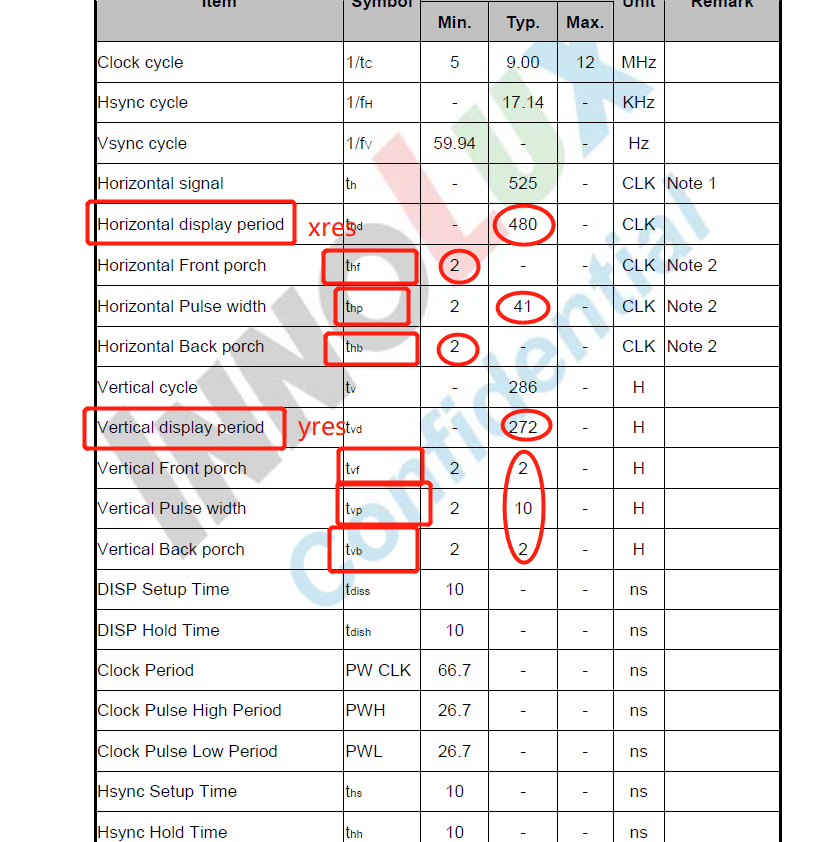
hsync



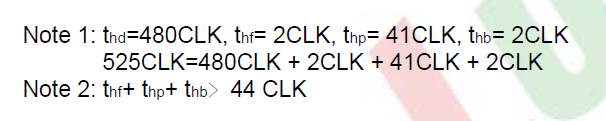
vsync

同低有效

### 时序，分辨率等







最后可算一下水平垂直周期总宽度是否等于其他值之和

fb\_base 结合start.S 0x33c00000

### 中间层控制lcd.c lcd\_controller.c

用lcd\_controller.c管理s3c2440\_lcd\_controller.c，向上接收传入的lcd参数，向下传给对应的lcd控制器。思路如下

1. 用数组保存下面各种lcd\_controller比如s3c2440\_lcd\_controller
2. 提供register\_lcd\_controller给下面的代码设置数组
3. 提供select\_lcd\_controller(name)给上面的代码选择某个lcd\_controller

最终s3c2440\_lcd\_controller.c里的register\_lcd\_controller()将自己放在p\_array\_lcd\_controller[]数组里，然后上层的lcd\_controller.c调用select\_lcd\_controller()传入想要选择的lcd控制器，在数组里找到名字相匹配的lcd控制器，并进行初始化

同理，通过lcd.c管理lcd\_4\_3.c思路如下

1. 用一个数组存放各类lcd的参数，比如lcd\_4\_3、lcd\_3\_5
2. 提供register\_lcd给下面的lcd来设置数组
3. select\_lcd (name)给上面的代码选择某个lcd

在lcd\_4.3.c把lcd参数注册进去

以后只需要在lcd.c里面选择某款lcd和lcd控制器即可，底层的只管添加种类，

最后lcd.c添加初始化函数

## 简单测试

lcd\_test.c所需步骤

1. 初始化lcd
2. 使能lcd（实际上调用的是lcd\_controller\_enable）
3. 获取lcd参数：fb\_base,xres,yres,bpp
4. 有了参数才可以往framebuff写数据

第三步，在lcd.c里实现get\_lcd\_params()

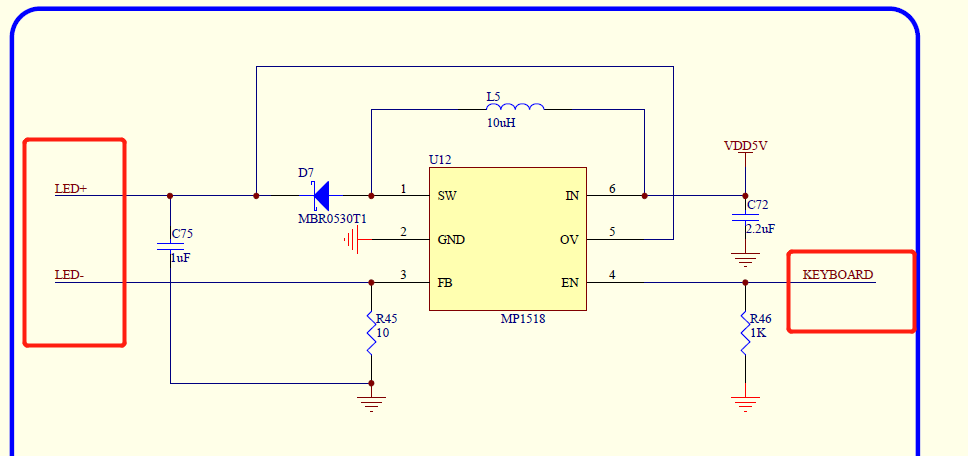
第四步，现在让屏幕全红，那么就要从framebuff基地址开始填充红色的数据。bpp = 16，R = 1，G = 0，B = 0所以数据位0xf700

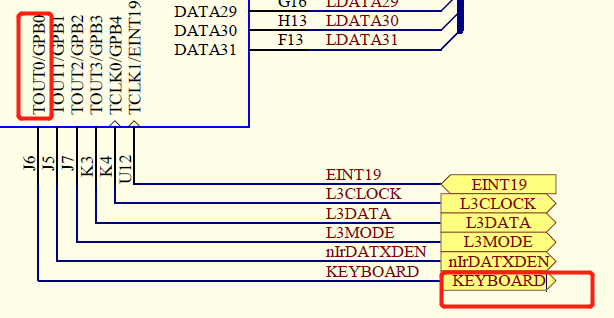
修改Makefile，出现错误，调试

### 最终发现个隐蔽的错误



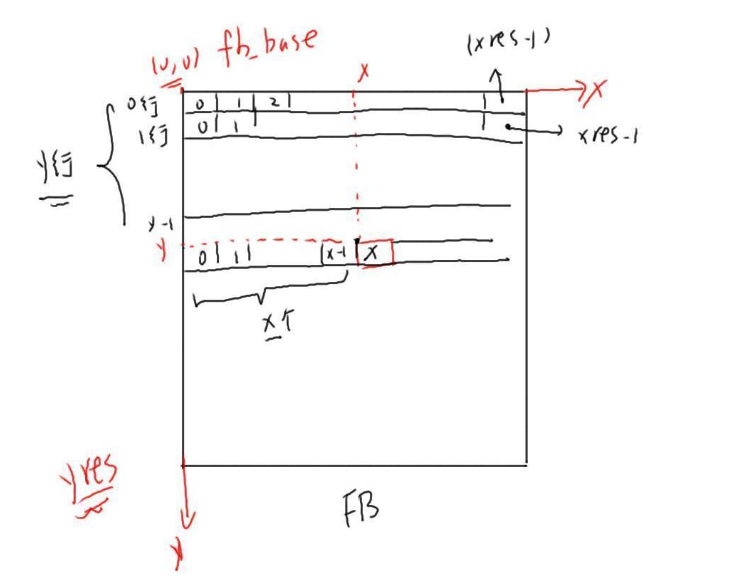
上面原理图有个LED+ LED-引脚，





output ，高电平有效

## 画点、文字



每个像素占bpp/8个字节

因此像素的起始地址：

= fb\_base + (xres \* bpp/8)\*y + (x \* bpp/8);

因此要获取这些数据

前面写过get\_lcd\_params

直接调用

对于8bpp，每个像素之占8位，1字节，用unsigned char类型

对于16bpp，每个像素之占16位，2字节，用unsigned short类型

对于32bpp，每个像素之占32位，4字节，用unsigned int类型

对三种情况设置不同的指针变量

传入坐标值

指向（坐标值）地址

\*p = color

对于传入的颜色数据，一般都是32bit，即0x00RRGGBB

对于8bpp，通过调色板索引实现，直接\*p = color

对于16bpp，需要进行颜色转换，需要写conver32bppto16bpp()函数

对于32bpp，大小刚好对应 ，同样直接 \*p = color

画点写完后

查找资料找到一份画线画圆的博客

复制下来，将其中的画点函数替换成上面写的

最后在main.c添加画线画圆测试代码

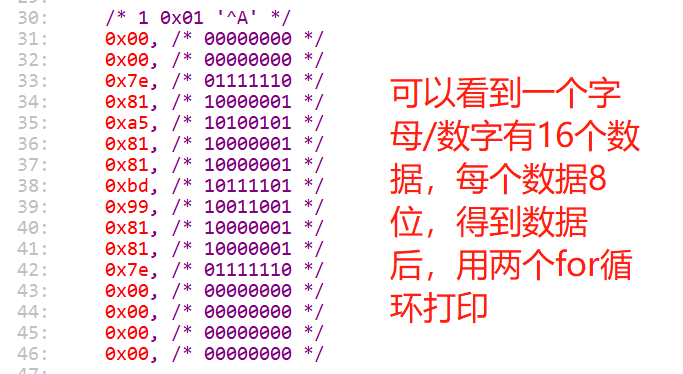
### 继续添加画文字

在内核中搜索font，打开font\_8x16.c可以看到里面有字符内容

这些数据在8\*16的区域里显示数字字母

新建一个font.c，根据字母的点阵在lcd上画文字，步骤

1. 根据带显示的字符的ascii码在font\_8x16中得到点阵数据
2. 根据点阵来设置对应像素的颜色
3. 根据点阵的某位决定是否描颜色



打印完一个字符后，打印一串字符

在每显示完一个字符后，x轴+8并考虑是否换行即可，注意\n\r

因为之前没有除法，在s3c2440\_lcd\_controller.c里



在lib1funcs.S里是有除法的，但功能不够强大，所以无法实现，所以这里可以通过以下步骤解决

