嵌入式系统原理及应用

第11章　LCD接口设计

张帆

中南大学自动化学院

第11章 LCD接口设计

11.1 LCD 控制器原理概述

11.1.1 LCD 控制器介绍

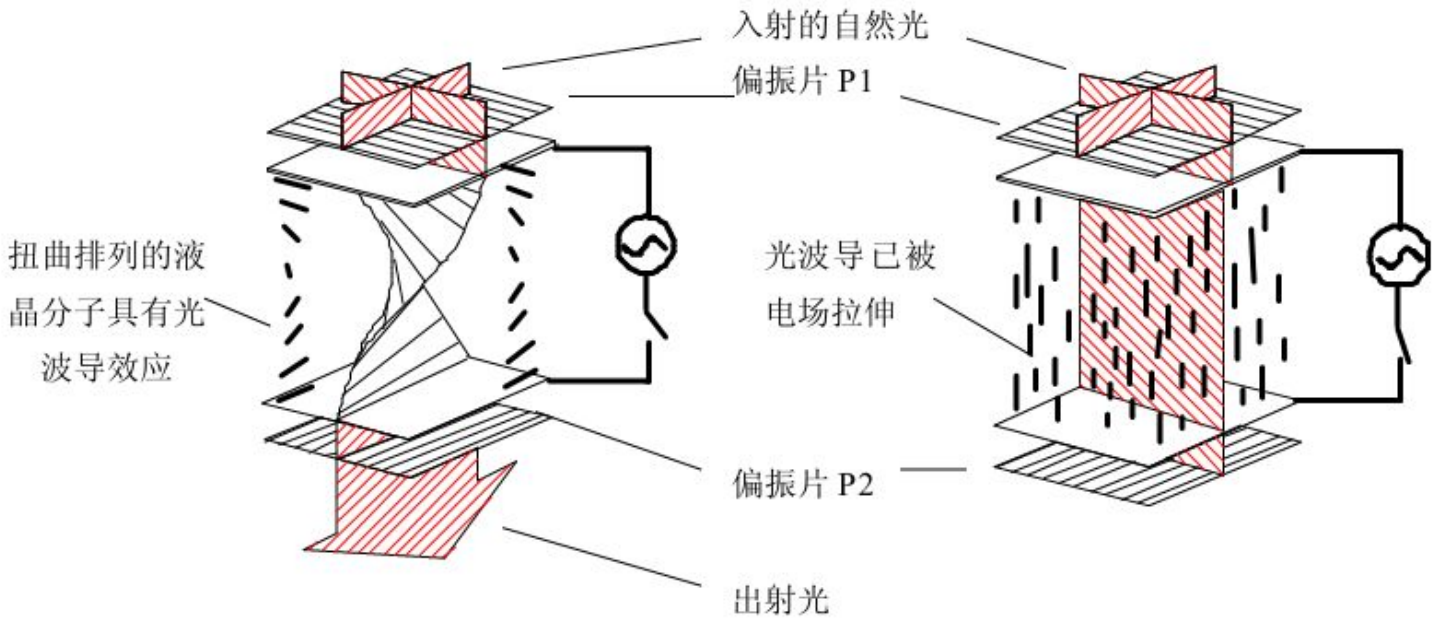
1．液晶屏的分类：按显示原理分为 STN 和 TFT 两种。

STN（Super Twisted Nematic，超扭曲向列）液晶屏：STN 液晶显示器与液 晶材料、光线的干涉现象有关，因此显示的色调以淡绿色与橘色为主。STN 液 晶显示器中，使用 x、y 轴交叉的单纯电极驱动方式，即 x、y 轴由垂直与水平 方向的驱动电极构成，水平方向驱动电极控制显示部分为亮或暗，垂直方向的 电极则负责驱动液晶分子的显示。STN 液晶显示屏加上彩色滤光片，并将单色 显示矩阵中的每一像素分成 3 个子像素，分别通过彩色滤光片显示红、绿、蓝 3 原色，也可以显示出色彩。单色液晶屏及灰度液晶屏都是 STN 液晶屏。

TFT（Thin Film Transistor，薄膜晶体管）彩色液晶屏：随着液晶显示技 术的不断发展和进步，TFT 液晶显示屏被广泛用于制作成计算机中的液晶显 示设备。TFT 液晶显示屏既可以在笔记本电脑上应用（现在大多数笔记本电 脑都使用 TFT 显示屏），也常用于主流台式显示器

LCD屏幕的工作原理

https://www.bilibili.com/video/BV1Sb411W7Zf/?spm_id_from=333.788.videocard.1



小知识

LCD：liquid crystal display 的简称，按驱动方式分为静态驱动（电路简单，效果好，成本高），简单矩阵驱动（扭转向列型(TN)和超扭转向列型（STN））以及主动矩阵驱动（TFT）。

（1）、 TN型驱动液晶，是LCD中最基本的，其他LCD都以TN型改进。他只能将入射光旋转90度，视角只有30度，色彩单一，对比度低，主要用于简单的数字符与文字的显示，如电子表及电子计算器等。

（2）、 STN型驱动液晶。可将入射光旋转180度至270度，也改善了视角，通过塔配色滤光片，将单色矩阵的任意像素分成3个子像素，红绿蓝。画面色彩对比度仍较小，反应速度也较慢，可作为一般的操作显示接口。

（3）、前两个都采用场电压驱动方式，如果现实尺寸加大，中心部分对电极变化的反应时间就会变长，显示器的速度跟不上，为了解决这个问题，主动矩阵驱动TFT被提出，他通过晶体管显示信号开启或者关闭液晶分子电压，从而避免了显示器对电场效应的依靠。

TFT LCD 的显示质量较TN/STN 更佳，画面显示对比度可达 150:1 以上，反应速度逼近 30ms 甚至更快，适用于 PDA、笔记本电脑、数码相机、MP4 等。

小知识

表 18.1 TN、STN 和 TFT 显示器的区别

类　别	TN	STN	TFT
原理	液晶分子，扭转 90°	扭转 180°～270°	液晶分子，扭转 90°
特性	黑白、单色低对比	黑白、彩色，低对比	彩色(1667 万色)，可媲美 CRT 显示器的全彩，高对比
动画显示	否	否	是
视角	30°以下	40°以下	80°以下
面板尺寸	1～3 英寸	1～12 英寸	37 英寸

11.1 LCD 控制器原理概述

2．液晶屏的显示

一块 LCD 屏显示图像不但需要 LCD 驱动器，还需要有相应的 LCD 控制器。通常LCD驱动器与 LCD 玻璃基板制作在一起，而 LCD 控制器则由外部电路来实现。许多 MCU 内部直接集成了 LCD 控制器，通过 LCD 控制器可以方便地控制 STN 和 TFT 屏。

驱动电路包括提供液晶屏的驱动电源和液晶分子偏置电压，以及液晶显示屏的驱动逻辑；

显示控制部分可由专门的硬件电路组成，也可以采用集成电路（IC）模块，比如 EPSON、Silicon Motion 的 显示卡驱动器等；还可以使用处理器外围 LCD 控制模块。

OLED与LCD
https://www.bilibili.com/video/BV1Wz411B7Tf/?spm_id_from=333.788.recommend_more_video.-1

小知识

LCD驱动器：LCD驱动器一般与LCD面板集成在一起，面板需要一定的模拟电信号来控制液晶分子，LCD驱动器芯片负责给面板提供控制液晶分子的模拟电信号，驱动器的控制信号（数字信号）来自于LCD控制器的提供的接口。

LCD控制器：LCD控制器集成在SoC内部，它负责通过数字接口向外部的LCD驱动器提供要显示的像素数字信号。它必须按照一定的时序和LCD驱动器通信，LCD控制器受SoC控制，SoC会从内存中拿出像素数据给LCD控制器并最终传给LCD驱动器。

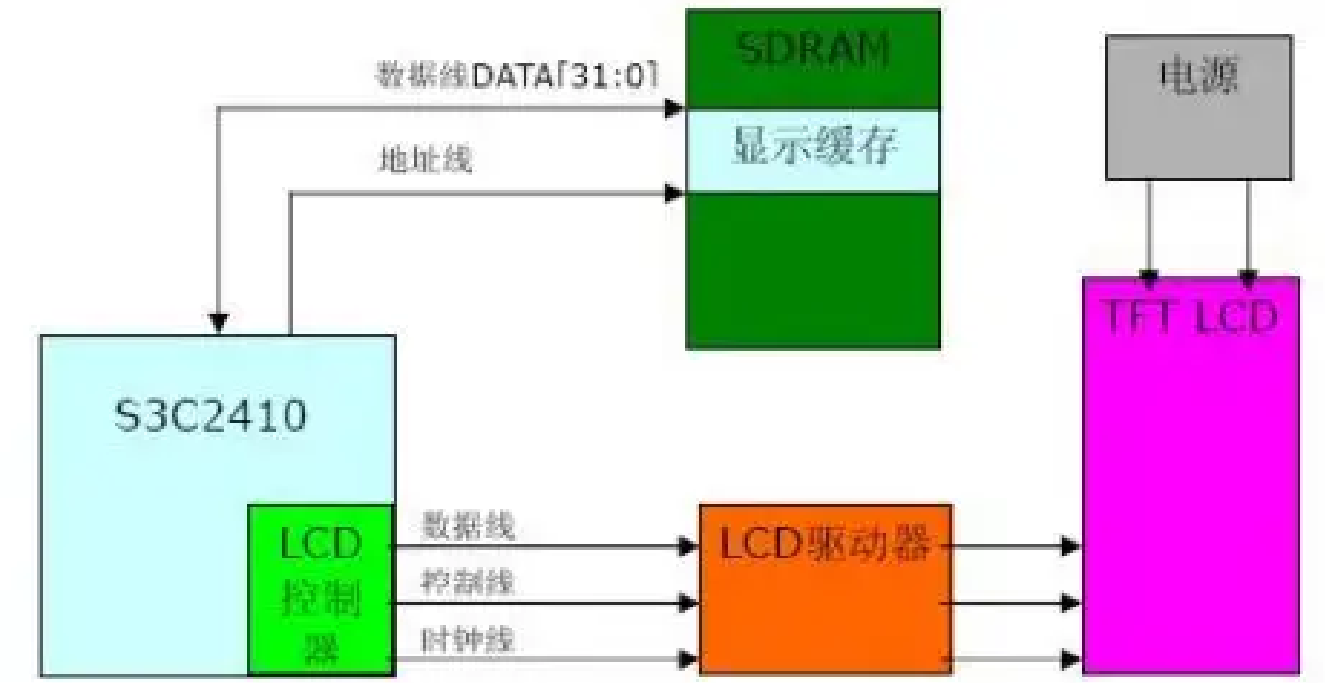
显存：SoC在内存中选一段内存，用来存放颜色数据，然后通过配置将LCD控制器和这一段内存连接起来，构成一个映射关系，一旦这个关系建立以后，LCD控制器就会自动从显存中读取像素数据传给LCD驱动器，LCD驱动器会自动的控制每个像素点的液晶分子，以形成最终的图像，建立这个映射以后就不需要SoC在来参与任何行为了。

总结一下：SoC控制LCD液晶显示的过程分为两个部分：

(1) SoC的LCD控制器引出一定的引脚与LCD驱动器连接，按照标准设置一定的时序;

(2) 把LCD要显示的像素信息放入内存中，在通过设置LCD控制器中的寄存器，与LCD控制器建立映射；之后过程就是LCD控制器芯片与驱动器芯片自动完成的事情了，整个LCD图像的显示过程就是这样。

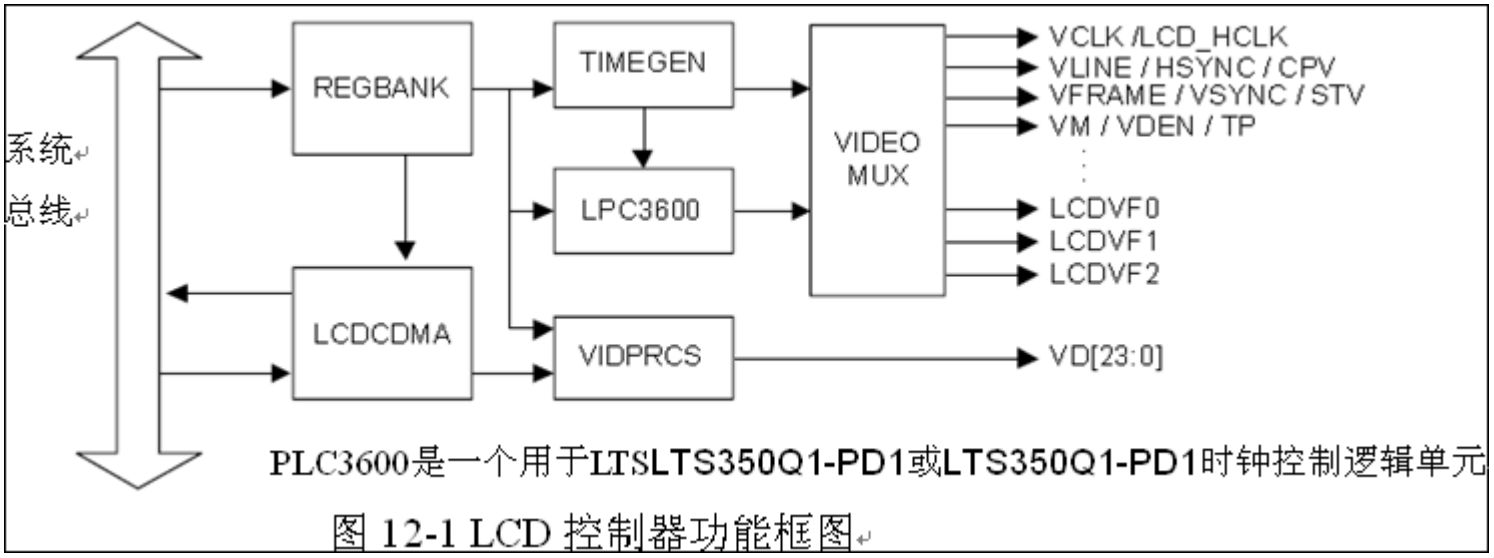
LCD系统结构


SDRAM
显示缓存
数据线DATA[31:0]
地址线
电源
TFT LCD
S3C2410
LCD
控制
器
数据线
控制线
时钟线
LCD驱动器

11.1 LCD 控制器原理概述

11.1.2 S5PC100 的 LCD 控制器介绍

S5PC100 处理器集成了 LCD 控制器，用于传输显示数据和产生控制信号。
支持屏幕水平和垂直滚动显示。
数据的传送采用 DMA方式
支持多种液晶屏(STN, TFT）。



PLC3600是一个用于LTSLTS350Q1-PD1或LTS350Q1-PD1时钟控制逻辑单元

图 12-1 LCD 控制器功能框图

LCD控制器

REGBANK有17个可编程寄存器组成，用来配置LCD控制器的各项参数。

LCDCDMA是专用DMA通道,自动从帧缓冲中传输视频数据到LCD控制器,利用DMA,视频数据可不经过CPU干涉就显示在屏幕上。

VIDPRCS接受从LCDCDMA来的视频数据并在将其改变到合适数据格式后经VD[23:0]将之送到LCD驱动器。

TIMEGEN产生LCD屏所需要的VFRAME、VLINE、VCLK、VM等控制信号。

Video Mux 控制器用于决定选用哪一个显示控制器来控制显示接口

11.1 LCD 控制器原理概述

STN LCD 显示器性能如下

（1）支持 3 种类型的扫描方式：4 位单扫描、4 位双扫描和 8 位单扫描。

（2）支持 256 色和 4096 色彩色 STN LCD。

（3）典型的实际屏幕大小是：640×480、320×240、160×160 等。

（4）最大虚拟屏幕占内存大小为 4 MB。

（5）256 色模式下最大虚拟屏幕大小：4096×1 024、2048×2 048、1024×4096 等。

TFT LCD 显示器性能如下

（1）支持 1、2、4 或 8 bpp 调色彩色显示。

（2）支持 16 bpp 和 24 bpp 非调色真彩显示。

（3）在 24 bpp 模式下，最多支持 16 种颜色。

（4）支持多种屏幕大小。

（5）典型的实际屏幕大小是：640×480、320×240、160×160 等。

（6）最大虚拟屏幕占内存大小为 4 MB。

（7）64K 色模式下最大虚拟屏幕大小：2048×1024 等。

11.1 LCD 控制器原理概述

1．S5PC100 LCD 控制器功能简述

S5PC100 所集成的 LCD 控制器功能很强大，其中包含了一个本地总线传输图像数据的逻辑模块，以及内置的图像处理单元。这些模块都可通过总线连接至外接的 LCD 接口

LCD 接口包含了3 种类型：RGB 接口（本书所讲）、间接的 I80 接口、ITU-RBT.601/656 接口

显示控制器支持**最多 5 个叠加图像窗口**，每个窗口都支持多种图像格式以及 256 灰度级绑定、颜色锁定、x-y 坐标控制、软件卷动、可变的窗体尺寸等。

显示控制器支持多种颜色格式，例如 RGB（1 bpp～24 bpp）、YcbCr4:4:4（限于本地总线），可编程支持不同需求的图像像素、数据线宽度、时序、刷新率。

11.1 LCD 控制器原理概述

2．LCD 外部接口信号

S5PC100 的 LCD 控制器包括了两个时序部分，一个是针对于 RGB 接口、 ITU-RBT.601/656 接口的时序，一个是针对间接 I80 接口的时序。

本章将重点介绍关 于 RGB 接口的控制器部分。

RGB VIME 产生的控制信号有
VCLK（LCD 时钟）：新的一个点图像数据开始传送
VSYNC（垂直同步信号）：新的一屏图像数据开始传送。
HSYNC（水平同步信号）：新的一行图像数据开始传输
VDEN（数据有效信号）

VD[23:0]：每个点的输出图像数据（1bit or 8bit or 16bit or 24bit）

#FF0000 （纯红），#008000 （纯绿），#0000FF （纯蓝）

11.1 LCD 控制器原理概述

下图所示为 LCD-RGB 接口时序。

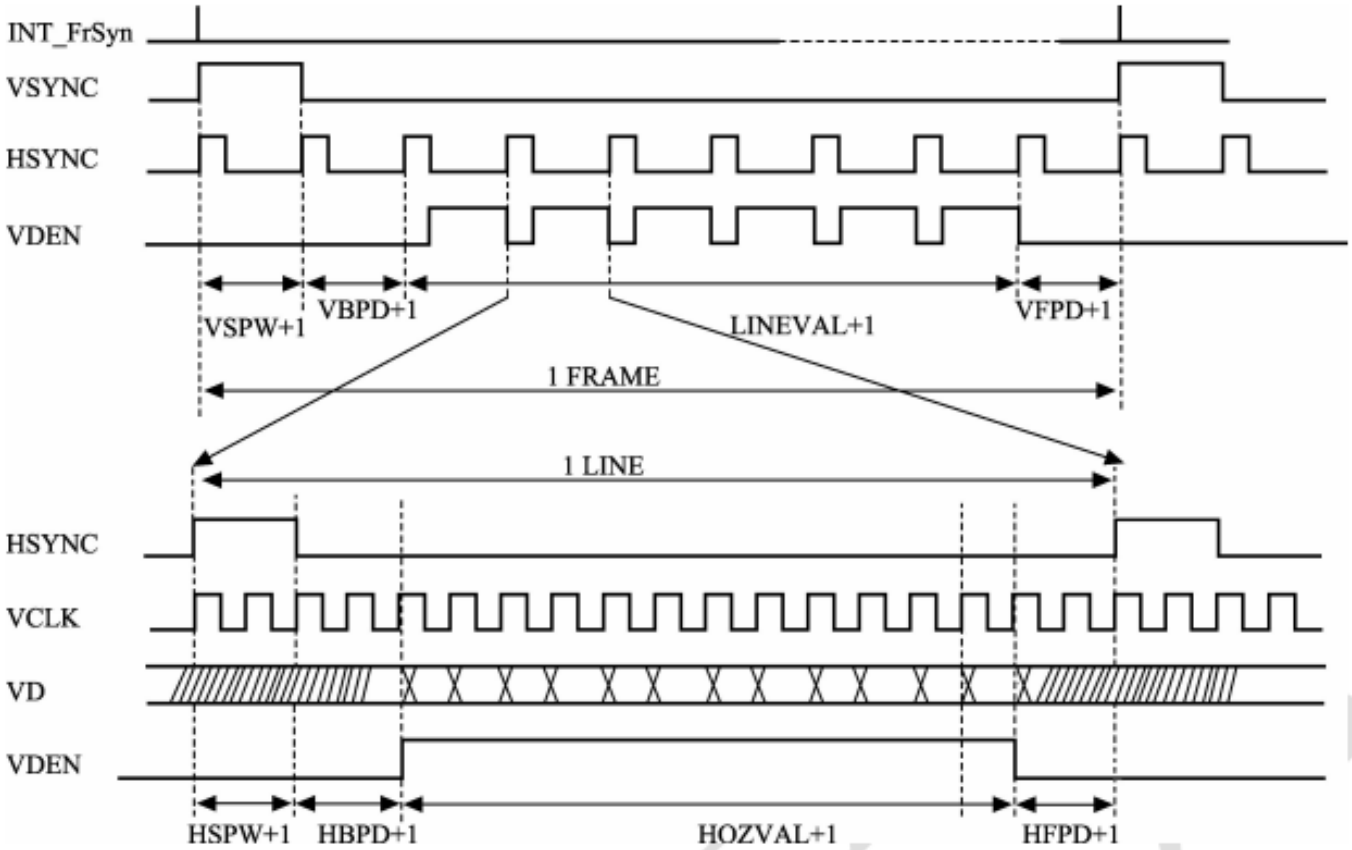


图 17-1　LCD RGB 接口时序图

小知识

- ＶＢＰＤ：表示在一帧图像开始时，帧同步信号以后的无效行数，垂直同步信号后肩；
- ＶＦＢＤ：表示在一帧图像结束后，帧同步信号以前的无效行数，垂直同步信号前肩；
- ＶＳＰＷ：表示垂直同步脉冲的宽度，用行数计算 ；

- ＨＢＰＤ：表示从水平同步信号开始到一行有效数据开始之间的ＶＣＬＫ的个数，水平同步信号后肩
- ＨＦＰＤ：表示一行有效数据结束到下一个水平同步信号开始之间的ＶＣＬＫ的个数，水平同步信号前肩
- ＨＳＰＷ：表示水平同步信号的宽度，用ＶＣＬＫ计算

小知识

- 在芯片中，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。这些值是通过对应的寄存器VIDTCON0、VIDTCON1和VIDTCON2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。

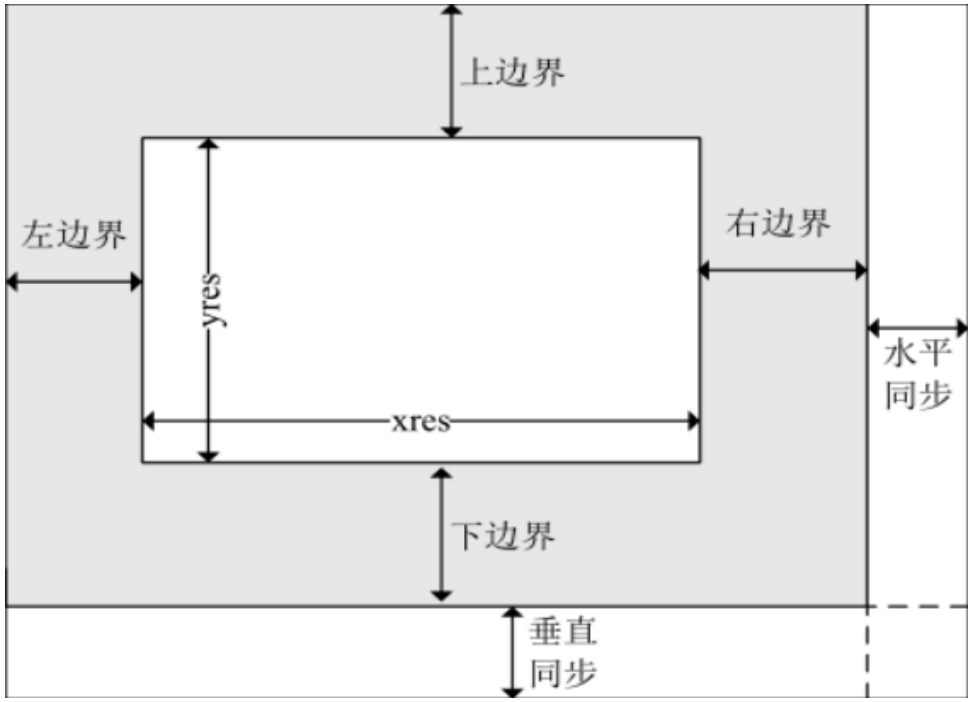
- 水平同步信号的脉宽、前肩和后肩分别为30、20和38，则HSPW＝30－1，HFPD＝20－1，HBPD＝38－1；

- 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。

小知识

帧同步信号VSYNC，每发出一个脉冲，都意味着新的一屏图像数据开始发送。行同步信号HSYNC，每发出一个脉冲都表明新的一行图像资料开始发送。在帧同步以及行同步的头尾都必须留有回扫时间。这样的时序安排起源于 CRT 显示器电子枪偏转所需要的时间，但后来成为实际上的工业标准，因此 TFT 屏也包含了回扫时间。

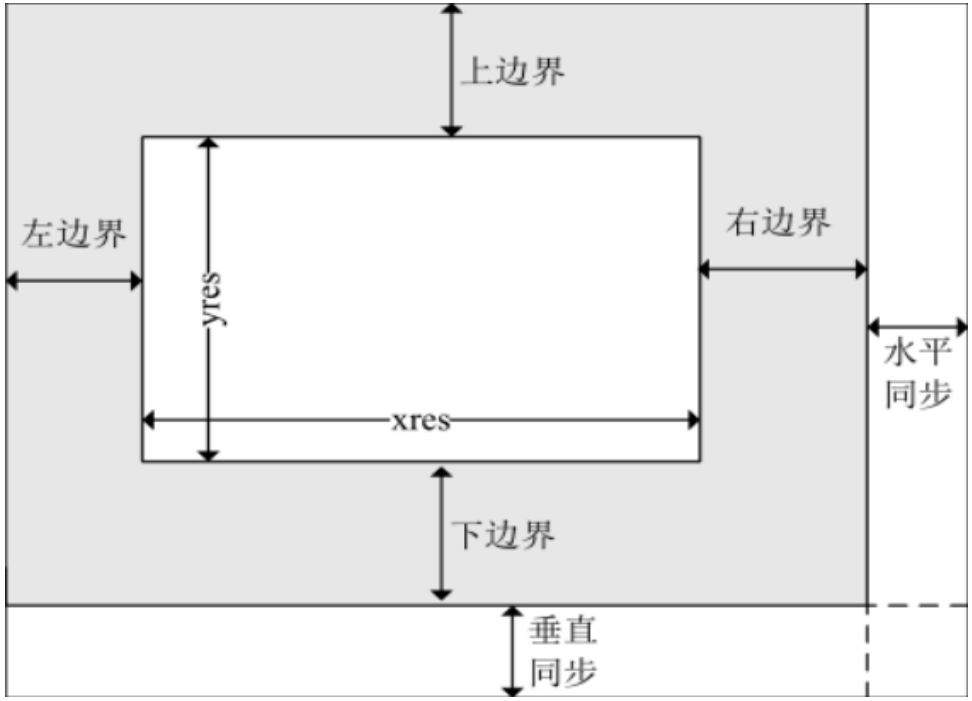
下图给出LCD 控制器中应该设置的 TFT 屏的参数，其中的上边界和下边界即为帧切换的回扫时间，左边界和右边界即为行切换的回扫时间，水平同步和垂直同步分别是行和帧同步本身需要的时间。xres 和 yres 则分别是屏幕的水平和垂直分辨率，常见的嵌入式设备的 LCD 分辨率主要为 320×240、640×480 等。



小知识

1、注意这些数字的单位。H开头的三个单位都是VCLK(像素时钟)，V开头的三个单位是行扫描时间。这样设置的好处是我们改变了像素时钟的设置时，不用改变这里的时序参数。

2、这些时序参数如果没设置好会影响什么？屏幕会跑偏。



11.1 LCD 控制器原理概述

17.1.3 S5PC100 的 LCD 控制器操作

基于前述接口时序图，简单看看LCD 的控制流程，给出几个简单公式：
HOZVAL = (Horizontal display size -1)
LINEVAL= (Vertical display size -1)
VCLK(Hz)= HCLK/(CLKVAL+1)　where CLKVAL >= 1

后面在配置寄存器的时候都需要使用到上述 3 个公式，这里先提出来。

VCLK(像素时钟频率)
HCLK（VCLK的允许最大值，手册会给具体数值）
目的：计算CLKVAL（分频）

11.1 LCD 控制器原理概述

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，它们分别叫作水平同步信号脉宽（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，必须包括这些信号。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。

11.1 LCD 控制器原理概述

在 S5PC100 中，还需要重点考虑 alpha 绑定机制，因为它是 5 个窗口叠加共同成像的原理，因此需要配置一下 alpha 绑定方程，如图所示。

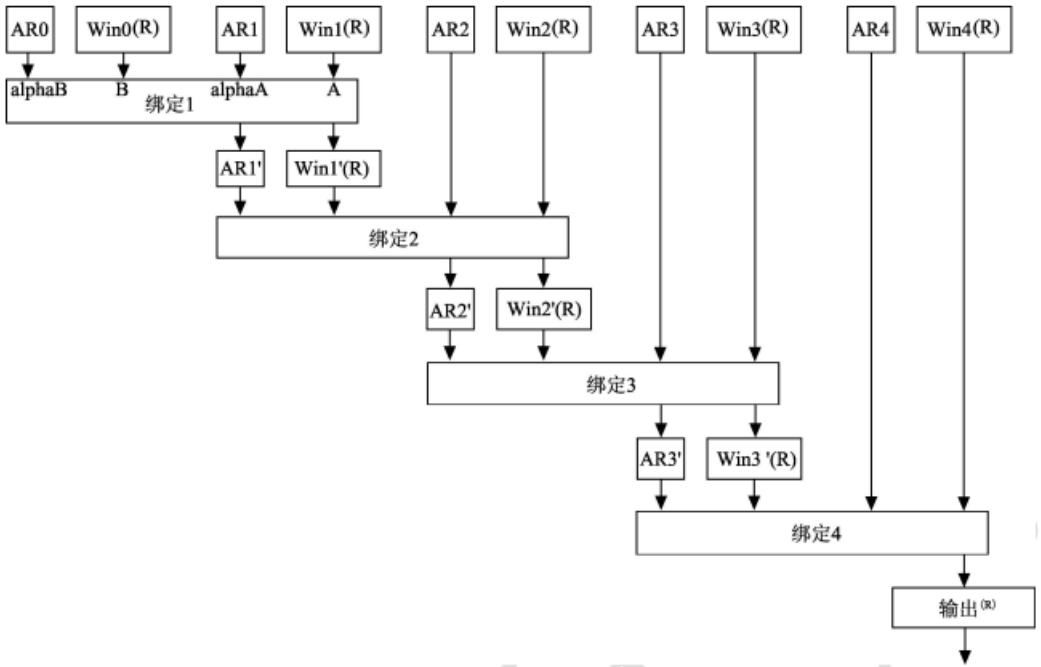


图 17-2　alpha 绑定方程

小知识

虚拟屏幕叠加

(1)虚拟屏幕的意思是，我们平时看到的屏幕上显示出来的场景实际是很多个屏幕显示叠加在一起的效果（譬如新闻图像、电视台台标、下方飘动的字幕新闻）

(2)像S5PC100的LCD控制器中有5个虚拟屏幕Window0到Window4，虚拟屏幕不存在于真实而存在于内存中。（LCD显示时实际是显示的是对应的内存中的显存区域的数值）虚拟屏幕其实就是一个内存中的显存区域，有几个显存区域就有几个虚拟屏幕，但是这些虚拟屏幕都被映射到一个真实的显示屏上面，所以你看到的显示效果实际是这几个虚拟屏幕的显示内容的叠加。（叠加时要注意上面一层会覆盖下面一层，所以要注意谁在前谁在后，设置寄存器时有这个选项）

(3)使用虚拟屏幕而不是整个LCD使用一个显存是有一定好处的：第一，可以保证不污染源图像，方便程序处理；第二，可以减少屏幕刷新，提高显示效率，减少CPU工作量。

小知识

虚拟显示

(1)如何实现在小分辨率的屏幕上（真实）显示大分辨率的图像

(2)细节上，我们需要屏幕上看到不同图像时，需要对显存区域进行刷新。即使我们只需要屏幕显示移动一点点，整个屏幕对应的显存空间也需要整个重新刷新，工作量和完全重新显示一幅图像是一样的。这个显然不好，这样CPU刷新屏幕的工作量太大了，效率很低。

(3)如何能够在显示一个大图片的不同区域时让CPU刷新屏幕工作量减少？有，方法就是虚拟显示。具体做法就是在内存中建立显示缓存的时候实际建立一个很大的区域，然后让LCD去对应其中的一部分区域作为有效的显示区域。将来要显示大图像时，直接将大图像全部一次性加载入显示缓存区，然后通过移动有效显示区域就可以显示大图像的不同区域了。

11.1 LCD 控制器原理概述

图中可以看到，每一次的叠加由两个窗口进行，窗口的顺序如下。

（1）X0 = 窗口 0 与窗口 1。

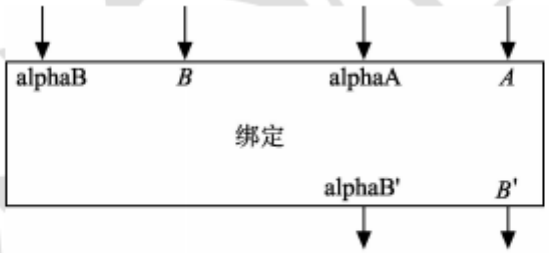
（2）X1 = 窗口 X0 与窗口 2。

（3）X2 = 窗口 X1 与窗口 3。

（4）X3 = 窗口 X2 与窗口 4。

最后的 X4 为 LCD 所呈现的图像，

绑定方程如下所示。



$$B' = a \times A + b \times B$$

$$\text{alphaB}' = p \times \text{alphaA} + q \times \text{alphaB}$$

a/b/p/q =0
or 1
or alphaA
or 1-alphaA
or alphaB
or 1-alphaB
or A
or 1-A
or B
or 1-B

图 17-3　绑定方程

11.1 LCD 控制器原理概述

B' 是一个色值函数，由 A 的色值分量与 B 的色值分量线性叠加而成。此处，A 的色值来自 win(n+1)，B 的色值来自 win(n)，a、b 则是线性因子（需要配置）。

同理 alphaB'是一个 B'色值的伴随灰度值函数(一组 alpha 通道)。 与前一个方程同构。p、q 也是线性因子，这样 4 个因子决定了两个窗口的最终叠加色值及伴随 alpha 的值

在 LCD 控制器中，只需要配置 WINn blending equation control register，并将具体的因子配好后，就可以实现了，注意 5 个窗口需要同时配置方程。

再来重点考察一下 alphaB 灰度值，它是一个 8 位 s 的值，0~255 分别代表了不同的灰度级。从方程中可以看出，如果一个 alpha 值与一个色值相乘后，就会得到一个 该色值的分量，这样两个窗口的叠加就靠不同的灰度来确定。换句话说，如果要使得 win1-win4 窗体整体透明，可以设置每层 alpha value 都为 0，并且色值方程配置 p_n=0， q_n=0，a_n=alphaA，b_n=(1-alphaA)，其中 n={1,2,3,4}，即可实现 Window 0 显示，而其 他窗口透明。当然，你也可以只使能 Windows 0，就不用考虑每一层的 alpha 值了。

11.1 LCD 控制器原理概述

17.1.4 LCD 控制器寄存器

由于在 S5PC100 中 LCD 控制器的寄存器众多，这里只简单介绍一下读者要用到、并且很重要的寄存器，见下表，如果想知道更多的信息，请参看 S5PC100 手册。

11.1 LCD 控制器原

17.1.4 LCD 控制器寄存器

VIDCON0，R/W,

ADDRESS=0xEE000000

主要：VIDOUT，CLKVAL
ENVID：LCD 视频输出逻辑使能/禁止

域	位	描　　述	复位值
保留	[31]	保留	0
INTERLACE_F	[29]	逐行扫描方式或者间隔扫描方式 0 = 逐行扫描 1 = 间隔扫描方式(only ITU601/656 Interface)	0
VIDOUT	[27:26]	输出格式： 000 = RGB I/F 001 = ITU601/656 010 = Indirect I80 I/F for LDI0 011 = Indirect I80 I/F for LDI1	000
PNRMODE	[18:17]	选择显示模式 (Where, VIDOUT[1:0] == 2'b00). 00 = RGB 格式 (RGB) 01 = RGB 格式 (BGR) 10 = Serial Format (R->G->B) 11 = Serial Format (B->G->R)	00
CLKVALUP	[16]	选择 CLKVAL_F 刷新时序的模式 0 = 总在刷新 1 =当一帧开始时刷新	0
CLKVAL_F	[15:6]	决定 VCLK 的速率及 CLKVAL[7:0] VCLK = HCLK / (CLKVAL+1) 且 CLKVAL >= 1 **Note.** 1. VCLK 最大值是 66 MHz 2. CLKSEL_F 寄存器选择的时钟源	0
VCLKFREE	[5]	VCLK 控制方式 0 =普通模式（ENVID） 1 =自由模式	0
CLKDIR	[4]	选择时钟源的通道 0 = 直接获取时钟（VCLK = Clock source） 1 = 除以分频值	0
CLKSEL_F	[2]	选择时钟源 0 = HCLK 1 = SCLK_LCD	0
ENVID	[1]	数据输出使能位 0 =禁止 1 =使能	0
ENVID_F	[0]	当前帧结束使能位 0 =禁止 1 =使能 如果该位被设置，则该位直到一帧结束时才被禁止	0

11.1 LCD 控制器原理概述

17.1.4 LCD 控制器寄存器

VIDCON1,R/W,

ADDRESS=0xEE000004

主要：极性

域	位	描　　述	复位值
LINECNT (read only)	[26:16]	提供 line counter 的计数值（read only） 从 0 到 LINEVAL	0
FSTATUS	[15]	场状态（read only） 0 = 非平坦场 1 = 平坦场	0
VSTATUS	[14:13]	垂直状态（read only） 00 = VSYNC　　01 = 后肩 10 = ACTIVE　　11 = 前肩	0
保留	[12:8]	保留	
IVCLK	[7]	VCLK 极性 0 = VCLK 下降沿取数据 1 = VCLK 上升沿取数据	0
IHSYNC	[6]	该位决定了 HSYNC 脉冲极性 0 = 普通　　1 = 反转	0
IVSYNC	[5]	该位决定了 VSYNC 脉冲极性 0 = 普通　　1 = 反转	0
IVDEN	[4]	该位决定了 VDEN 信号极性 0 = 普通　　1 = 反转	0
保留	[3:0]	保留	0x0

11.1 LCD 控制器原理概述

- VIDTCON0,R/W, ADDRESS=0xEE000010

域	位	描　　述	复位值
VBPD	[23:16]	垂直后肩所占周期	0x00
VFPD	[15:8]	垂直前肩所占周期	0x00
VSPW	[7:0]	垂直同步脉冲宽度	0x00

- VIDTCON1,R/W,ADDRESS=0xEE000014

域	位	描　　述	复位值
HBPD	[23:16]	水平后肩所占周期	0x00
HFPD	[15:8]	水平前肩所占周期	0x00
HSPW	[7:0]	水平同步脉冲宽度	0x00

- VIDTCON2,R/W,ADDRESS=0xEE000018

域	位	描　　述	复位值
LINEVAL	[21:11]	该位决定了显示屏的垂直尺寸	0
HOZVAL	[10:0]	该位决定了显示屏的水平尺寸	0

11.1 LCD 控制器原理概述

VIDTCON2, R/W

ADDRESS=0xEE000020

域	位	描　　述	复位值
ENLOCAL	[22]	数据存取路径. 0 = DMA 方式 1 = 本地方式（CAMIF 0 ）	0
BUFSTATUS	[21]	缓冲区的编号（这是因为 windows 0,1 可以有两个缓冲区） (Read Only) 0 = 缓冲区 0　　1 = 缓冲区 1	0
BUFSEL	[20]	选择缓冲区(0/1) 0 = 缓冲区 0　　1 = 缓冲区 1	0
BUFAUTOEN	[19]	双缓冲自动控制位 0 = BUFSEL, 1 = 自动改变由 Trigger Input 控制	0
位 SWP	[18]	位交换控制位 0 = 禁止交换　　1 = 使能交换	0
BYTSWP	[17]	字节交换控制位 0 = 禁止交换　　1 = 使能交换	0
HAWSWP	[16]	半字交换控制位 0 = 禁止交换　　1 = 使能交换	0
WSWP	[15]	字交换控制位 0 = 禁止交换　　1 = 使能交换	0
保留	[14]	保留	0
InRGB	[13]	输入的源图像的格式 (Only for 'EnLcal' enable) 0 = RGB 1 = YCbCr	0
保留	[12:11]	保留 (should be 00)	0
BURSTLEN	[10:9]	DMA's Burst 最大长度 00 = 16 字 01 = 8 字 10 = 4 字	0
保留	[8:7]	保留	0

11.1 LCD 控制器原理概述

VIDTCON2,R/W,

ADDRESS=0xEE000020

BLD_PIX	[6]	选择绑定的类型 0 = 平面绑定 1= 像素绑定	0
BPPMODE_F	[5:2]	BPP (位 s Per Pixel)模式 0000 = 1 bpp 0001 = 2 bpp 0010 = 4 bpp 0011 = 8 bpp (palletized) 0100 = 8 bpp (无调色盘, A: 1-R:2-G:3-B:2) 0101 = 16 bpp (无调色盘, R:5-G:6-B:5) 0110 = 16 bpp (无调色盘, A:1-R:5-G:5-B:5) 0111 = 16 bpp (无调色盘, I :1-R:5-G:5-B:5) 1000 = unpacked 18 bpp (无调色盘, R:6-G:6-B:6) 1001 = unpacked 18 bpp (无调色盘, A:1-R:6-G:6-B:5) 1010 = unpacked 19 bpp (无调色盘, A:1-R:6-G:6-B:6) 1011 = unpacked 24 bpp (无调色盘, R:8-G:8-B:8) 1100 = unpacked 24 bpp (无调色盘,A:1-R:8-G:8-B:7) 1101 = unpacked 25 bpp (无调色盘, A:1-R:8-G:8-B:8) 1110 = unpacked 13 bpp (无调色盘,A:1-R:4-G:4-B:4) 1111 = unpacked 15 bpp (无调色盘, R:5-G:5-B:5)	
ALPHA_SEL	[1]	选择 alpha 值的方式 平面绑定时： 0 = using ALPHA0_R/G/B values 1 = using ALPHA1_R/G/B values 像素绑定时： 0 = AEN 使能位置 （细节请参看 S5PC100 手册）	
ENWIN_F	[0]	0 = 窗口禁止 1 = 窗口使能	

11.1 LCD 控制器原理概述

FRAME BUFFER ADDRESS 0,R/W,ADDRESS=0xEE0000A0

域	位	描　　述	复位值
VBANK_F	[31:24]	决定系统内存中的 bank 地址	0
VBASEU_F	[23:0]	决定 frame buffer 的起始地址	0

FRAME BUFFER ADDRESS 1,R/W,ADDRESS=0xEE0000D0

域	位	描　　述	复位值
VBASEL_F	[23:0]	决定了 frame buffer 的结束地址	0x0

FRAME BUFFER ADDRESS 2 ,R/W,ADDRESS=0xEE000100

域	位	描　　述	复位值
OFFSIZE_F	[25:13]	虚拟屏幕偏移（byte）	0
PAGEWIDTH_F	[12:0]	虚拟页宽度（byte)	0

11.1 LCD 控制器原理概述

WINDOW 1 BLENDING EQUATION CONTROL REGISTER ,R/W,ADDRESS=0xEE000244

域	位	描　　述	复位值
保留	[31:22]	保留	0x000
Q_FUNC	[21:18]	alphaB 值为 0000 = 0 (zero) 0001 = 1 (max) 0010 = **alphaA (alpha value of *foreground) 0011 = 1 – alphaA 0100 = alphaB 0101 = 1 – alphaB 011x = 保留 100x = 保留 1010 = *A* (foreground color data) 1011 = 1 – *A* 1100 = *B* (background color data) 1101 = 1 – *B* 111x = 保留	0x0
保留	[17:16]	保留	00
P_FUNC	[15:12]	alpha 值为 同上	0x0
保留	[11:10]	保留	00
B_FUNC	[9:6]	B 的值为 同上	0x3
保留	[5:4]	保留	00
A_FUNC	[3:0]	A 的值为 同上	0x2

11.2 LCD 控制器应用示例

编写一个驱动例子，将一张分辨率为 480 ×272，颜色深度为 16 位图片显示在 LCD 屏，请结合下面的流程图（图 17-4）及代 码模块，深入理解 LCD 驱动流程

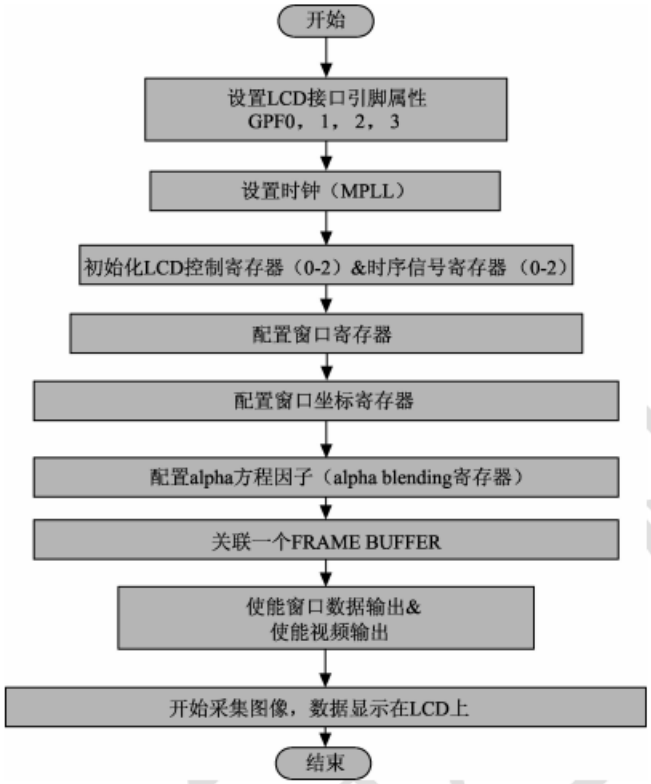


图 17-4　LCD 控制流程图

11.2 LCD 控制器应用示例

（1）下面代码，主要是一对 LCD 屏幕物理参数的设置、时序的配置，还有基本 寄存器的配置。

/*这个结构体包含了 LCD 屏幕的物理参数，如时序信号的极性 * 屏幕像素大小，颜色深度，刷新频率 */

```
static struct LcdPhyInfo innolux430 = {
  .width = 480,
  .height = 272,
  .bpp = 16,
  .freq = 60,

  .timing = {
      .h_fp = 2,
      .h_bp = 2,
      .h_sw = 41,
      .v_fp = 2,
      .v_fpe = 1,
      .v_bp = 2,
      .v_bpe = 1,
      .v_sw = 10,
  },

  .polarity = {
      .rise_vclk = 0,
      .inv_hsync = 1,
      .inv_vsync = 1,
      .inv_vden = 0,
  },
};
```

11.2 LCD 控制器应用示例

继续

```
struct window win =
{
  .RgbMode = 0x5,              //RGB 模式;
  .DataComeFrom=_DMA,          //使用 DMA 方式获取数据
  .BLD_PIX=0,                 //使用数据混合模式
  .ALPHA_SEL=0,               //使用灰度级模式 alpha0_R/G/B
  .OSD_LEFTOPX  =0,
  .OSD_LEFTOPY =0,
  .OSD_RIGHTBOTX = 480,
  .OSD_RIGHTBOTY = 272,
  .OSDSIZE = 480*272,
};
```

11.2 LCD 控制器

继续

```
/*下面这个结构体中的配置,请读者对照前面给的那个 alpha 绑定公式并务必理解其含义*/
struct AlphaEquationFactor AlphaEquGlobal[]=
{
  [0] = {
            .winnum = 0,
            .q  = 0x0,
            .p  = 0x0,
            .b=0x3,
            .a=0x2,
       },
  [1] = {
            .winnum = 1,
            .q  = 0x0,
            .p  = 0x0,
            .b  = 0x3,
            .a  = 0x2,
       },
  [2] = {
            .winnum = 2,
            .q  = 0x0,
            .p  = 0x0,
            .b  = 0x3,
            .a  = 0x2,
       },
  [3] = {
            .winnum = 3,
            .q  = 0x0,
            .p  = 0x0,
            .b  = 0x3,
            .a  = 0x2,
       },

};                          //它包含了 4 个窗口的灰度混合方程的因子
/*该结构体使用贯穿全部代码*/
```

11.2 LCD 控制器应用示例

继续

```
struct mylcd Lcd =
{
  .Phyinfo = &innolux430,
  .wcount  =1,                              //只使能 WIN0;
  .win[0] = &win,
  .AlphaEqu = AlphaEquGlobal,
  .fb = &fbaddr,
  .hoz = 480,
  .line = 272,
}
```

11.2 LCD 控制器应用示例

（2）首先是配置 VIDEO MAIN （主）和 VIDEO TIME （时序）寄存器。

```
int InitGobalReg(struct mylcd *lcd)
{
  unsigned int cfg;
  struct LcdPhyInfo *phyinfo = lcd->Phyinfo;
  cfg =0;
  cfg |= S3C_VIDCON0_CLKDIR_DIVIDED;    //选择时钟源路径
  cfg |= S3C_VIDCON0_CLKVAL_F(15);      //设置时钟
  writel(cfg, LCDBASEADDR + S3C_VIDCON0);
  /*下列所涉及的物理时序值需要参考所使用的 LCD 屏的出厂值，可参考对应的手册。*/
  cfg = 0;                              //下面对 vclk、hsync、vsync、vden 的时
钟极性进行设置
  if (phyinfo->polarity.rise_vclk)
      cfg |= S3C_VIDCON1_IVCLK_RISING_EDGE;
  if (phyinfo->polarity.inv_hsync)
      cfg |= S3C_VIDCON1_IHSYNC_INVERT;
  if (phyinfo->polarity.inv_vsync)
      cfg |= S3C_VIDCON1_IVSYNC_INVERT;
  if (phyinfo->polarity.inv_vden)
      cfg |= S3C_VIDCON1_IVDEN_INVERT;
  writel(cfg, LCDBASEADDR + S3C_VIDCON1);
```

11.2 LCD 控制器应用示例

继续

```
//下面对行列同步宽度及前肩后肩进行设置
  cfg = 0;
  cfg |= S3C_VIDTCON0_VBPDE(phyinfo->timing.v_bpe-1);
  cfg |= S3C_VIDTCON0_VBPD(phyinfo->timing.v_bp-1);
  cfg |= S3C_VIDTCON0_VFPD(phyinfo->timing.v_fp-1);
  cfg |= S3C_VIDTCON0_VSPW(phyinfo->timing.v_sw-1);
  writel(cfg, LCDBASEADDR + S3C_VIDTCON0);
  cfg = 0;
  cfg |= S3C_VIDTCON1_VFPDE(phyinfo->timing.v_fpe-1);
  cfg |= S3C_VIDTCON1_HBPD(phyinfo->timing.h_bp-1);
  cfg |= S3C_VIDTCON1_HFPD(phyinfo->timing.h_fp-1);
  cfg |= S3C_VIDTCON1_HSPW(phyinfo->timing.h_sw-1);
  writel(cfg, LCDBASEADDR + S3C_VIDTCON1);
  cfg =0;
  cfg |= S3C_VIDTCON2_HOZVAL(lcd->hoz-1);
  cfg |= S3C_VIDTCON2_LINEVAL(lcd->line-1);
  writel(cfg, LCDBASEADDR + S3C_VIDTCON2);
  return 0;
}
```

11.2 LCD 控制器应用示例

（3）设置 win 控制寄存器。

```
static int window_control(struct mylcd *lcd,int whichwin)
{
  unsigned int cfg;
  cfg = 0;
  cfg |=S3C_WINCON_HAWSWP_ENABLE;
  cfg |=((lcd->win[whichwin]->RgbMode)<<2); //设置 RGB565
  writel(cfg, LCDBASEADDR+ S3C_WINCON(whichwin));
  return 0;
}
```

（4）设置 win position 寄存器。

```
static void SetWinpos(struct window *win,int inum)
{
   unsigned int cfg;
   cfg=(win->OSD_LEFTOPX<<11) |(win->OSD_LEFTOPY<<0);
   writel(cfg,LCDBASEADDR+S3C_VIDOSD_A(inum));
writel((win->OSD_RIGHTBOTX<<11)|(win->OSD_RIGHTBOTY<<0),LCDBASEADDR+S3C_V
IDOSD_B(inum));
  writel(win->OSDSIZE,LCDBASEADDR+S3C_VIDOSD_C(inum));
}
```

11.2 LCD 控制器应用示例

（5）配置 alpha 绑定函数的因子。

```
void ConfigAlphaEquFactor(struct mylcd *lcd)
{
  struct AlphaEquationFactor  *AlphaEqu = lcd->AlphaEqu;
  writel(set_factor(&AlphaEqu[0]),S3C_BLENDEQ1 + LCDBASEADDR);
  writel(set_factor(&AlphaEqu[1]),S3C_BLENDEQ2 + LCDBASEADDR);
  writel(set_factor(&AlphaEqu[2]),S3C_BLENDEQ3 + LCDBASEADDR);
  writel(set_factor(&AlphaEqu[3]),S3C_BLENDEQ4 + LCDBASEADDR);
}
```

11.2 LCD 控制器应用示例

（6）调试与实验结果。

作为测试的图片，可以先使用类似于 Image2Lcd 这样的软件，将一张 16 位的位 图转换为 480×272×2 的数组，将其关联到 Frame Buffer 地址。

然后，编译生成的.elf 文件，硬件接线。并连接好 FS_JTAG 仿真器套件。

将程 序编译后获得.elf 文件，将该文件通过仿真器下载并运行在目标板上，这时图片就能显示在 LCD 的显示屏幕上了。

编程思路和步骤

一：初始编程

（一）.建立重要结构体

一般设置LCD控制器，只需要直接写相应的寄存器就可以了，但是为了简化程序，结构体在这里必不可少，这个结构体包含了设置LCD的重要参数：

1.引脚极性：可能HSYNC(行同步信号)在外接的LCD是高电平有效，但是芯片的控制器的HSYNC是低电平有效，这时就需要把HSYNC的引脚极性设置为反转，以匹配外部控制的需要。

2.时序：我们需要把外界LCD的时序告诉LCD控制器，通过参数，设置LCD控制器的时序。

3.分辨率，以及bpp(每个像素占多少位，以及在framBuffer存放的格式)。

4.frambuffer的地址，在内存可以划分出一块内存区域制定为frambuffer,当需要向LCD写数据时，只需往这块内存区域写数据，LCD控制器就会自动把颜色数据发送到LCD，然后显示在外接的LCD上。

编程思路和步骤

（二）.创建结构体变量，并赋值（以下的名字大部分都是举例）

在上一步中只是创建了一个包含了lcd各种参数的结构体，并没有创建结构体变量，现在需要创建一个lcd_params类型的结构体变量，创建的变量名为lcd_4_3_params，这个变量包含了共5项，需要对这些成员变量赋值，以便后面设置LCD控制器时，直接读取结构体变量lcd_4_3_params的值就OK了。

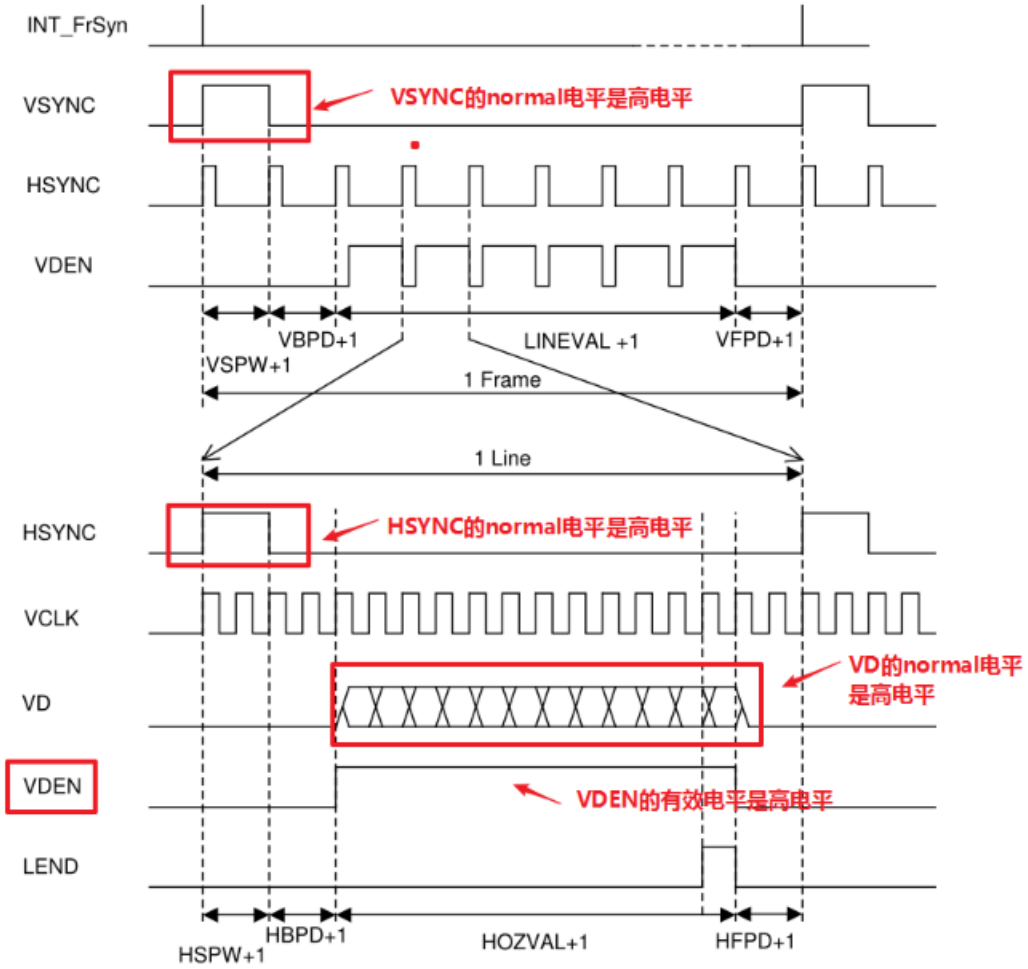
第一项：name，此成员变量起识别作用，可随意取值

第二项：pins_pol这个成员变量是pins_polarity类型的，它是一个结构体，保存了有关极性引脚。

之所以需要设置极性的引脚，是因为芯片的LCD控制器是可编程的，通过设置可以符合很多不同外接的LCD屏幕，但是为了确保LCD控制器能够控制外接的LCD，那就需要确保所发出的有效电平（比如低电平），和外接LCD所接的有效电平（必须也为低电平）是一致的。因此芯片专门有这样一个寄存器来设置这些默认极性是否需要反转，此寄存器为VIDCON1.

编程思路和步骤

通过查看芯片的LCD控制器的时序图，下边所标的电平就是LCD控制器所认为的normal 极性

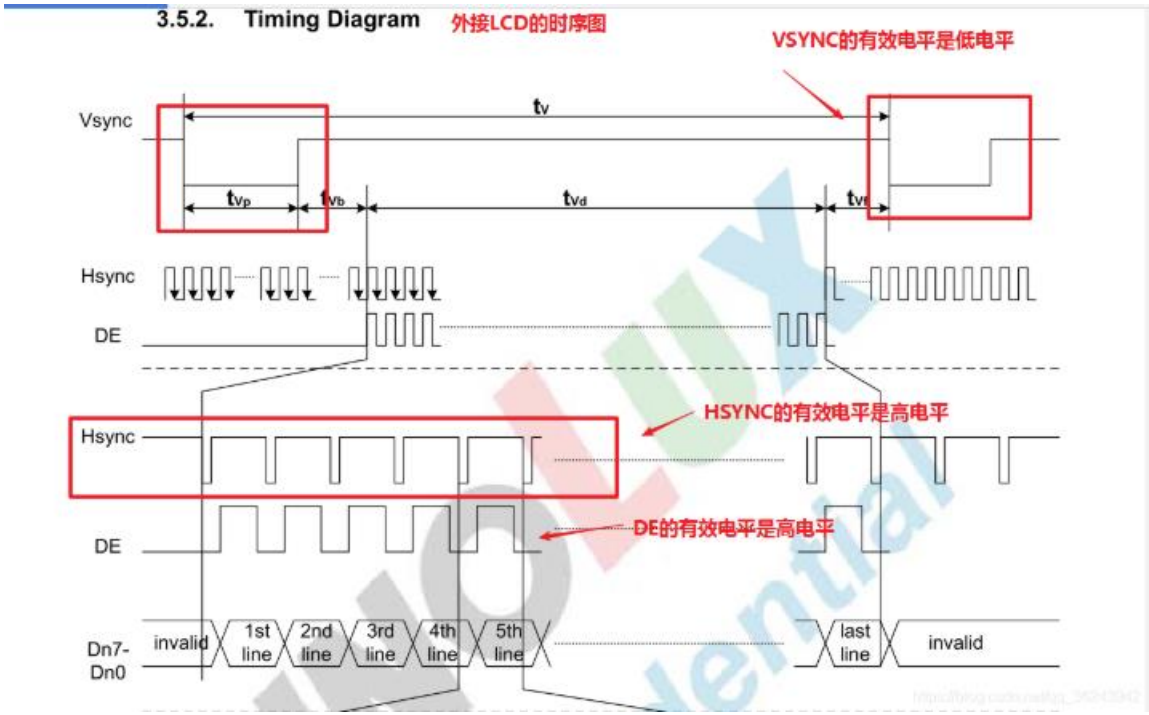


编程思路和步骤

接着看看外接的LCD屏幕的时序图，看看它需要的是什么时序：

可以看出外接LCD的VSYNC的有效电平是低电平，所以说LCD控制器的默认normal电平并不能满足外接时序的需要，所以需要设置LCD控制器的VSYNC的极性为反转极性，这样原本高电平为有效电平，现在变成低电平。

其他引脚极性都是符合外接LCD的，所以直接采用默认的值就可以了。



编程思路和步骤

（三）：time_seq这个成员变量是time_sequence结构体类型的变量，其保存了有关于时序的变量。

外接LCD芯片给出了这些时序的时间值，肯定就能找到一张表，它会给出一些这些时间段的最大/最小值，有些会给出建议设置的值，直接按照给出的值进行设置就OK了。

外接的LCD时序中有几个重要的参数是：

```
/* 垂直方向 */
tvp;     /* vysnc脉冲宽度 */
tvb;     /* 上边黑框, Vertical Back porch */
tvf;     /* 下边黑框, Vertical Front porch */
/* 水平方向 */
thp;    /* hsync脉冲宽度 */
thb;    /* 左边黑框, Horizontal Back porch */
thf;    /* 右边黑框, Horizontal Front porch */
/*像素时钟频率*/
vclk;
```

编程思路和步骤

（四）：分辨率和bpp

这个LCD型号是AT043TN24,查找相关参数得知分辨率大小为：480*272

bpp的含义是：每一个像素所占的位数，这款LCD控制器支持三种格式：
8bpp,16bpp,24bpp,4bpp,2bpp

而使用不同的bpp,肯定是会对颜色的数据大小产生影响，16bpp比8bpp所占的内存空间大，那么这些颜色在frambuffer存放的内存就需要多一些，而且设置不同的bpp，在frambuffer存放格式是可以自己来设置的。

编程思路和步骤

（五）：是frambuffer的起始地址

在这里指定frambuffer的地址，当设置完毕，LCD控制器会配合时序通过LCDDMA直接从frambuffer取颜色数据（数据的bpp和数据存放格式需要字节设置），发送到VD数据线上（其实这VD数据线就像是能移动并且发出各种颜色的电子枪），最后从屏幕中显示出来。

当我们需要去改变颜色时，只需要改变frambuffer的数据，在相应区域存入不同的颜色数据，LCDDMA自动从frambuffer取数据，显示在LCD屏幕上就OK了。

编程思路和步骤

注意：选定的这款frambuffer区域应该是未使用的，否则，当我们一写好数据颜色，马上就会被改变了。就达不到我们想要显示的效果了。

备注：这块frambuffer的区域的大小是可以计算的，但是前提是需要先知道bpp和数据存放格式，以及分辨率

这些在重要参数结构体中可以计算，假设

bpp等于16，分辨率为：480*272 = 131648 像素点
那么每个像素点占用16bit，也就是两个字节。
这块frambuffer所占的字节数是：131648*2字节=263296 字节。

假如设置frambuffer的起始地址是：0x33a00000 那么frambuffer的区域就是从：0x33a00000 - 33a40480

如果为24bpp的话占用的内存会增加。

编程思路和步骤

二：初始化LCD使用到的引脚

查看电路图

1.背光使用到了GPB0,设置为输出，高电平为使能

2.使用到了数据线VD，VD3- VD7,VD10-VD15,VD19-VD23 ，共16条，应配置为LCD模式

三：设置LCD控制器

为了程序看起来简单和扩展性，把LCD参数的设置打包成一个函数，这个函数有一个参数是lcd_params的指针类型，根据需要设置的具体LCD参数，使用对应的结构体变量对LCD控制器进行设置，例如：函数如下：

```
void s3c2440_lcd_controller_init(p_lcd_params plcdparams)
```

这里设置所有必要的寄存器：LCDCON1 - LCDCON5 ,LCDSADDR1 - LCDSADDR2

编程思路和步骤

四.使能和禁用LCD控制器函数

五.从结构体变量获取LCD重要参数

六.写测试函数

有了上面的准备工作，从现在开始就显示简单一点了,刚开始写测试代码，为了显得简单一点，让整个屏幕显示出同一种颜色、

如何做呢？直接往frambuffer填入颜色数值就行了。且设置bpp = 16，那么一个颜色的数据就占用两个字节。

小结

- 本章主要介绍了基于S5PC100 的LCD控制器，操作TFT LCD的方法。
- 通过一个实例实现了在LCD屏上显示一幅图片。

思考与练习

11-1 TFT液晶显示屏外部接口信号有哪些？

11-2 简述VFRAME、VLINE、VCLK这几个信号的作用？

11-3 编程实现在LCD上显示一幅你自己的图片

谢谢！