个人资料

xubin341719

访问： 537044次
积分： 5973
等级： BLOG 6
排名： 第1561名

原创： 69篇 转载： 3篇
译文： 0篇 评论： 402条

Android LCD(四)：LCD驱动调试篇

分类： S5PXX(三星) linux android 嵌入式入门 2013-06-26 18:18 10386人阅读 评论(10) 收藏 举报

关键词：**android LCD TFTSN75LVDS83B TTL-LVDS LCD**电压背光电压

平台信息：

内核：**linux2.6/linux3.0**

系统：**android/android4.0**

平台：**samsung exynos 4210**、**exynos 4412** 、**exynos 5250**

作者：**xubin341719**(欢迎转载，请注明作者)

欢迎指正错误，共同学习、共同进步！！

下载链接：**LCD**规格书（**404**份），之前工作用用到的 、 **LCD**规格书**00** 、 **LCD**规格书**01** 、 **LCD**测试图片，彩条灰阶等

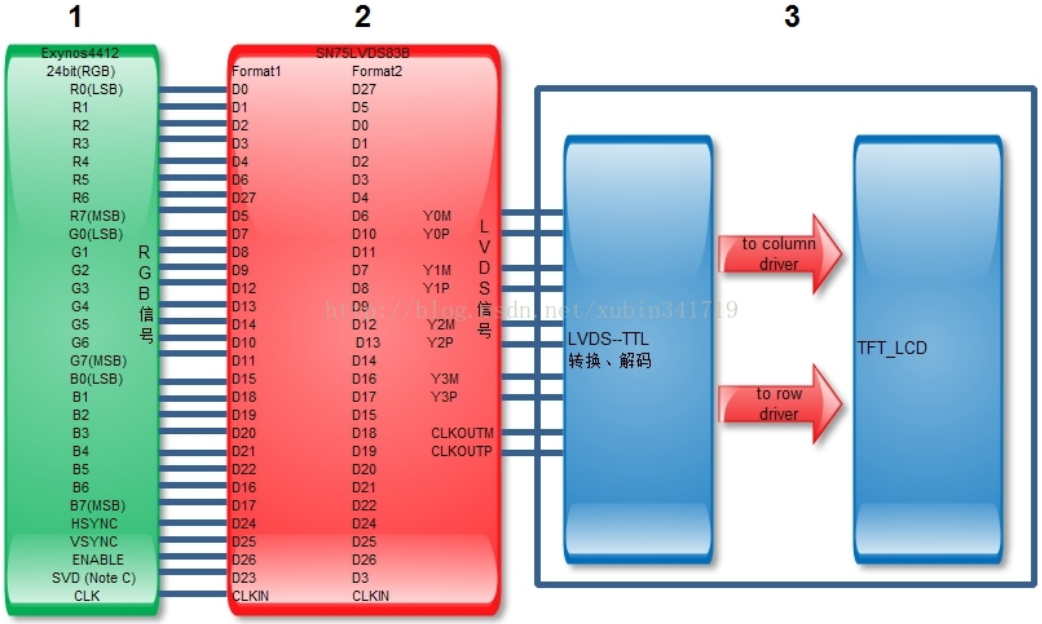
Android LCD(一)：LCD基本原理篇
Android LCD(二)：LCD常用接口原理篇
Android LCD(三)：Samsung LCD接口篇
Android LCD(四)：LCD驱动调试篇

这篇我们以一个实例来说明,Samsung Exynos4412搭配TTL转LVDS芯片SN75LVDS83B、LVDS接口LCD为例说明。从硬件接口、驱动配置、背光PWM调节三部分说明。

下载：SN75LVDS83B规格书

一、**LCD**接口原理以及硬件电路

Samsung Exynos4412、SN75LVDS83B、LVDS接口LCD（24bit）为例说明，三者的关系如下：



如上图所示，我们在应用中我，主控（Exynos4412）输出RGB信号到TFT-LCD大体经过三部分：

2012年06月 (11)
2012年04月 (1)
2012年03月 (1)
2012年02月 (2)
2011年12月 (12)
2011年11月 (2)

阅读排行

android 电容屏（一）：
(28792)
android 电池（二）：an
(24779)
android camera(二)：摄
(24660)
android 电容屏（三）：
(22132)
android camera(四)：ca
(18475)
android HDMI(一)：HDI
(18184)
android camera(一)：ca
(17204)
Android 4.0 虚拟按键、
(16460)
android camera(三)：ca
(15784)
android 电容屏（二）：
(15588)

评论排行

android 电容屏（三）：(66)
android camera(四)：ca (45)
android HDMI(一)：HDI (21)
android camera(二)：摄 (20)
android 电池（三）：an (19)
android 电池（二）：an (16)
android电池（四）：电 (13)
蓝牙核心技术概述（四） (12)
android camera(三)：ca (12)
Android LCD(一)：LCD (11)

推荐文章

最新评论

Android BlueDroid（三）：Blue
xuexingyang: 说好的下一节呢。。。。都在等呢。。。。楼主加油啊～～～

蓝牙核心技术概述（三）：蓝牙
as453866908: 博主写的非常棒！ 帮我解决了很多疑惑，但是我这里用蓝牙有个问题就是蓝牙复位后我设置了使能Simple...

Android BlueDroid（一）：Blue
auxor: 我想在把Android手机变成蓝牙耳机设备，这个在技术上是否可行？我理解只要在手机上实现了Heads...

Android BlueDroid（一）：Blue
auxor: 我想在把Android手机变成蓝牙耳机设备，这个在技术上是否可行？我理解只要在手机上实现了Heads...

Android BlueDroid（一）：Blue
auxor: 看到了你写的Bluedroid概述文章，特来请教几个问题。我想在把Android手机变成蓝牙耳机设备...

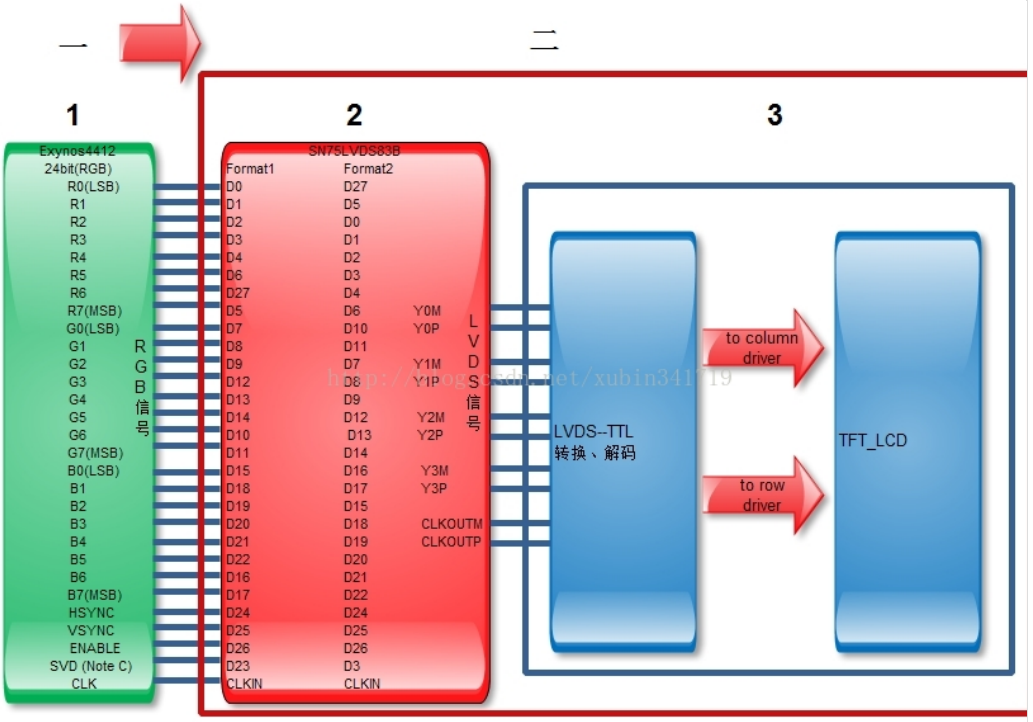
android 电容屏（三）：驱动调
sunnyliang1: 楼主很棒！感谢！

Android bluetooth介绍（三）：
wi100sh: 多谢分享～

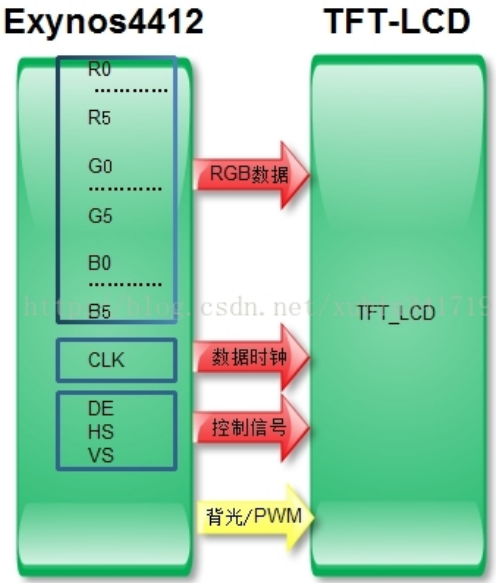
android camera(一)：camera模
u010423298: 楼主，求推荐相关

（1）、标号1部分，主控（Exynos4412）输出TTL信号；
（2）、标号2部分，TTL（RGB）-LVDS转换芯片SN75LVDS83B,把ＴＴＬ信号转换成LVDS信号，传输到显示器的LVDS接收端；这部分有SN75LVDS83B编码芯片自动完成，所以我们不需要程序控制；
（3）、标号3部分，分两个小部分，LVDS转换成TTL，TFT-LCD显示部分；我们前面说过，TFT-LCD其实只识别TTL信号，所以要有一个转换的过程，先把LVDS信号转换、解码成TTL信号，在TFT-LCD上显示。

有上面的过程，其实我们关心调试的部分只有标号1部分到标号2部分，后面标号2到标号3的部分是自动完成的，不需要我们程序上控制，把标号2部分、标号3部分合并：



标号二部分可以理解为一个TTL（RGB）接口的LCD,如下图所示，标号一部分就是主控信号输出端，简化图如下所示:

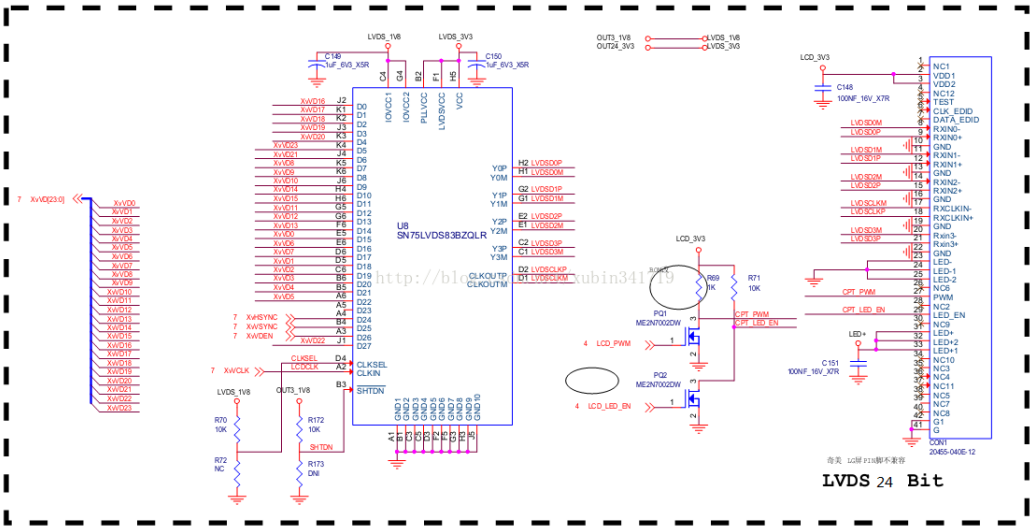


其实最简单的做法就是找个TTL接口的TFT-LCD，这样直接接上就可以。下面我们看下硬件上的电路连接：这个和我们上篇用的相同。

书籍，论坛，谢谢啦，网上一搜都是卖相机的

Android LCD(一)：LCD基本原理 hexiaolong2009: 博主以前做FAE的？看你的文章对器件的原理了解的十分透彻啊！多谢分享！

Android bluetooth介绍（一）：基 hustljh: 请教博主，对于uart传输的数据，经过HCI层时有ACL和SCO两种数据格式，分别什么时候用？播放m...



有上面图可以看出：硬件连接

网络标号	说明	管脚
XvVD[0：23] XvVDEN XvVSYNC XvHSYNC XvVCLK	RGB数据、使能、行场同步、时钟信号	这是TTL信号输出
LCD_PWM	调节背光	XpwmTOUT1/LCD_PWM/GPD0_1
LCD_LED_EN	LCD电压（TFT电压）使能	GPC1_2
LED_BL_EN	LED背光使能	GPL2_4

上面可分为几部分，电路连接部分分析：

（**1**）、**TTL**数据部分

这张图有木有烂掉呀，哈哈，就是这些数据了。还有有木有想起来摄像头的数据（ITU接口）也是这样的？？其实视频这种信号的原理是通用的，所以LCD通了，摄像头也就知道怎么回事了。

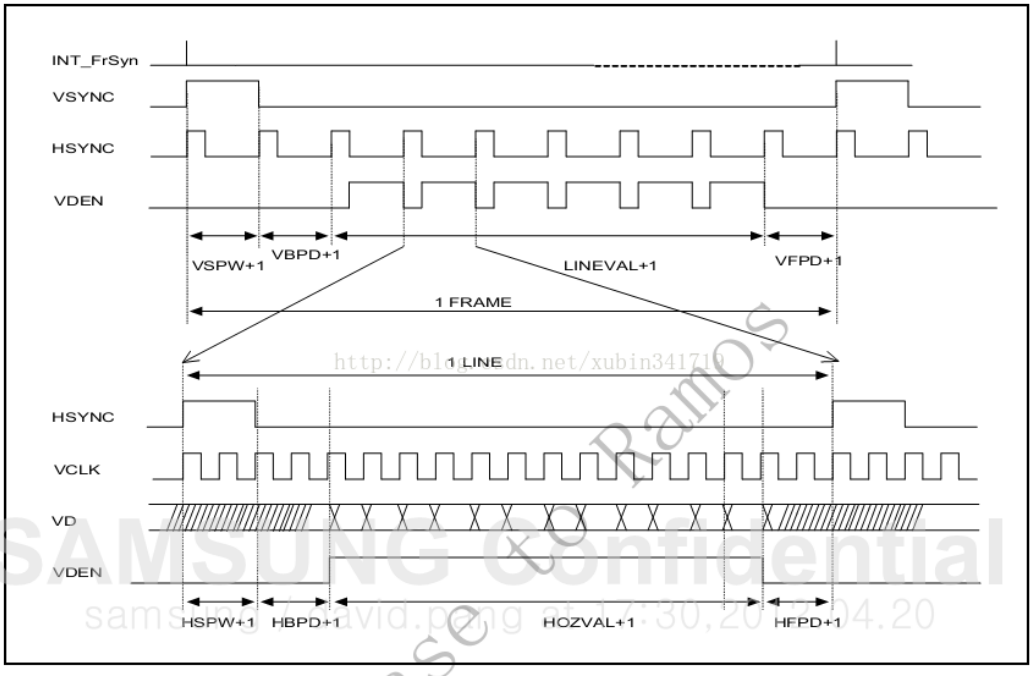
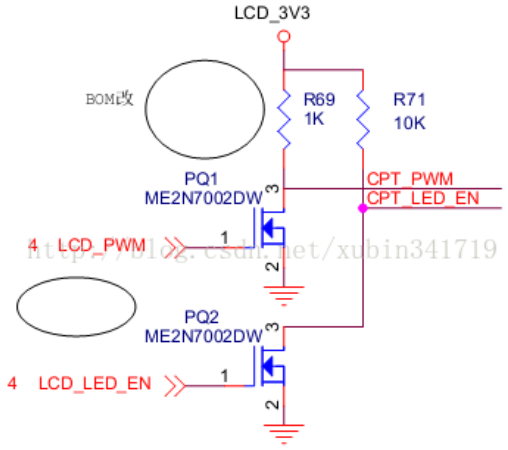


Figure 41-39 LCD RGB Interface Timing

（2）、**PWM**背光调节

PWM其实也是芯片的一个功能模块，看到他的管脚就是一个复用脚XpwmTOUT1/LCD_PWM/GPD0_1。上一篇我们粗略的了解了PWM，就是用到这里。但是有一个疑问，PWM是调节背光电压的，背光电压一般都是12V以上的，我们PWM只有0-3V的样子，Exynos4412的IO只有1.8V。怎么调节电压？？？

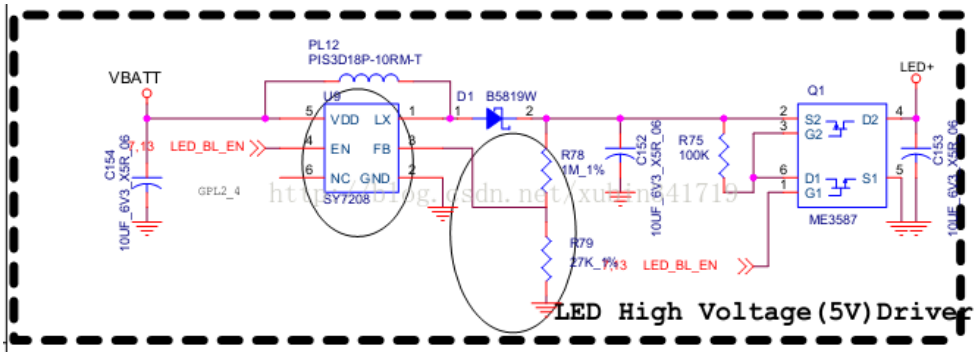


其实这个PWM只是给LCD上PWM控制部分，真正的电压还是通过LCD控制板上的电路实现。

（3）、**LED**背光、**LCD**电压控制

a、背光：**LED+**

我们可以看到这个升压电路，通过SY7208把VBATT升压到18V，供给LED背光。SY7208最大升压26V。这个电压是提供给我们前面讲的背光的，也就是CCFL灯管或者LED背光组的电压。



b、**LCD**电压

这个电压也就是给你我们TFT阵列组用的，控制LCD液晶元素。

这部分电路分析完成，我们就有比较清晰的思路出，要一个LCD工作，要完成两部分内容：LCD上电控制，背光、LCD电压；信号输出。

二、**LCD** 驱动部分调试

LCD这部分，像上篇我们说的frambuffer这些部分一般平台都是可以用的，除非你是芯片厂的要写这部分。

一般公司拿到的demo板子这部分都是通的，只是针对自己的ICD换一些参数。

下面我们针对三星平台我们调试LCD的时时候程序方面的改动：

1、屏参数的配置

/kernel/drivers/video/Samsung/s3cfb_wa101s.c

```
static struct s3cfb_lcd wa101 = {

        .width  = 1280,//LCD 分辨率宽1280
        .height = 800, //LCD 分辨率高 800
        .bpp    = 24,//CLD 数据位 24bit
        .freq   = 60,//LCD 像素时钟 60MHz
        .timing = {//LCD porch无效值
                .h_fp   = 70,
                .h_bp   = 70,
                .h_sw   = 20,

                .v_fp   = 10,
                .v_fpe  = 0,
                .v_bp   = 10,
                .v_bpe  = 0,
                .v_sw   = 3,
        },

        .polarity = {//时钟、行场的极性;
                .rise_vclk      = 1,
                .inv_hsync      = 1,
                .inv_vsync      = 1,
                .inv_vden       = 0,
        },
};
/* name should be fixed as 's3cfb_set_lcd_info' */
void s3cfb_set_lcd_info(struct s3cfb_global *ctrl)//初始化结构体
{
        wa101.init_ldi = NULL;
        ctrl->lcd = &wa101;
#endif
}
```

还能想起上一篇的如何阅读规格书中的那些参数不，把这些填入就可以。

2、数据管脚初始化

kernel/arch/arm/mach-exynos/setup-fb-s5p.c

```
void s3cfb_cfg_gpio(struct platform_device *pdev)
{
        s3cfb_gpio_setup_24bpp(EXYNOS4_GPF0(0), 8, S3C_GPIO_SFN(2), S5P_GPIO_DRVSTR_LV4);
        s3cfb_gpio_setup_24bpp(EXYNOS4_GPF1(0), 8, S3C_GPIO_SFN(2), S5P_GPIO_DRVSTR_LV4);
        s3cfb_gpio_setup_24bpp(EXYNOS4_GPF2(0), 8, S3C_GPIO_SFN(2), S5P_GPIO_DRVSTR_LV4);
        s3cfb_gpio_setup_24bpp(EXYNOS4_GPF3(0), 4, S3C_GPIO_SFN(2), S5P_GPIO_DRVSTR_LV4);
}
```

LCD 数据脚初始化，驱动能力设为最高S5P_GPIO_DRVSTR_LV4；管脚驱动能力，S5P_GPIO_DRVSTR_LV1-4四个等级选择。

3、时钟控制部分

kernel/arch/arm/mach-exynos/setup-fb-s5p.c

```
int s3cfb_clk_on(struct platform_device *pdev, struct clk **s3cfb_clk)
{
        struct clk *sclk = NULL;
        struct clk *mout_mpll = NULL;
        struct clk *lcd_clk = NULL;

        u32 rate = 0;
        int ret = 0;

        lcd_clk = clk_get(&pdev->dev, "lcd");
        if (IS_ERR(lcd_clk)) {
                dev_err(&pdev->dev, "failed to get operation clk for fimd\n");
                goto err_clk0;
        }

        ret = clk_enable(lcd_clk);
        if (ret < 0) {
                dev_err(&pdev->dev, "failed to clk_enable of lcd clk for fimd\n");
                goto err_clk0;
        }
        clk_put(lcd_clk);

        sclk = clk_get(&pdev->dev, "sclk_fimd");
        if (IS_ERR(sclk)) {
```

```
                dev_err(&pdev->dev, "failed to get sclk for fimd\n");
                goto err_clk1;
        }

        if (soc_is_exynos4210())
                mout_mpll = clk_get(&pdev->dev, "mout_mpll");
        else
                mout_mpll = clk_get(&pdev->dev, "mout_mpll_user");

        if (IS_ERR(mout_mpll)) {
                dev_err(&pdev->dev, "failed to get mout_mpll for fimd\n");
                goto err_clk2;
        }

        ret = clk_set_parent(sclk, mout_mpll);
        if (ret < 0) {
                dev_err(&pdev->dev, "failed to clk_set_parent for fimd\n");
                goto err_clk2;
        }

        if ((soc_is_exynos4412()) && (samsung_rev() >= EXYNOS4412_REV_2_0))
                ret = clk_set_rate(sclk, 880000000);
        else
                ret = clk_set_rate(sclk, 800000000);
        if (ret < 0) {
                dev_err(&pdev->dev, "failed to clk_set_rate of sclk for fimd\n");
                goto err_clk2;
        }
        dev_dbg(&pdev->dev, "set fimd sclk rate to %d\n", rate);

        clk_put(mout_mpll);

        ret = clk_enable(sclk);
        if (ret < 0) {
                dev_err(&pdev->dev, "failed to clk_enable of sclk for fimd\n");
                goto err_clk2;
        }

        *s3cfb_clk = sclk;

        return 0;

err_clk2:
        clk_put(mout_mpll);
err_clk1:
        clk_put(sclk);
err_clk0:
        clk_put(lcd_clk);

        return -EINVAL;
}
int s3cfb_clk_off(struct platform_device *pdev, struct clk **clk)
{
        struct clk *lcd_clk = NULL;
```

```
        lcd_clk = clk_get(&pdev->dev, "lcd");
        if (IS_ERR(lcd_clk)) {
                printk(KERN_ERR "failed to get ip clk for fimd0\n");
                goto err_clk0;
        }

        clk_disable(lcd_clk);
        clk_put(lcd_clk);

        clk_disable(*clk);
        clk_put(*clk);

        *clk = NULL;

        return 0;

err_clk0:
        clk_put(lcd_clk);

        return -EINVAL;
}

void s3cfb_get_clk_name(char *clk_name)
{
        strcpy(clk_name, "sclk_fimd");
}
```

4、背光、**LCD**电压的控制

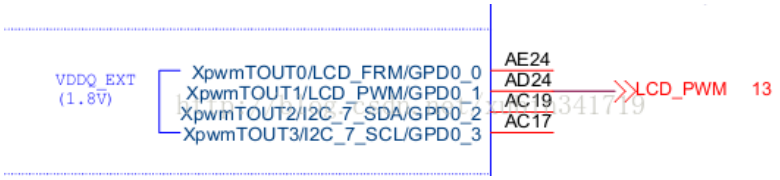
LCD_LED_EN	LCD电压（TFT电压）使能	GPC1_2
LED_BL_EN	LED背光使能	GPL2_4

```
int s3cfb_backlight_on(struct platform_device *pdev)
{
        int err;
        pwm_set();
        err = gpio_request_one(EXYNOS4_GPL2(4), GPIOF_OUT_INIT_HIGH, "GPL2_4");
        if (err) {
                printk(KERN_ERR "failed to request GPL2 for "
                        "lcd backlight control\n");
                return err;
        }
        s3c_gpio_setpull(EXYNOS4_GPL2(4),S3C_GPIO_PULL_NONE);
        gpio_direction_output(EXYNOS4_GPL2(4), 1);
        gpio_free(EXYNOS4_GPL2(4));
        mdelay(20);
        err = gpio_request_one(EXYNOS4_GPC1(2), GPIOF_OUT_INIT_HIGH, "GPC1_2");
        if (err) {
                printk(KERN_ERR "failed to request GPC1 for "
                        "lcd backlight control\n");
                return err;
```

```
        }
        s3c_gpio_setpull(EXYNOS4_GPC1(2),S3C_GPIO_PULL_NONE);
        gpio_direction_output(EXYNOS4_GPC1(2), 0);
        gpio_free(EXYNOS4_GPC1(2));

        mdelay(20);
        err = gpio_request(EXYNOS4_GPD0(1), "GPD0_1");

        if (err) {
                printk(KERN_ERR "failed to request GPD0_1 for "
                        "lcd pwm control\n");
                return err;
        }
        s3c_gpio_setpull(EXYNOS4_GPD0(1),S3C_GPIO_PULL_NONE);
        s5p_gpio_set_drvstr(EXYNOS4_GPD0(1), S5P_GPIO_DRVSTR_LV4);
        gpio_direction_output(EXYNOS4_GPD0(1), 1);
        s3c_gpio_cfgpin(EXYNOS4_GPD0(1), EXYNOS4_GPD_0_1_TOUT_1);

        gpio_free(EXYNOS4_GPD0(1));
        mdelay(20);
        return 0;

        return 0;
}

int s3cfb_backlight_off(struct platform_device *pdev)
{
        int err;
        err = gpio_request_one(EXYNOS4_GPL2(4), GPIOF_OUT_INIT_LOW, "GPL2_4");
        if (err) {
                printk(KERN_ERR "failed to request GPL2 for "
                        "lcd backlight control\n");
                return err;
        }
        s3c_gpio_setpull(EXYNOS4_GPL2(4),S3C_GPIO_PULL_NONE);
        gpio_direction_output(EXYNOS4_GPL2(4), 0);
        gpio_free(EXYNOS4_GPL2(4));

        err = gpio_request_one(EXYNOS4_GPC1(2), GPIOF_OUT_INIT_HIGH, "GPC1_2");
        if (err) {
                printk(KERN_ERR "failed to request GPC1 for "
                        "lcd backlight control\n");
                return err;
        }
        gpio_free(EXYNOS4_GPC1(2));

        return 0;
}
```

5、PWM的设置

arch/arm/mach-exynos/mach-smdk4x12.c



Arch/arm/mach-exynos/mach-smdk4x12.c

```
/* LCD Backlight data */
static struct samsung_bl_gpio_info smdk4x12_bl_gpio_info = {
        .no = EXYNOS4_GPD0(1),//PWM管脚XpwmTOUT1/LCD_PWM/GPD0_1
        .func = S3C_GPIO_SFN(2),
};
static struct platform_pwm_backlight_data smdk4x12_bl_data = {
        .pwm_id = 1,//PWM ID PWM编号为1号；
        .pwm_period_ns  = 90000,  //22k
};
static void __init smdk4x12_machine_init(void)
{
.................
samsung_bl_set(&smdk4x12_bl_gpio_info, &smdk4x12_bl_data);//在初始化的时候把对应的结构体初始化
.................
}
```

samsung_bl_set看下这个函数的实现：

kernel/arch/arm/palt-samsung/dev-backlight.c

```
void samsung_bl_set(struct samsung_bl_gpio_info *gpio_info,
        struct platform_pwm_backlight_data *bl_data)
{
        int ret = 0;
        struct platform_device *samsung_bl_device;
        struct platform_pwm_backlight_data *samsung_bl_data;

        samsung_bl_device = kmemdup(&samsung_dfl_bl_device,
                        sizeof(struct platform_device), GFP_KERNEL);// (1) 、分配内存空间；
        if (!samsung_bl_device) {
                printk(KERN_ERR "%s: no memory for platform dev\n", __func__);
                return;
        }

        samsung_bl_data = s3c_set_platdata(&samsung_dfl_bl_data,
                sizeof(struct platform_pwm_backlight_data), samsung_bl_device);// (2) 、
        if (!samsung_bl_data) {
                printk(KERN_ERR "%s: no memory for platform dev\n", __func__);
                goto err_data;
        }

        /* Copy board specific data provided by user */
        samsung_bl_data->pwm_id = bl_data->pwm_id;//(3) 、把具体配置的数据给samsung_bl_data
        samsung_bl_device->dev.parent =
                        &s3c_device_timer[samsung_bl_data->pwm_id].dev;

        if (bl_data->max_brightness)//(4)、对bl_data的结构体检查，如果没有复制则用default的值
                samsung_bl_data->max_brightness = bl_data->max_brightness;
        if (bl_data->dft_brightness)
                samsung_bl_data->dft_brightness = bl_data->dft_brightness;
```

```
        if (bl_data->lth_brightness)
                samsung_bl_data->lth_brightness = bl_data->lth_brightness;
        if (bl_data->pwm_period_ns)
                samsung_bl_data->pwm_period_ns = bl_data->pwm_period_ns;
        if (bl_data->init)
                samsung_bl_data->init = bl_data->init;
        if (bl_data->notify)
                samsung_bl_data->notify = bl_data->notify;
        if (bl_data->exit)
                samsung_bl_data->exit = bl_data->exit;
        if (bl_data->check_fb)
                samsung_bl_data->check_fb = bl_data->check_fb;

        /* Keep the GPIO info for future use */
        s3c_device_timer[samsung_bl_data->pwm_id].dev.platform_data = gpio_info;

        /* Register the specific PWM timer dev for Backlight control */
        ret = platform_device_register(//(5)、注册PWM设备驱动；
                        &s3c_device_timer[samsung_bl_data->pwm_id]);
        if (ret) {
                printk(KERN_ERR "failed to register pwm timer for backlight: %d\n", ret);
                goto err_plat_reg1;
        }

        /* Register the Backlight dev */
        ret = platform_device_register(samsung_bl_device);//(6)、注册背光设备驱动；
        if (ret) {
                printk(KERN_ERR "failed to register backlight device: %d\n", ret);
                goto err_plat_reg2;
        }

        return;

err_plat_reg2://(7)、如果有异常的情况下退出；
        platform_device_unregister(&s3c_device_timer[samsung_bl_data->pwm_id]);
err_plat_reg1:
        kfree(samsung_bl_data);
err_data:
        kfree(samsung_bl_device);
        return;
}
```

(1)、分配内存空间

```
        samsung_bl_device = kmemdup(&samsung_dfl_bl_device,
                        sizeof(struct platform_device), GFP_KERNEL);
```

其中：

```
static struct platform_pwm_backlight_data samsung_dfl_bl_data = {
        .max_brightness = 255,
        .dft_brightness = 140,
        .pwm_period_ns  = 78770,
        .init           = samsung_bl_init,
        .exit           = samsung_bl_exit,
};
```

```
static struct platform_device samsung_dfl_bl_device = {
        .name           = "pwm-backlight",
};
```

(2)、

(3)、把具体配置的数据给**samsung_bl_data**

```
arch/arm/mach-exynos/mach-smdk4x12.c
static struct samsung_bl_gpio_info smdk4x12_bl_gpio_info = {
        .no = EXYNOS4_GPD0(1),//PWM管脚XpwmTOUT1/LCD_PWM/GPD0_1
        .func = S3C_GPIO_SFN(2),
};
static struct platform_pwm_backlight_data smdk4x12_bl_data = {
        .pwm_id = 1,//PWM ID PWM编号为1号；
        .pwm_period_ns  = 90000,  //22k
};
```

(4)、对**bl_data**的结构体检查，如果没有复制则用**default**的值

参考（1）中的那些值。

(5)、注册**PWM**设备驱动；

```
    ret = platform_device_register(
            &s3c_device_timer[samsung_bl_data->pwm_id]);
```

其中s3c_device_timer[]这个结构体如下：

```
struct platform_device s3c_device_timer[] = {
        [0] = { DEFINE_S3C_TIMER(0, IRQ_TIMER0) },
        [1] = { DEFINE_S3C_TIMER(1, IRQ_TIMER1) },
        [2] = { DEFINE_S3C_TIMER(2, IRQ_TIMER2) },
        [3] = { DEFINE_S3C_TIMER(3, IRQ_TIMER3) },
        [4] = { DEFINE_S3C_TIMER(4, IRQ_TIMER4) },
```

我们饿samsung_bl_data->pwm_id=1;所以选择[1] = { DEFINE_S3C_TIMER(1, IRQ_TIMER1) },

(6)、注册背光设备驱动；

```
    ret =platform_device_register(samsung_bl_device);
```

其中：samsung_bl_device

```
        samsung_bl_data = s3c_set_platdata(&samsung_dfl_bl_data,
                sizeof(struct platform_pwm_backlight_data), samsung_bl_device);
```

```
root@android:/sys/devices/platform/s3c24xx-pwm.1/pwm-backlight.0/backlight/pwm-backlight.0 #
root@android:/sys/devices/platform/s3c24xx-pwm.1/pwm-backlight.0/backlight/pwm-backlight.0 #
root@android:/sys/devices/platform/s3c24xx-pwm.1/pwm-backlight.0/backlight/pwm-backlight.0 # cat
130|root@android:/sys/devices/platform/s3c24xx-pwm.1/pwm-backlight.0/backlight/pwm-backlight.0 #
130|root@android:/sys/devices/platform/s3c24xx-pwm.1/pwm-backlight.0/backlight/pwm-backlight.0 # ls
actual_brightness
bl_power
brightness
device
max_brightness
power
subsystem
type
uevent
acklight.0 # cat actual_brightness                                    backlight.0 # cat
249
root@android:/sys/devices/platform/s3c24xx-pwm.1/pwm-backlight.0/backlight/pwm-backlight.0 #
acklight.0 # cat bl_power                                             backlight.0 # cat
0
acklight.0 # cat brightness                                           backlight.0 # cat
249
root@android:/sys/devices/platform/s3c24xx-pwm.1/pwm-backlight.0/backlight/pwm-backlight.0 #
```

(7)、如果有异常的情况下退出；

6、**PWM_BL**背光驱动分析：

Kernel/drivers/video/backlight/pwm_bl.c

（**1**）、驱动注册：

```
static struct platform_driver pwm_backlight_driver = {
        .driver         = {
                .name   = "pwm-backlight",
                .owner  = THIS_MODULE,
        },
        .probe          = pwm_backlight_probe,
        .remove         = pwm_backlight_remove,
        .suspend        = pwm_backlight_suspend,
        .resume         = pwm_backlight_resume,
};

static int __init pwm_backlight_init(void)
{
        return platform_driver_register(&pwm_backlight_driver);
}
```

（**2**）、**probe**函数分析

```
static int pwm_backlight_probe(struct platform_device *pdev)
{
        struct backlight_properties props;
        struct platform_pwm_backlight_data *data = pdev->dev.platform_data;
        struct backlight_device *bl;
        struct pwm_bl_data *pb;
        int ret;

        if (!data) {
                dev_err(&pdev->dev, "failed to find platform data\n");
                return -EINVAL;
        }

        if (data->init) {
                ret = data->init(&pdev->dev);
                if (ret < 0)
                        return ret;
        }

        pb = kzalloc(sizeof(*pb), GFP_KERNEL);
        if (!pb) {
                dev_err(&pdev->dev, "no memory for state\n");
                ret = -ENOMEM;
                goto err_alloc;
        }

        global_pb=pb;
        INIT_DELAYED_WORK_DEFERRABLE(&key_event, key_event_work);//1）、任务队列初始化；

        pb->period = data->pwm_period_ns;//2）、pb结构体初始化；
        pb->notify = data->notify;
        pb->check_fb = data->check_fb;
```

```
        pb->lth_brightness = data->lth_brightness *
                (data->pwm_period_ns / data->max_brightness);
        pb->dev = &pdev->dev;

        pb->pwm = pwm_request(data->pwm_id, "backlight");
        if (IS_ERR(pb->pwm)) {
                dev_err(&pdev->dev, "unable to request PWM for backlight\n");
                ret = PTR_ERR(pb->pwm);
                goto err_pwm;
        } else
                dev_dbg(&pdev->dev, "got pwm for backlight\n");

        memset(&props, 0, sizeof(struct backlight_properties));
        props.type = BACKLIGHT_RAW;
        props.max_brightness = data->max_brightness;
        bl = backlight_device_register(dev_name(&pdev->dev), &pdev->dev, pb,
                                       &pwm_backlight_ops, &props);
        if (IS_ERR(bl)) {
                dev_err(&pdev->dev, "failed to register backlight\n");
                ret = PTR_ERR(bl);
                goto err_bl;
        }
        global_bl=bl;
        bl->props.brightness = data->dft_brightness;
        backlight_update_status(bl);//3）、更新背光状态；

        platform_set_drvdata(pdev, bl);
        return 0;

err_bl:
        pwm_free(pb->pwm);
err_pwm:
        kfree(pb);
err_alloc:
        if (data->exit)
                data->exit(&pdev->dev);
        return ret;
}
```

1）、任务队列初始化；

把key_event_work加入key_event队列，

```
INIT_DELAYED_WORK_DEFERRABLE(&key_event, key_event_work);
```

队列调度函数：

```
static void key_event_work(struct work_struct *work)
{
        global_pb->period=90000;
        global_bl->props.brightness=global_brightness;
        backlight_update_status(global_bl);
        return ;
}
```

backlight_update_status

```
static inline void backlight_update_status(struct backlight_device *bd)
```

```
{
        mutex_lock(&bd->update_lock);
        if (bd->ops && bd->ops->update_status)
                bd->ops->update_status(bd);
        mutex_unlock(&bd->update_lock);
}
```

update_status在pwm_backlight_ops结构体重指定：

```
static const struct backlight_ops pwm_backlight_ops = {
        .update_status  = pwm_backlight_update_status,
        ……}
```

pwm_backlight_update_status我们后面分析，这个其实就是我们PWM设定实现的具体实施过程。

2）、**pb**结构体初始化：

```
        pb->period = data->pwm_period_ns;       pb->notify = data->notify;
        pb->check_fb = data->check_fb;
        pb->lth_brightness = data->lth_brightness *
                (data->pwm_period_ns / data->max_brightness);
        pb->dev = &pdev->dev;
        pb->pwm = pwm_request(data->pwm_id, "backlight");
```

3）、更新背光状态

```
        backlight_update_status(bl);
```

（4）、PWM_SET

当UI设置PWM时，会调用到驱动中的pwm_set(void)这个函数。这个函数主要在开机时使用。

```
int pwm_set(void)
{
        int error;
        struct backlight_device *bl = global_bl;
        struct pwm_bl_data *pb = global_pb;
        printk("%s__%d\n",__func__,pb->period);
        pb->period=410000;
        backlight_update_status(bl);
        schedule_delayed_work(&key_event, 600); //调用队列，跟新亮度信息；


        return 0;
}
```

（5）、pwm_backlight_update_status这个就是**PWM**变化的具体实现，当应用层调节时，会调用到这个函数，把改变的值填入寄存器。

```
static int pwm_backlight_update_status(struct backlight_device *bl)
{
        struct pwm_bl_data *pb = dev_get_drvdata(&bl->dev);
        int brightness = bl->props.brightness;
        int max = bl->props.max_brightness;
        //if(brightness==0)
        //      return 0;
        //printk("#####%s#%d__%d\n",__func__,pb->period,brightness);
        global_brightness=brightness;
        if (bl->props.power != FB_BLANK_UNBLANK)
                brightness = 0;
```

```
        if (bl->props.fb_blank != FB_BLANK_UNBLANK)
                brightness = 0;

        if (pb->notify)
                brightness = pb->notify(pb->dev, brightness);

        if (brightness == 0) {
                pwm_config(pb->pwm, 0, pb->period);
                pwm_disable(pb->pwm);
        } else {
        #if 1
                brightness = pb->lth_brightness +
                        (brightness * (pb->period - pb->lth_brightness) / max);
        #else
         brightness = pb->lth_brightness +
                        (((pb->period - pb->lth_brightness) / max) * brightness );
        #endif

                pwm_config(pb->pwm, brightness, pb->period);//这里对PWM寄存器的具体操作；
                pwm_enable(pb->pwm);
        }
        return 0;
}
```

```
<4>[   24.255286] #####pwm_backlight_update_status#90000__165
<4>[   24.305292] #####pwm_backlight_update_status#90000__172
<4>[   24.354550] #####pwm_backlight_update_status#90000__177
<4>[   24.404535] #####pwm_backlight_update_status#90000__181
<4>[   24.454131] #####pwm_backlight_update_status#90000__185
<4>[   24.504029] #####pwm_backlight_update_status#90000__187
<4>[   24.553325] #####pwm_backlight_update_status#90000__190
<4>[   24.602884] #####pwm_backlight_update_status#90000__192
<4>[   24.652212] #####pwm_backlight_update_status#90000__195
<4>[   24.801082] #####pwm_backlight_update_status#90000__196
<4>[   24.851187] #####pwm_backlight_update_status#90000__198
<4>[   24.900578] #####pwm_backlight_update_status#90000__199
<4>[   24.956439] #####pwm_backlight_update_status#90000__201
<4>[   25.002287] #####pwm_backlight_update_status#90000__202
<4>[   25.050954] #####pwm_backlight_update_status#90000__203
<4>[   25.102272] #####pwm_backlight_update_status#90000__204
<4>[   25.149257] #####pwm_backlight_update_status#90000__205
<4>[   25.347610] #####pwm_backlight_update_status#90000__206
<5>[   25.935565] CPU1: shutdown
```


三、**LCD UBOOT**下的控制（待整理.....）

LCD在UBOOT下的控制，这部分我们没做过，后面有机会做了再把这部分完善，或者找个机会把代码详细看看。

上一篇 Android LCD(三)：Samsung LCD接口篇

下一篇 蓝牙核心技术概述（一）:蓝牙概述