A10 lcd 用户配置手册

1. 总体介绍

显示驱动中开放出来两个文件给用户进行屏相关的配置，分别是 lcd0_panel_cfg.c 和 lcd1_panel_cfg.c，分别对应两个屏（A10 支持双屏输出）。这两个文件是与操作系统无关的，即无论用在哪个操作系统上(boot, melis, linux 或 wince)，它们的代码应该是完全一致的。所有与操作系统相关的操作都由驱动抽象成统一的接口给用户使用。

pin 脚相关的配置在文件 sys_config.fex 里定义。

2. lcd0_panel_cfg.c和lcd1_panel_cfg.c

在驱动里需要配置的文件有两个，lcd0_panel_cfg.c 和 lcd1_panel_cfg.c，分别对应两个屏幕。以下是这两个文件的一览，分别列出了文件中必须要包含的 4 个函数。

file: lcd0_panel_cfg.c

```
static void LCD_cfg_panel_info(__panel_para_t * info)
{
//……
}

static __s32 LCD_open_flow(__u32 sel)
{
//……
}

static __s32 LCD_close_flow(__u32 sel)
{
//……
}

void LCD_get_panel_funs_0(__lcd_panel_fun_t * fun)
{
    fun->cfg_panel_info = LCD_cfg_panel_info;
    fun->cfg_open_flow = LCD_open_flow;
    fun->cfg_close_flow = LCD_close_flow;
}
```

file: lcd1_panel_cfg.c

```
static void LCD_cfg_panel_info(__panel_para_t * info)
{
//……
```

```
}

static __s32 LCD_open_flow(__u32 sel)
{
//……
}

static __s32 LCD_close_flow(__u32 sel)
{
//……
}

void LCD_get_panel_funs_1(__lcd_panel_fun_t * fun)
{
    fun->cfg_panel_info = LCD_cfg_panel_info;
    fun->cfg_open_flow = LCD_open_flow;
    fun->cfg_close_flow = LCD_close_flow;
}
```

3. 函数介绍

1) LCD_cfg_panel_info

函数功能：配置屏的基本参数。

示例：

```
static void LCD_cfg_panel_info(__panel_para_t * info)
{
    memset(info,0,sizeof(__panel_para_t));

    /*屏的基本信息*/
    info->lcd_x                 = 480;
    info->lcd_y                 = 272;
    info->lcd_dclk_freq         = 9;    //MHz
    info->lcd_pwm_freq          = 1;    //KHz
    info->lcd_srgb              = 0x00202020;
    info->lcd_swap              = 0;

    /*屏的接口配置信息*/
    info->lcd_if                    = 0;//0:HV , 1:8080 I/F, 2:TTL I/F, 3:LVDS
```

```
    /*屏的 HV 模块相关信息*/
    info->lcd_hv_if                 = 0;        //0:hv parallel 1:hv serial
    info->lcd_hv_smode              = 0;        //0:RGB888 1:CCIR656
    info->lcd_hv_syuv_if            = 0;        //serial YUV format
    info->lcd_hv_hspw               = 41;       //hsync plus width
    info->lcd_hv_vspw               = 10;       //vysnc plus width

    /*屏的 HV 配置信息*/
    info->lcd_hbp                   = 2;        //hsync back porch
    info->lcd_ht                    = 525;      //hsync total cycle
    info->lcd_vbp                   = 2;        //vsync back porch
    info->lcd_vt                    = (2 * 286);//vysnc total cycle *2

    //cpu 屏幕的配置信息
    info->lcd_cpu_if        =0;//0:18bit 4:16bit
    info->lcd_frm           = 1;  //0: disable; 1: enable rgb666 dither; 2:enable rgb656 dither

    /*屏的 IO 配置信息*/
    info->lcd_io_cfg0               = 0x00000000;
    info->lcd_io_cfg1               = 0x00000000;
    info->lcd_io_strength           = 0;
}
```

__panel_para_t 数据结构定义：

```
typedef struct
{
    __u8    lcd_if;//0:hv(sync+de); 1:8080; 2:ttl; 3:lvds
    __u8    lcd_swap;
    __u16   lcd_x;
    __u16   lcd_y;
    __u16   lcd_dclk_freq;

    __u8    lcd_uf;
    __u16   lcd_vt;
    __u16   lcd_ht;
    __u16   lcd_vbp;
    __u16   lcd_hbp;

    __u8    lcd_hv_if;
    __u8    lcd_hv_smode;
    __u8    lcd_hv_s888_if;
    __u8    lcd_hv_syuv_if;
    __u8    lcd_hv_vspw;
```

```
__u16    lcd_hv_hspw;

__u8     lcd_hv_lde_used;
__u8     lcd_hv_lde_iovalue;

__u32    lcd_ttl_stvh;
__u32    lcd_ttl_stvdl;
__u32    lcd_ttl_stvdp;

__u32    lcd_ttl_ckvt;
__u32    lcd_ttl_ckvh;
__u32    lcd_ttl_ckvd;

__u32    lcd_ttl_oevt;
__u32    lcd_ttl_oevh;
__u32    lcd_ttl_oevd;

__u32    lcd_ttl_sthh;
__u32    lcd_ttl_sthd;
__u32    lcd_ttl_oehh;
__u32    lcd_ttl_oehd;

__u32    lcd_ttl_revd;

__u32    lcd_ttl_datarate;
__u32    lcd_ttl_revsel;
__u32    lcd_ttl_datainv_en;
__u32    lcd_ttl_datainv_sel;
__u8     lcd_cpu_if;
__u8     lcd_cpu_da;

__u8     lcd_frm;
__u32    lcd_io_cfg0;
__u32    lcd_io_cfg1;

__u32    lcd_srgb;
__u32    lcd_io_strength;

__u32    lcd_pwm_freq;
__u32    lcd_pwm_pol;

__u32    start_delay; //not need to set for user
__u32    tcon_index; //not need to set for user
```

}__panel_para_t;

2) LCD_open_flow

函数功能：定义开屏的流程。

示例:

```
static __s32 LCD_open_flow(__u32 sel)
{
    LCD_OPEN_FUNC(sel, LCD_power_on, 10); //打开 LCD 供电,并延迟 10ms
    LCD_OPEN_FUNC(sel, TCON_open, 200); //打开 LCD 控制器,并延迟 200ms
    LCD_OPEN_FUNC(sel, LCD_bl_open, 0); //打开背光,并延迟 0ms

    return 0;
}
```

在上例子中，开屏总共有三个步骤，分别是打开 LCD power，延迟 10ms 后打开 LCD 控制器，延迟 200ms 打开背光，再延迟 0ms 后完成开屏操作。

LCD_OPEN_FUNC 函数的第一个参数 sel 用户可以忽略，是驱动用来传递参数用的。

LCD_OPEN_FUNC 函数的第二个参数是一个函数指针，其类型是：void (*LCD_FUNC)(__u32 sel)，用户自己定义的函数必须也要用统一的形式。比如：

```
void do_something_else(__u32 sel)
{
    //do something
}
```

LCD_OPEN_FUNC 函数的第三个参数是执行该步骤后需要延迟的时间，时间单位是毫秒。

注意在该函数里每个开屏步骤都要以统一的格式来描述，即 LCD_OPEN_FUNC(sel, function, delay_time)。因为该函数只会在最开始的时候调用一遍，目的是把每个步骤记录下来，并没有真正执行每个步骤(驱动会在合适的时间让它们执行)。而想要让驱动将你需要的步骤记录下来的唯一方式就是用上面的格式表示它们。

因此下面这种方式是错误的：

```
static __s32 LCD_open_flow(__u32 sel)
{
    LCD_OPEN_FUNC(sel, LCD_power_on, 10); //打开 LCD 供电,并延迟 10ms
    do_something_else();
    LCD_OPEN_FUNC(sel, TCON_open, 200); //打开 LCD 控制器,并延迟 200ms
    LCD_OPEN_FUNC(sel, LCD_bl_open, 0); //打开背光,并延迟 0ms

    return 0;
}
```

而应该用下面这种方式：

```
static __s32 LCD_open_flow(__u32 sel)
{
    LCD_OPEN_FUNC(sel, LCD_power_on, 10); //打开 LCD 供电,并延迟 10ms
    LCD_OPEN_FUNC(sel, do_something_else,0);
    LCD_OPEN_FUNC(sel, TCON_open, 200); //打开 LCD 控制器,并延迟 200ms
    LCD_OPEN_FUNC(sel, LCD_bl_open, 0); //打开背光,并延迟 0ms

    return 0;
}
```

3) LCD_close_flow

该函数与 LCD_open_flow 类似，区别只是该函数是定义关屏的流程。

4) LCD_get_panel_funs_0

该函数无需用户修改，只在文件 lcd0_panel_cfg.c 中有定义。

5) LCD_get_panel_funs_1

该函数无需用户修改，只在文件 lcd1_panel_cfg.c 中有定义。

4. 可供用户使用的函数介绍

为了让用户在配置屏文件时更加方便，并忽略操作系统的差异，驱动会提供接口统一的函数给用户使用，下面将逐一进行介绍。

1) LCD_delay_ms

void LCD_delay_ms(__u32 ms);

2) TCON_open

void TCON_open(__u32 sel)；
打开 LCD 控制器。

3) TCON_close

void TCON_close(__u32 sel);

关闭 LCD 控制器。

4) LCD_PWM_EN

void LCD_PWM_EN (__u32 sel, __bool b_en);
b_en==0: disable pwm，将 PWM pin 设为输入，并把 PWM 模块关闭。
b_en==1: enable pwm，将 PWM pin 设为 PWM，并把 PWM 模块打开。

5) LCD_BL_EN

void LCD_BL_EN (__u32 sel, __bool b_en);
打开或关闭 LCD 背光;
b_en==0: set LCD_BL_EN IO to disable backlight;
b_en==1: set LCD_BL_EN IO to enable backlight;

6) LCD_PWR_EN

void LCD_PWR_EN(__u32 sel, __bool b_en);
打开或关闭 LCD-VCC;
b_en==0: set PWR_EN IO to disable lcd power;
b_en==1: set PWR_EN IO to enable lcd power;

7) LCD_cpu_register_irq

void LCD_cpu_register_irq(__u32 sel, void (*Lcd_cpuisr_proc) (void));

用于 cpu 屏，注册 cpu 屏的中断处理函数，驱动会在每个 vblanking 中断里调用一下用户注册的中断处理函数 Lcd_cpuisr_proc。

8) LCD_CPU_WR

void LCD_CPU_WR(__u32 sel, __u32 index, __u32 data);

9) LCD_CPU_WR_INDEX (RS=0)

void LCD_CPU_WR_INDEX(__u32 sel,__u32 index);

10) LCD_CPU_WR_DATA (RS=1)

void LCD_CPU_WR_DATA(__u32 sel, __u32 data);

11) LCD_CPU_AUTO_FLUSH

void LCD_CPU_AUTO_FLUSH(__u32 sel, __bool en);

12) LCD_GPIO_request

__s32 LCD_GPIO_request(__u32 sel,__u32 io_index) ;
used for 2/3-wire I/F,request io;
io_index=0/1/2/3

13) LCD_GPIO_release

__s32 LCD_GPIO_release(__u32 sel,__u32 io_index);
used for 2/3-wire I/F,release io

14) LCD_GPIO_set_attr

__s32 LCD_GPIO_set_attr(__u32 sel,__u32 io_index, __bool b_output);
used for 2/3-wire I/F
b_output==0: input; b_output==1:output

15) LCD_GPIO_read

__s32 LCD_GPIO_read(__u32 sel,__u32 io_index);
used for 2/3-wire I/F

16) LCD_GPIO_write

__s32 LCD_GPIO_write(__u32 sel,__u32 io_index, __u32 data);
used for 2/3-wire I/F

17) sys_get_wvalue

#define sys_get_wvalue(n) (*((volatile __u32 *)(n))) /* word input */
32 位的读操作；n 为地址。

18) sys_put_wvalue

#define sys_put_wvalue(n,c) (*((volatile __u32 *)(n)) = (c)) /* word output */
32 位的写操作；n 为地址，c 为值。

5. sys_config.fex

LCD gpio 配置示例(23 evb)：

```
;----------------------------------------------------------------------------------
;lcd0 io interface configuration
;----------------------------------------------------------------------------------
[lcd0_para]
lcd0_para_used    = 1

LCD_BL_EN_USED    = 0
LCD_BL_EN             =

LCD_POWER_USED    = 1
LCD_POWER             = port:PH08<1><default><default><0>

LCD_PWM_EN_USED = 1
LCD_PWM_EN          = port:PB02<2><default><default>< default>

LCD_GPIO_0          =
LCD_GPIO_1          =
LCD_GPIO_2          =
LCD_GPIO_3          =

LCDD0       = port:PD00<2><default><default><default>
LCDD1       = port:PD01<2><default><default><default>
LCDD2       = port:PD02<2><default><default><default>
LCDD3       = port:PD03<2><default><default><default>
LCDD4       = port:PD04<2><default><default><default>
LCDD5       = port:PD05<2><default><default><default>
LCDD6       = port:PD06<2><default><default><default>
LCDD7       = port:PD07<2><default><default><default>
LCDD8       = port:PD08<2><default><default><default>
LCDD9       = port:PD09<2><default><default><default>
LCDD10      = port:PD10<2><default><default><default>
LCDD11      = port:PD11<2><default><default><default>
LCDD12      = port:PD12<2><default><default><default>
LCDD13      = port:PD13<2><default><default><default>
LCDD14      = port:PD14<2><default><default><default>
LCDD15      = port:PD15<2><default><default><default>
```

```
LCDD16      = port:PD16<2><default><default><default>
LCDD17      = port:PD17<2><default><default><default>
LCDD18      = port:PD18<2><default><default><default>
LCDD19      = port:PD19<2><default><default><default>
LCDD20      = port:PD20<2><default><default><default>
LCDD21      = port:PD21<2><default><default><default>
LCDD22      = port:PD22<2><default><default><default>
LCDD23      = port:PD23<2><default><default><default>
LCDCLK      = port:PD24<2><default><default><default>
LCDDE       = port:PD25<2><default><default><default>
LCDHSYNC = port:PD26<2><default><default><default>
LCDVSYNC = port:PD27<2><default><default><default>

;-------------------------------------------------------------------------------
;lcd1 io interface configuration
;-------------------------------------------------------------------------------
[lcd1_para]
lcd1_para_used = 0

LCD_BL_EN_USED    = 0
LCD_BL_EN            =

LCD_POWER_USED    = 0
LCD_POWER            =

LCD_PWM_EN _USED = 0
LCD_PWM_EN          = port:PI03<2><default><default>< default>

LCD_GPIO_0          =
LCD_GPIO_1          =
LCD_GPIO_2          =
LCD_GPIO_3          =

LCDD0       = port:PH00<2><default><default><default>
LCDD1       = port:PH01<2><default><default><default>
LCDD2       = port:PH02<2><default><default><default>
LCDD3       = port:PH03<2><default><default><default>
LCDD4       = port:PH04<2><default><default><default>
LCDD5       = port:PH05<2><default><default><default>
LCDD6       = port:PH06<2><default><default><default>
LCDD7       = port:PH07<2><default><default><default>
LCDD8       = port:PH08<2><default><default><default>
LCDD9       = port:PH09<2><default><default><default>
```

LCDD10 = port:PH10<2><default><default><default>
LCDD11 = port:PH11<2><default><default><default>
LCDD12 = port:PH12<2><default><default><default>
LCDD13 = port:PH13<2><default><default><default>
LCDD14 = port:PH14<2><default><default><default>
LCDD15 = port:PH15<2><default><default><default>
LCDD16 = port:PH16<2><default><default><default>
LCDD17 = port:PH17<2><default><default><default>
LCDD18 = port:PH18<2><default><default><default>
LCDD19 = port:PH19<2><default><default><default>
LCDD20 = port:PH20<2><default><default><default>
LCDD21 = port:PH21<2><default><default><default>
LCDD22 = port:PH22<2><default><default><default>
LCDD23 = port:PH23<2><default><default><default>
LCDCLK = port:PH24<2><default><default><default>
LCDDE = port:PH25<2><default><default><default>
LCDHSYNC = port:PH26<2><default><default><default>
LCDVSYNC = port:PH27<2><default><default><default>

1) lcd0_para_used

0: lcd0 interface not exist;
1:lcd0 interface exist;

2) lcd1_para_used

0: lcd1 interface not exist;
1:lcd1 interface exist;

3) LCD_BL_EN_USED

0: LCD_BL_EN pin used
1: LCD_BL_EN pin not used

4) LCD_BL_EN

LCD_BL_EN pin config;
示例：port:PH08<1><0><default><0>
PH08 output 0 to enable 背光，output 1 to disable 背光，pull up/down disable。

PH：端口分组；
08：组内序号；
第一个尖括号：功能分配，与 SPEC 描述一致；1 为输出；
第二个尖括号：内置电阻；使用 0 的话，标示内部电阻高阻态，如果是 1 则是内部电阻上拉，2 就代表内部电阻下拉。使用 default 的话代表默认状态，即电阻上拉。其它数据无效。
第三个尖括号：驱动能力；
第四个尖括号：输出 0/1 有效；default 表驱动能力是等级 1。

5) LCD_POWER_USED

0: LCD-VCC control pin used
1: LCD-VCC control pin not used

6) LCD_POWER

示例：port:PH08<1><0><default><0>
PH08 output 0 to enable LCD-VCC,output 1 to disable LCD-VCC,pull up/down disable

7) LCD_PWM_USED

0: PWM pin used
1: PWM pin not used

8) LCD_PWM

示例：port:PB02<2><0><default>< default>
PWM pin PB02 output PWM signal;

第一个尖括号：
1: PWM pin output 0/1
2: PWM pin output PWM signal

6. 操作指南

1) 目录

lcd0_panel_cfg.c 和 lcd1_panel_cfg.c 所在的目录是：
…\linux-2.6.36\drivers\video\sun4i\disp\de_bsp\lcd

2) 编辑与编译

lcd0_panel_cfg.c 和 lcd1_panel_cfg.c 文件是在内核态的，因此编辑这两个文件时需注意这点。比如打印是使用 printk 等。
如果当前方案只有一个 LCD 屏，只需要修改文件 lcd0_panel_cfg.c；如果有两个屏（一样或不一样）,则两个文件都必须修改。
目录 lcd_bak 是一些 LCD 配置文件的备份，用户可以作为参考。

lcd0_panel_cfg.c 和 lcd1_panel_cfg.c 文件是显示驱动的一部分，而显示驱动是编译进内核里面的，所以每次修改这两个文件都必须重新编译内核，并重新打包。
修改文件 sys_config.fex 只需重新打包即可。