A10 lcd User Configuration Guide

肖丹灵 Dan Xiao Ling
danling@allwinnertech.com

1. Overview

Display driver out in the open two documents related to the user configuration screen, respectively lcd0_panel_cfg.c and lcd1_panel_cfg.c, corresponding to the two screens (A10 supports dual monitor output). These two files are operating system independent, that is, no matter which operating system (boot, melis, linux or wince), their code should be exactly the same. All operating system-related operations by the driver into a single abstract interface to the users.

pin pin configuration in the file sys_config.fex relevant in the definition.

2. lcd0_panel_cfg.c and lcd1_panel_cfg.c

In the driver configuration files need to have two, lcd0_panel_cfg.c and lcd1_panel_cfg.c, corresponding to the two screens.
The following is a list of these two documents were listed in the file must contain the four functions.

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. Function Descriptions

1) LCD_cfg_panel_info

Function: Configure the basic parameters screen.
Example:

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

   /* Basic information screen */
   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;

    /* Screen interface configuration information */
    info->lcd_if                = 0;//0:HV , 1:8080 I/F, 2:TTL I/F, 3:LVDS

    /* HV module information screen */
        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

    /* HV configuration screen */
    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 Screen configuration information
        info->lcd_cpu_if      =0;//0:18bit 4:16bit
        info->lcd_frm         = 1;  //0: disable; 1: enable rgb666 dither; 2:enable rgb656 dither

    /*IO configuration information screen*/
    info->lcd_io_cfg0           = 0x00000000;
    info->lcd_io_cfg1           = 0x00000000;
    info->lcd_io_strength       = 0;
}
```

__panel_para_t data structure definition：

```
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

Function: Defines the opening screen of the process.
Example:

```
static __s32 LCD_open_flow(__u32 sel)
{
    LCD_OPEN_FUNC(sel, LCD_power_on, 10); //Open the LCD power supply, and delay 10ms
    LCD_OPEN_FUNC(sel, TCON_open, 200); //Open the LCD controller, and 200ms delay
    LCD_OPEN_FUNC(sel, LCD_bl_open, 0); //Turn on the backlight, and delay 0ms
    return 0;
}
```

In the example, the opening screen, a total of three steps, namely, open the LCD power, LCD controller and opened the 10ms delay, delay 200ms turn on the backlight, and then completed 0ms delay opening screen operation.

LCD_OPEN_FUNC The first parameter sel function can be ignored, is the drive to pass parameters to use.

LCD_OPEN_FUNC Function of the second parameter is a function pointer, its type is：void (*LCD_FUNC) (__u32 sel)， User-defined function must have a unified form. Such as：

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

LCD_OPEN_FUNC The third parameter is a function of the steps required to implement the time delay， Time in milliseconds.

Note that the function in each step should be to open the screen to describe the uniform format, LCD_OPEN_FUNC(sel, function, delay_time). Because the function is only called when the beginning again, the purpose is to record each step, and did not really perform each step (at the right time to drive them perform). And want to drive to record the steps you need only way is to use the above format them.

Therefore, following this approach is wrong：

```
static __s32 LCD_open_flow(__u32 sel)
{
```

```
    LCD_OPEN_FUNC(sel, LCD_power_on, 10); //Open the LCD power supply, and delay 10ms
    do_something_else();
    LCD_OPEN_FUNC(sel, TCON_open, 200); //Open the LCD controller, and 200ms delay
    LCD_OPEN_FUNC(sel, LCD_bl_open, 0); //Turn on the backlight, and delay 0ms

    return 0;
}
```

The following should be used in this way：

```
static __s32 LCD_open_flow(__u32 sel)
{
    LCD_OPEN_FUNC(sel, LCD_power_on, 10); //Open the LCD power supply, and delay 10ms
    LCD_OPEN_FUNC(sel, do_something_else,0);
    LCD_OPEN_FUNC(sel, TCON_open, 200); //Open the LCD controller, and 200ms delay
    LCD_OPEN_FUNC(sel, LCD_bl_open, 0); //Turn on the backlight, and delay 0ms

    return 0;
}
```

3) LCD_close_flow

The function and LCD_open_flow similar, only that the function is defined off-screen process.

4) LCD_get_panel_funs_0

The function without user modification, there is only defined in the file lcd0_panel_cfg.c.

5) LCD_get_panel_funs_1

The function without user modification, there is only defined in the file lcd1_panel_cfg.c.

4. Description of functions available to users

In the configuration screen to allow users to file more convenient, and ignore the differences between the operating system, drivers will provide a unified interface, use the function to the user, the following will be introduced one by one.

1) LCD_delay_ms

void LCD_delay_ms(__u32 ms);

2) TCON_open

void TCON_open(__u32 sel) ;
Open the LCD controller.

3) TCON_close

void TCON_close(__u32 sel);
Close the LCD controller.

4) LCD_PWM_EN

void LCD_PWM_EN (__u32 sel, __bool b_en);
b_en==0: disable pwm, The PWM pin as input, and the PWM module is turned off.
b_en==1: enable pwm, The PWM pin is set to PWM, and the PWM module to open.

5) LCD_BL_EN

void LCD_BL_EN (__u32 sel, __bool b_en);
LCD backlight on or off;
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);
Open or close the 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));

Screen for cpu, cpu screen registered the interrupt handler, the driver will call each vblanking interrupt what the user registered in the interrupt handler 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-bit read operation; n the addresses.

18) sys_put_wvalue

#define sys_put_wvalue(n,c) (*((volatile __u32 *)(n)) = (c))   /* word output */
32-bit write operation; n for the address, c is the value.

5. sys_config.fex

LCD gpio configuration example (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;
Example: port:PH08<1><0><default><0>
PH08 output 0 to enable backlight, output 1 to disable the backlight， pull up/down disable。

PH: port grouping;
08: group number;
The first bracket: function allocation, consistent with the description of SPEC; 1 output;
The second angle brackets: built-in resistance; use 0, then labeled the internal resistance of a high impedance state, if it is an internal pull-up resistor, 2 represents the internal resistance of the drop-down. Representatives using the default if the default state, that is, pull-up resistor. Other data is invalid.
The third angle brackets: drive capacity;
The fourth angle brackets: output 0 / 1 effective; default is level 1 table-driven capability.

5) LCD_POWER_USED

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

6) LCD_POWER

Example: 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

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

The first angle brackets:
1: PWM pin output 0/1
2: PWM pin output PWM signal

6. Operation Guide

1) Contents

lcd0_panel_cfg.c and lcd1_panel_cfg.c directory is:
…\linux-2.6.36\drivers\video\sun4i\disp\de_bsp\lcd

2) Edit and compile

lcd0_panel_cfg.c and lcd1_panel_cfg.c file in kernel mode, so editing these two files need to pay attention to this point. Such as print is to use printk and so on.
If the current program is only an LCD screen, just change the file lcd0_panel_cfg.c; If you have two screens (the same or different), then the two files must be modified.
Directory lcd_bak some LCD backup configuration file, the user can be used as reference.

lcd0_panel_cfg.c and lcd1_panel_cfg.c file is part of the display driver, and the display driver is compiled into the kernel inside, so every time modify these two files must be recompiled the kernel, and re-packaged.
Modify the file sys_config.fex can just re-packaged.