A10 LCD 调试手册

Revision 1.0

Record of Revision

Ver.	Revise Date	Page	Content	Author
V1.0	Nov, 18th 2011		First draft	Part 1,3,5 by dlp dulianping@allwinnerme.com Part 2,4 by danling danling@allwinnertech.com

目录

1. 整体介绍.....3
2. 软件配置说明.....4
2.1. 屏文件说明.....4
2.2. 开关屏流程.....5
2.2.1. 开关屏步骤函数说明.....6
2.2.2. 开关屏流程函数说明.....8
2.3. 对屏的初始化.....9
2.3.1. IO模拟串行接口初始化.....9
2.3.2. CPU屏 8080 总线初始化.....10
2.4. 其它函数.....12
2.4.1. 用户自定义函数.....12
2.4.2. GPIO操作函数.....13
2.4.3. 延时函数.....15
2.5. fex文件.....16
2.5.1. 电源控制IO的定义.....16
2.5.2. 模拟串行接口的IO定义.....17
2.5.3. LCD IO定义.....17
3. TCON参数说明.....19
3.1. 接口参数说明.....19
3.1.1. lcd_if.....19
3.1.2. lcd_hv_if.....19
3.1.3. lcd_hv_srgb_seq0.....19
3.1.4. lcd_hv_srgb_seq1.....20
3.1.5. lcd_hv_syuv_seq.....20
3.1.6. lcd_hv_syuv_fdly.....20
3.1.7. lcd_cpu_if.....20
3.1.8. lcd_lvds_ch.....21
3.1.9. lcd_lvds_bitwidth.....21
3.1.10. lcd_lvds_mode.....21
3.1.11. lcd_frm.....21
3.2. 时序参数说明.....22
3.2.1. lcd_x.....22
3.2.2. lcd_y.....22
3.2.3. lcd_ht.....23
3.2.4. lcd_hbp.....23
3.2.5. lcd_vt.....23
3.2.6. lcd_vbp.....23
3.2.7. lcd_hv_hspw.....23
3.2.8. lcd_hv_vspw.....24
3.2.9. lcd_dclk_freq.....24
3.2.10. lcd_io_cfg0.....24

3.3. 其他参数说明.....24
3.3.1. lcd_pwm_not_used 24
3.3.2. lcd_pwm_ch 25
3.3.3. lcd_pwm_freq 25
3.3.4. lcd_pwm_pol.....25
3.3.5. lcd_gamma_correction_en 25
3.3.6. lcd_gamma_tbl.....25
4. 操作指南.....26
4.1. sys_config1.fex配置 26
4.2. boot 阶段LCD配置.....27
4.3. linux 阶段LCD配置.....30
4.4. 调试.....32
5. 附录.....33
5.1. 屏接口说明.....33
5.1.1. HV RGB同步屏接口 33
5.1.2. CPU/8080 屏接口 35
5.1.3. LVDS屏接口.....35
5.2. A10 与屏的连接说明.....37
5.2.1. LCD IO PORT定义.....38
5.2.2. HV Parallel RGB屏参考连接图.....39
5.2.3. HV Serial RGB屏参考连接图.....40
5.2.4. CPU Parallel RGB666 屏参考连接图.....41
5.2.5. LVDS 2 Single Link屏参考连接图 42
5.2.6. LVDS Dual Link屏参考连接图 43
5.3. 屏文件实例.....44
5.3.1. sys_config1.fex 44
5.3.2. hv_800x480.c 46
5.3.3. hv_800x480_td043.c 49
5.3.4. lvds_1024x600_hds100ifw1.c.....55
5.3.5. cpu_320x240_kgm281i0.c 58
5.4. LCD CHECK LIST.....63

1. 整体介绍

A10 有两路显示系统，支持双屏输出，并行像素数据输出的接口形式，LCD0 从 PD 口输出，LCD1 从 PH 口输出，LVDS0 和 LVDS1 都是从 PD 口输出,如表 1-1 所示。

如果一路输出使用 Dual Link LVDS，占用了 LVDS 的所有引脚，另外一路只能使用其他接口形式输出；其他接口形式组合的双屏输出都支持。

表 1-1 中列出了 A10 支持的接口形式及支持的最大分辨率，LCD0 与 LCD1 完全一致。

表 1-1 A10 LCD 输出 IO 口

Interface		Solutions	LCD0	LCD1
同步 RGB 接口	Parallel RGB	1920 × 1080	PD	PH
	Serial RGB	1280 × 720		
	CCIR656	1280 × 720		
CPU/80 接口	Parallel RGB666	1280 × 720		
	Parallel RGB565	1280 × 720		
	Serial RGB666	1280 × 720		
	Serial RGB565	1280 × 720		
LVDS 接口	Single Link	1920 × 1080	PD0-PD9	PD10-PD19
	Dual Link	1920 × 1080	PD0-PD19	

注：Dual Link LVDS 从 PD 口输出，另一屏显只能选择非 LVDS 从 PH 输出

屏的各种接口说明可参见附录 5.1，各接口与 A10 引脚的连接可以参考附录 5.2。

2. 软件配置说明

2.1. 屏文件说明

在目录…\linux-2.6.36\drivers\video\sun4i\disp\de_bsp\lcd，有 lcd0_panel_cfg.c 和 lcd1_panel_cfg.c 两个文件，分别对应 LCD0 和 LCD1 的配置，这两个文件中，定义了 TCON 的参数，开关屏的流程，还有对屏的初始化操作。

对 IO 位置的定义，包括电源控制，配屏使用的 GPIO，以及 LCD 控制器 IO 的定义在 sys_config.fex 中。

lcd0_panel_cfg.c 和 lcd1_panel_cfg.c 中提供的函数接口如下图所示：

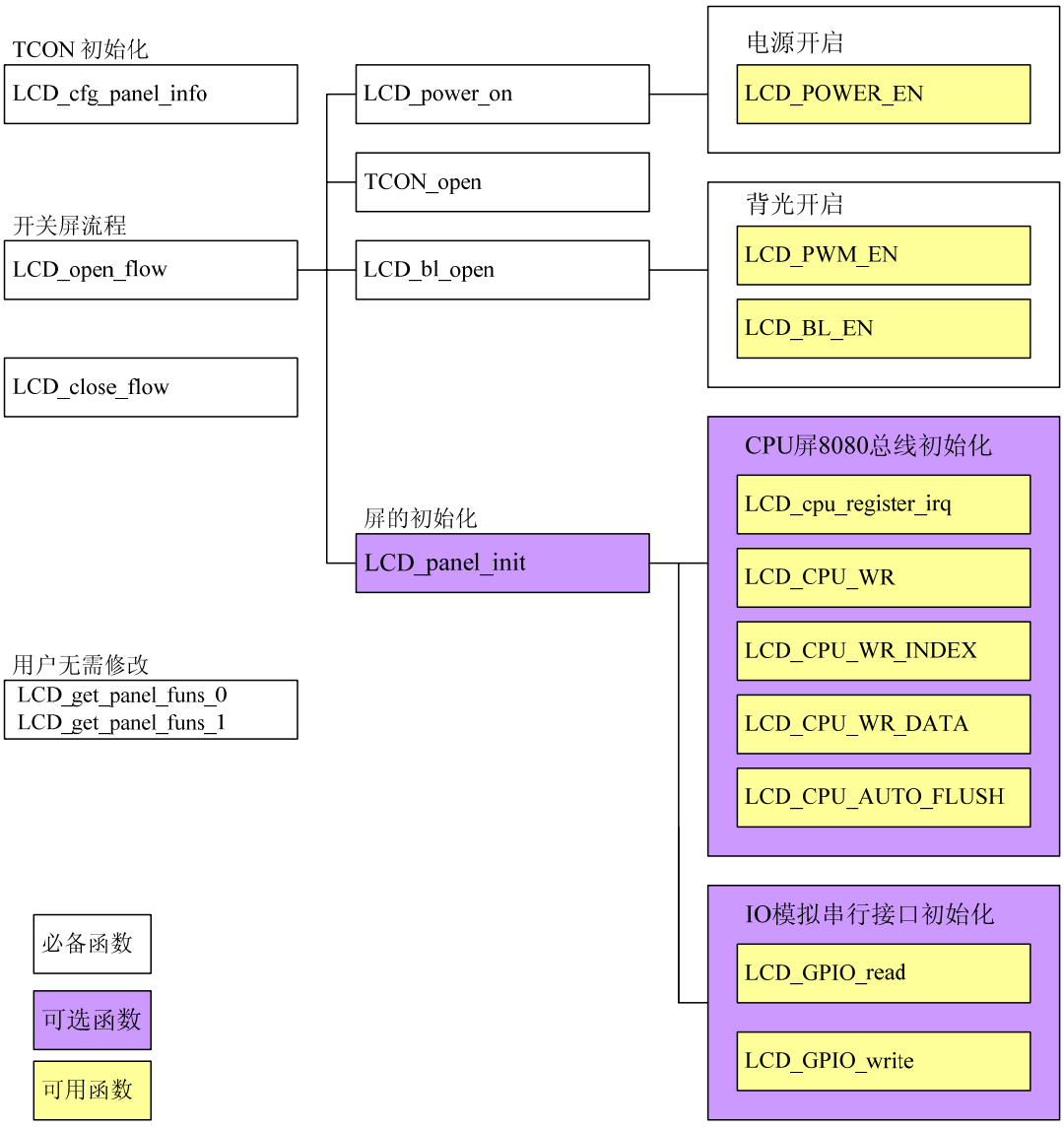


图 2-1 配屏文件中的函数列表

LCD_cfg_panel_info ， LCD_open_flow ， LCD_close_flow 和 LCD_get_panel_funs_0/ LCD_get_panel_funs_1 是必须包含的 4 个函数。

函数：LCD_cfg_panel_info

功能：配置 A10 的 TCON 基本参数

原型：static void LCD_cfg_panel_info(__panel_para_t * info)

参数的定义见“3 TCON 参数说明”。

函数：LCD_open_flow

功能：定义开屏的流程

原型：static __s32 LCD_open_flow(__u32 sel)

具体说明见“2.2 开关屏流程”。

函数：LCD_close_flow

功能：定义关屏的流程

原型：static __s32 LCD_close_flow(__u32 sel)

该函数与 LCD_open_flow 对应

函数：LCD_get_panel_funs_0/ LCD_get_panel_funs_1

功能：

原型：void LCD_get_panel_funs_0(__lcd_panel_fun_t * fun)/

void LCD_get_panel_funs_1(__lcd_panel_fun_t * fun)

该函数无需用户修改，LCD_get_panel_funs_0 只在文件 lcd0_panel_cfg.c 中定义，LCD_get_panel_funs_1 只在文件 lcd1_panel_cfg.c 中定义。

2.2. 开关屏流程

开关屏的常见操作流程如图 2-2 所示。

其中，LCD_open_flow 和 LCD_close_flow 称为开关屏流程函数，方框中的函数，如 LCD_power_on，TCON_open 等函数，称为开关屏步骤函数。

部分屏不需要进行初始化操作，LCD_panel_init 及 LCD_panel_exit 这两个步骤函数（图中紫色框部分）可以省去。

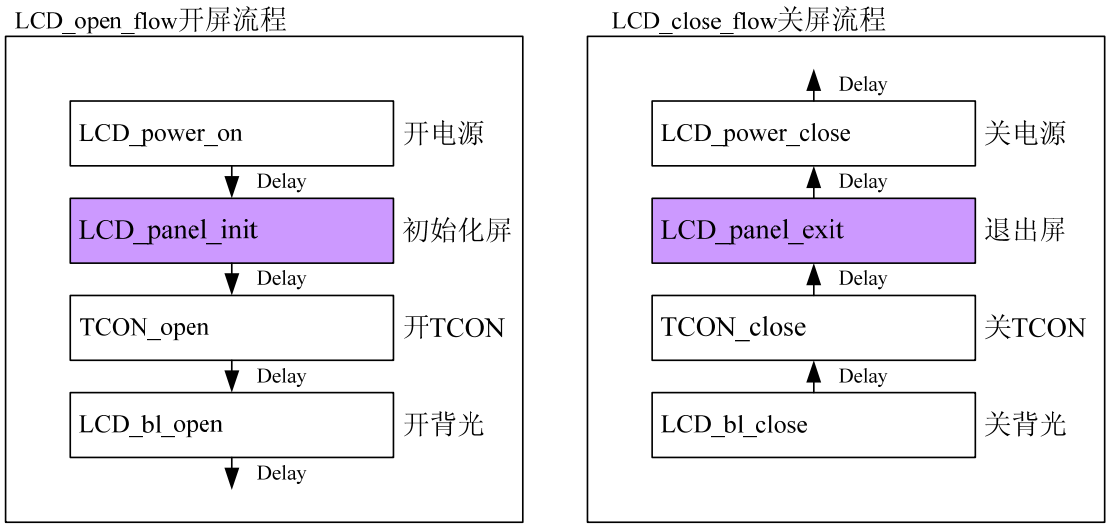


图 2-2 开关屏流程

2.2.1. 开关屏步骤函数说明

开屏的步骤函数有 LCD_panel_init，TCON_open，LCD_power_on，LCD_bl_open。

函数：LCD_panel_init

功能：对屏初始化

原型：static void LCD_panel_init(__u32 sel)

可参考“2.3 对屏的初始化”。部分屏不需要进行初始化操作，LCD_panel_init 及 LCD_panel_exit 这两个步骤函数可以省去。

函数：TCON_open

功能：打开 A10 TCON

原型：__s32 TCON0_open(__u32 sel)

该函数由显示驱动提供，用户无需实现。

函数：LCD_power_on

功能：打开 LCD 电源

原型：static void LCD_power_on(__u32 sel)

显示驱动提供 LCD_PWR_EN 函数可供调用，用户也可自由实现函数内容。

函数：LCD_bl_open

功能：打开 LCD 背光

原型：static void LCD_bl_open(__u32 sel)

显示驱动提供 LCD_PWM_EN 和 LCD_BL_EN 函数可供调用，用户也可自由实现函数内容。

LCD_PWM_EN，LCD_BL_EN，LCD_PWR_EN 这三个函数是通过 GPIO 控制实现电源和背光的开启关闭，IO 的位置及属性定义在 sys_config.fex 文件中。

函数：LCD_PWM_EN

功能：打开或关闭 LCD 背光调节的 PWM 信号

原型：void LCD_PWM_EN (__u32 sel, __bool b_en)

参数说明：

b_en=0：将 PWM pin 设为输入，并把 PWM 模块关闭

b_en=1：将 PWM pin 设为 PWM，并把 PWM 模块打开

对应于 sys_config.fex 文件的 lcd_pwm。

函数：LCD_BL_EN

功能：打开或关闭 LCD 背光

原型：void LCD_BL_EN (__u32 sel, __bool b_en)

参数说明：

b_en=0：设置 LCD 背光控制 IO 为对应电平，关闭背光

b_en=1：设置 LCD 背光控制 IO 为对应电平，打开背光

对应于 sys_config.fex 文件的 lcd_bl_en;

函数：LCD_PWR_EN

功能：打开或关闭 LCD 电源

原型：void LCD_PWR_EN(__u32 sel, __bool b_en)

参数说明：

b_en=0：设置 LCD 电源控制 IO 为对应电平，关闭 LCD 电源

b_en=1：设置 LCD 电源控制 IO 为对应电平，打开 LCD 电源

对应于 sys_config.fex 文件的 lcd_power。

关屏的步骤函数与开屏的步骤函数相对应。

2.2.2. 开关屏流程函数说明

函数：LCD_open_flow

功能：初始化开关屏的步骤流程

原型：static __s32 LCD_open_flow(__u32 sel)

函数常用内容为：

```
static __s32 LCD_open_flow(__u32 sel)
{
    LCD_OPEN_FUNC(sel, LCD_power_on,10);
    LCD_OPEN_FUNC(sel, LCD_panel_init, 50);
    LCD_OPEN_FUNC(sel, TCON_open, 100);
    LCD_OPEN_FUNC(sel, LCD_bl_open, 0);
    return 0;
}
```

如上，初始化整个开屏的流程步骤为四个：

1、打开 LCD 电源，再延迟 10ms；

2、初始化屏，再延迟 50ms；

3、打开 A10 TCON，再延迟 200ms；

4、打开背光，再延迟 0ms。

LCD_open_flow 函数只会系统初始化的时候调用一次，执行每个 LCD_OPEN_FUNC 即是把对应的开屏步骤函数进行注册，并没有执行该开屏步骤函数。LCD_open_flow 函数的内容必须统一用 LCD_OPEN_FUNC(sel, function, delay_time)进行函数注册的形式，确保正常注册到开屏步骤中。

函数：LCD_OPEN_FUNC

功能：注册开屏步骤函数到开屏流程中

原型：void LCD_OPEN_FUNC(__u32 sel, LCD_FUNC func, __u32 delay)

参数说明：

func 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 user_defined_func(__u32 sel)
{
    //do something
}
```

delay 是执行该步骤后，再延迟的时间，时间单位是毫秒。

2.3. 对屏的初始化

一部分屏需要进行初始化操作，在开屏步骤函数中，对应于 LCD_panel_init 函数。在 A10 中，提供了两种方式对屏的初始化。

一种是通过 8080 总线的方式，使用的是 LCDIO（PD,PH）。这种初始化方式，用于 CPU 屏中，其总线的引脚位置定义与 CPU 屏一致。

一种是 SPI 或 IIC 等串行的方式，使用的是 A10 的 GPIO 引脚模拟实现。模拟 GPIO 的引脚位置定义见于 sys_config.fex 中。

2.3.1. IO 模拟串行接口初始化

IO 模拟串行接口初始化可以参考附录 5.3.2 中的实例。

IO 的位置（PIN 脚）定义，默认属性（输入输出）定义及默认输出值在 sys_config.fex，具体请参考 2.5.2。

显示驱动提供 2 个接口函数可供使用。说明如下：

函数：LCD_GPIO_read

功能：读取 LCD_GPIO PIN 脚上的电平

原型：__s32 LCD_GPIO_read(__u32 sel,__u32 io_index);

参数说明：

io_index = 0：对应于 sys_config.fex 中的 lcd_gpio_0

io_index = 1：对应于 sys_config.fex 中的 lcd_gpio_1

io_index = 2：对应于 sys_config.fex 中的 lcd_gpio_2

io_index = 3：对应于 sys_config.fex 中的 lcd_gpio_3

函数返回值为对应 IO 的输入电平，只用于该 GPIO 定义为输入的情形。

函数：LCD_GPIO_write

功能：LCD_GPIO PIN 脚上输出高电平或低电平

原型：__s32 LCD_GPIO_write(__u32 sel,__u32 io_index, __u32 data);

参数说明：

io_index = 0：对应于 sys_config.fex 中的 lcd_gpio_0

io_index = 1：对应于 sys_config.fex 中的 lcd_gpio_1

io_index = 2：对应于 sys_config.fex 中的 lcd_gpio_2

io_index = 3：对应于 sys_config.fex 中的 lcd_gpio_3

data = 0：对应 IO 输出低电平

data = 1：对应 IO 输出高电平

只用于该 GPIO 定义为输出的情形。

函数：LCD_GPIO_set_attr

功能：设置 LCD_GPIO PIN 脚为输入或输出模式

原型：__s32 LCD_GPIO_set_attr(__u32 sel,__u32 io_index, __bool b_output);

参数说明：

io_index = 0：对应于 sys_config.fex 中的 lcd_gpio_0

io_index = 1：对应于 sys_config.fex 中的 lcd_gpio_1

io_index = 2：对应于 sys_config.fex 中的 lcd_gpio_2

io_index = 3：对应于 sys_config.fex 中的 lcd_gpio_3

b_output = 0：对应 IO 设置为输入

b_output = 1：对应 IO 设置为输出

2.3.2. CPU 屏 8080 总线初始化

CPU 屏的初始化可以参考“附录 5.3.5”的实例。

显示驱动提供 5 个接口函数可供使用。如下：

函数：LCD_CPU_WR

功能：设定 CPU 屏的指定寄存器为指定的值

原型：void LCD_CPU_WR(__u32 sel, __u32 index, __u32 data)

函数内容为

```
void LCD_CPU_WR(__u32 sel, __u32 index, __u32 data)
{
    LCD_CPU_WR_INDEX(sel, index);
    LCD_CPU_WR_DATA(sel, data);
}
```

实现了 8080 总线上的两个写操作。

LCD_CPU_WR_INDEX 实现第一个写操作，这时 PIN 脚 RS（A1）为低电平，总线数据上的数据内容为参数 index 的值。

LCD_CPU_WR_DATA 实现第二个写操作，这时 PIN 脚 RS（A1）为高电平，总线数据上的数据内容为参数 data 的值。

函数：LCD_CPU_WR_INDEX

功能：设定 CPU 屏为指定寄存器

原型：void LCD_CPU_WR(__u32 sel, __u32 index, __u32 data)

void LCD_CPU_WR_INDEX(__u32 sel,__u32 index);

具体说明见 LCD_CPU_WR。

函数：LCD_CPU_WR_DATA

功能：设定 CPU 屏寄存器的值为指定的值

原型：void LCD_CPU_WR_DATA(__u32 sel, __u32 data);

具体说明见 LCD_CPU_WR。

函数：LCD_CPU_AUTO_FLUSH

功能：开启 CPU 屏的刷新

原型：void LCD_CPU_AUTO_FLUSH(__u32 sel, __bool en);

参数说明：

en = 1：8080 总线上开始传送显示 BUFFER 的数据，实现 CPU 屏的刷新

函数：LCD_cpu_register_irq

功能：设置 LCD_GPIO PIN 脚为输入或输出模式

原型：void LCD_CPU_register_irq(__u32 sel, void (*Lcd_cpuisr_proc) (void))

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。

CPU 屏的初始化对应于开屏步骤函数的 LCD_panel_init。在 CPU 屏 LCD_panel_init 函数的最后，需要进行两个操作步骤：

1、使用 LCD_CPU_register_irq 注册 CPU 屏的中断处理函数 Lcd_cpuisr_proc，该函数的内容，可以是 CPU 屏 GRAM 的 X 和 Y 坐标设置为零的操作，以保证异步屏每帧进行一次同步。

2、调用 LCD_CPU_AUTO_FLUSH(sel,1)打开显示数据传送。

示例如下：

```
static void LCD_panel_init(__u32 sel)
{
    kgm281i0_init(sel);                                  //initial lcd panel
    kgm281i0_write_gram_origin(sel);                     //set gram origin
    LCD_CPU_register_irq(sel,Lcd_cpuisr_proc);           //resgister cpu irq func
    LCD_CPU_AUTO_FLUSH(sel,1);                           //start sent gram data
}
```

区别于模拟串行接口的初始化，LCD_open_flow 中，CPU 屏的初始化 LCD_panel_init 放在 TCON_open 之后，示例如下：

```
static __s32 LCD_open_flow(__u32 sel)
{
    LCD_OPEN_FUNC(sel, LCD_power_on,10);
    LCD_OPEN_FUNC(sel, TCON_open, 100);
    LCD_OPEN_FUNC(sel, LCD_panel_init, 50);
    LCD_OPEN_FUNC(sel, LCD_bl_open, 0);
    return 0;
}
```

2.4. 其它函数

2.4.1. 用户自定义函数

函数：LCD_user_defined_func

功能：用户可自由定义的函数

原型：static __s32 LCD_user_defined_func(__u32 sel, __u32 para1, __u32 para2, __u32 para3)

该函数是给用户作扩展使用的,可以在此函数里实现任何你想实现的代码,然后在应用层进行调用.

比如对于 3D 屏,可以在该文件里实现屏的 2D 和 3D 的切换,示例如下：

```
static __s32 LCD_user_defined_func(__u32 sel, __u32 para1, __u32 para2,
__u32 para3)
{
    if(para1 == 0)
    {
    //switch to 2D mode
    }
    else
    {
    //switch to 3D mode
    }
    return 0;
}
```

在用户空间的调用代码示例如下（切换到 3D 模式）：

```
unsigned long arg[4];
int dispfh;

if((dispfh = open("/dev/disp",O_RDWR)) == -1)
{
    printf("open file /dev/disp fail. \n");
    return 0;
}
arg[0] = 0;//lcd0
arg[1] = 1;//switch to 3D mode
ioctl(dispfh, DISP_CMD_LCD_USER_DEFINED_FUNC,(unsigned long)arg);
```

2.4.2. GPIO 操作函数

用户有可能有需要自己对某些 GPIO 进行操作,显示驱动封装了几个函数提供给用户,它们屏避了操作系统间的差异,也就是说在不同的操作系统中都可以使用。

函数：OSAL_GPIO_Request

功能：申请 GPIO;

原型：__hdle OSAL_GPIO_Request(user_gpio_set_t *gpio_list, __u32 group_count_max);

参数说明：

gpio_list 为 GPIO 的设置,该结构体如下:

```
typedef struct
{
    char    gpio_name[32];
    int port;
    int port_num;
    int mul_sel;
    int pull;
    int drv_level;
    int data;
}user_gpio_set_t;
```

group_count_max: 要设置 GPIO 的个数.

函数返回: 成功返回 GIPO 的句柄, 失败返回 0.

函数：OSAL_GPIO_Release

功能：释放 GPIO.

原型：__s32 OSAL_GPIO_Release(__hdle p_handler, __s32 if_release_to_default_status);

参数说明：

p_handler: GPIO 的句柄.

if_release_to_default_status: 0/1: 表示释放后的 GPIO 处于输入状态;2: 表示释放后的 GPIO 状态不变.

函数返回：成功返回 0, 失败返回错误号

将 GIPO PH6 输出高电平，示例如下：

```
static void LCD_vcc_on(__u32 sel)
{
    user_gpio_set_t gpio_list;
    int hdl;

    gpio_list.port = 8;// 1:A; 2:B; 3:C; 4:D;5:E;6:F;7:G;8:H.....
    gpio_list.port_num = 6;
    gpio_list.mul_sel = 1;
    gpio_list.pull = 0;
    gpio_list.drv_level = 0;
    gpio_list.data = 1;

    hdl = OSAL_GPIO_Request(&gpio_list, 1);
    OSAL_GPIO_Release(hdl, 2);
};
```

2.4.3. 延时函数

驱动提供了毫秒和微秒级的延时给用户使用，不过建议如果延时时间比较长的话可以在开关屏流程里新添新的函数. 因为在 boot 系统里延时是死等的,效率会比较低;如果放在开关屏流程里的话会启用 timmer 去做延时,在延时期间 CPU 可以做其它的工作.

函数：LCD_delay_ms

功能：延时 ms 毫秒

原型：void LCD_delay_ms(__u32 ms)

函数：LCD_delay_us

功能：延时 us 微秒

原型：void LCD_delay_us(__u32 us)

2.5. fex 文件

在 sys_config.fex 文件中，第一个参数定义了 LCD0 和 LCD1 的使能，定义如下：

lcd0_para_used/ lcd1_para_used

0: lcd0/lcd1 interface not exist;

1:lcd0/lcd1 interface exist;

其他定义为 LCD 所需的 A10 PIN 脚的分配情况。主要包括三个部分。

2.5.1. 电源控制 IO 的定义

电源控制 IO 在 sys_config.fex 文件中定义如下：

```
lcd_power       = port:PH08<1><0><default><0>
;lcd_bl_en      = port:PH07<1><0><default><1>
lcd_pwm         = port:PB02<2><0><default><default>
```

示例 1：lcd_power = port:PH08<1><0><default><1>

含义：lcd_power 引脚为 PH08，PH08 输出高电平时打开 LCD 供电；上下拉不使能。

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；default 表驱动能力是等级 1

第四个尖括号：输出有效所需电平；即是 LCD 供电电路上，lcd_power 使能所需的电平，0 为低电平，1 为高电平。

如果该功能引脚不需要，可以通过加注释的方式去掉。如上面代码的 lcd_bl_en。

lcd_bl_en

LCD_BL_EN pin config;

示例 2：lcd_bl_en = port:PH07<1><0><default><1>

含义：lcd_bl_en 引脚为 PH07，PH07 输出高电平时打开 LCD 背光；上下拉不使能。

lcd_bl_en 的定义与 lcd_power 一致。如果该功能引脚不需要，可以通过加注释的方式去掉。

示例 3：lcd_pwm = port:PB02<2><0><default><default>

含义：PB2 输出 PWM 信号，对应背光亮度大小的调节

A10 只有 PB2 和 PI3 可以输出 PWM 信号。如果该功能引脚不需要，可以通过加注释的方式去掉。

2.5.2. 模拟串行接口的 IO 定义

模拟串行接口 IO 在 fex 文件中定义如下：

```
lcd_gpio_0      = port:ph10<1><0><default><1>
lcd_gpio_1      = port:ph11<1><0><default><1>
;lcd_gpio_2     =
;lcd_gpio_3     =
```

示例：lcd_gpio_0 = port:ph10<1><0><default><1>

含义：lcd_gpio_0 引脚为 PH10，PH10 为输出，默认输出电平为高。

第一个尖括号：功能分配；0 为输入，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；default 表驱动能力是等级 1

第四个尖括号：表示默认值；即是当设置为输出时，该引脚输出的电平，0 为低电平，1 为高电平。

如果该功能 IO 不用，可通过注释方式（如上的 lcd_gpio_2）避免冲突，显示驱动对注释 IO 不进行初始化操作。

2.5.3. LCD IO 定义

LCD0 的 IO 定义如下，若该引脚 IO 需要应用于其他方面，可通过注释方式（如下的 lcdde）避免冲突，显示驱动对注释 IO 不进行初始化操作。

LCD1 的 IO 定义与 LCD0 相似，不同的地方在于 LCD1 为 PH 口。

```
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>
```

3. TCON 参数说明

3.1. 接口参数说明

3.1.1. lcd_if

Lcd Interface

设置相应值的对应含义为：

0：HV(RGB 同步屏)接口

1：CPU(8080)接口

2：Reserved

3：LVDS 接口

3.1.2. lcd_hv_if

Lcd HV panel Interface

这个参数只有在 lcd_if=0 时才有效。定义 RGB 同步屏下的几种接口类型。

设置相应值的对应含义为：

0：Parallel RGB

1：Serial RGB

2：Serial YUV (CCIR656)

RGB 同步屏的接口类型可参考“附录 5.1.1 HV RGB 同步屏接口”。

3.1.3. lcd_hv_srgb_seq0

Lcd HV panel Serial RGB output Sequence in Odd lines

这个参数只有在 lcd_if=0 且 lcd_hv_if=1（Serial RGB）时才有效。

定义奇数行 RGB 输出的顺序

0: R→G→B

1: B→R→G

2: G→B→R

3.1.4. lcd_hv_srgb_seq1

Lcd HV panel Serial RGB output Sequence in Even lines

这个参数只有在 lcd_if=0 且 lcd_hv_if=1（Serial RGB）时才有效。

定义偶数行 RGB 输出的顺序

0: R→G→B

1: B→R→G

2: G→B→R

3.1.5. lcd_hv_syuv_seq

Lcd HV panel Serial YUV output Sequence

这个参数只有在 lcd_if=0 且 lcd_hv_if=2（Serial YUV）时才有效。

定义 YUV 输出格式

0：YUYV

1：YVYU

2：UYVY

3：VYUY

3.1.6. lcd_hv_syuv_fdly

Lcd HV panel Serial YUV F line Delay

这个参数只有在 lcd_if=0 且 lcd_hv_if=2（Serial YUV）时才有效。

定义 CCIR656 编码时 F 相对有效行延迟的行数

0：F toggle right after active video line

1：Delay 2 lines (CCIR NTSC)

2：Delay 3 lines (CCIR PAL)

3.1.7. lcd_cpu_if

Lcd CPU panel Interface

这个参数只有在 lcd_if=1 时才有效。

设置相应值的对应含义为：

0：18bit/1cycle parallel (RGB666)

4：16bit/1cycle parallel (RGB565)

6：18bit/3cycle parallel (RGB666)

7：16bit/2cycle parallel (RGB565)

CPU 屏/8080 的接口类型可参考“附录 5.1.2 CPU/8080 屏接口”。

3.1.8. lcd_lvds_ch

Lcd LVDS panel Channel number

设置相应值的对应含义为：

0：Single Link

1：Dual Link

相关说明可参见“附录 5.1.3 LVDS 屏接口”。

3.1.9. lcd_lvds_bitwidth

Lcd LVDS panel Bit Width

设置相应值对应含义为：

0：24bit

1：18bit

相关说明可参见“附录 5.1.3 LVDS 屏接口”。

3.1.10. lcd_lvds_mode

Lcd LVDS Mode

这个参数只有在 lcd_lvds_bitwidth=0 时才有效

设置相应值对应含义为：

0：NS mode

1：JEIDA mode

NS mode 和 JEIDA mode 的说明可参见“附录 5.1.3 LVDS 屏接口”。

3.1.11. lcd_frm

Lcd Frame Rate Modulator

FRM 是解决 PIN 减少导致的色深问题。

这个参数设置相应值对应含义为：

0：RGB888 → RGB888 direct

1：RGB888 → RGB666 dither

2：RGB888 → RGB565 dither

补充说明：对接口参数的配置可以参考表 3-1。确定好接口类型，最后确保对应的一行的相关参数都有配置。

表 3-1 LCD 接口参数配置说明

Interface		Parameter
同步 RGB 接口 (lcd_if=0)	Parallel RGB (lcd_hv_if=0)	
	Serial RGB (lcd_hv_if=1)	lcd_hv_srgb_seq0 lcd_hv_srgb_seq1
	CCIR656 (lcd_hv_if=2)	lcd_hv_syuv_seq lcd_hv_syuv_fdly
CPU/8080 接口 (lcd_if=1)	Parallel RGB666 (lcd_cpu_if=0)	lcd_frm=1
	Parallel RGB565 (lcd_cpu_if=4)	lcd_frm=2
	Serial RGB666 (lcd_cpu_if=5)	
	8bit2cycle serial (lcd_cpu_if=7)	
LVDS 接口 (lcd_if=3)	Single channal (lcd_lvds_ch=0)	24bit: lcd_lvds_bitwidth=0 lcd_lvds_mode 18bit: lcd_lvds_bitwidth=1 lcd_frm=1
	Dual channal (lcd_lvds_ch=1)	

注：请先确定好接口类型，最后确保对应的那一行的参数都有配置

3.2. 时序参数说明

3.2.1. lcd_x

显示屏宽的像素个数

3.2.2. lcd_y

显示屏高的像素个数

3.2.3. lcd_ht

Horizontal Total time

指一行总的 dclk 的 cycle 个数。见图 3-1。

图 3-1 水平方向时序信号图

3.2.4. lcd_hbp

Horizontal Back Porch

指有效行间，行同步信号（hsync）开始，到有效数据开始之间的 dclk 的 cycle 个数。见图 3-1。

3.2.5. lcd_vt

Vertical Total time

指两场的总行数。见图 2。

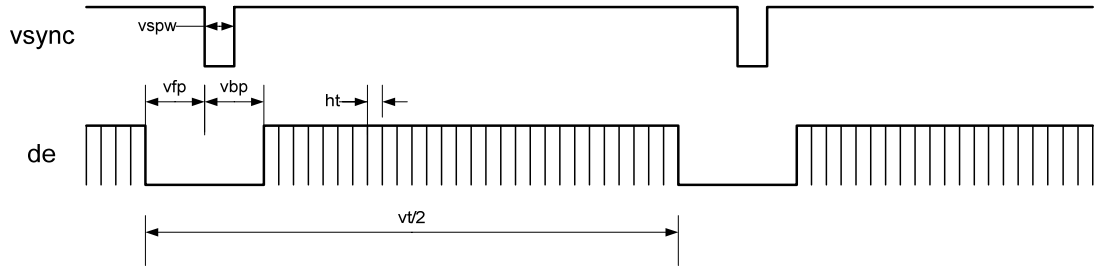


图 3-2 垂直方向时序信号图

3.2.6. lcd_vbp

Vertical Back Porch

指场同步信号（vsync）开始，到有效数据行开始之间的行数。见图 3-2。

3.2.7. lcd_hv_hspw

Horizontal Sync Pulse Width

指行同步信号的宽度。单位为 1 个 dclk 的时间（即是 1 个 data cycle 的时间）。

见图 3-1。

3.2.8. lcd_hv_vspw

Vertical Sync Pulse Width

指场同步信号的宽度。单位为行。见图 3-2

3.2.9. lcd_dclk_freq

Data Clock Frequency

指 PIN 总线上数据的传送频率。单位为 MHz

屏幕刷新帧数 = (dclk_freq) / (ht×vt/2)

3.2.10. lcd_io_cfg0

Lcd IO Configuration0

这个参数提供 RGB 同步屏的相位调节。

lcd_dclk_freq < 40 时，该参数可设置为 0x00000000，0x04000000，0x10000000，0x14000000，0x20000000，0x24000000，对应 LCD DCLK 的六个不同相位。

lcd_dclk_freq > 40 时，该参数可设置为 0x00000000，0x04000000 对应 LCD DCLK 的两个不同相位。

补充说明 1：hbp 在部分屏规格书的定义里并不包括 hspw。这种情况下，要正确配置 AW 的 TCON， hbp(aw)=hbp(panel)+hspw(panel)。vbp 的定义同 hbp。

补充说明 2：A10 的 TCON 中，hfp，vfp 不能为 0。

3.3. 其他参数说明

3.3.1. lcd_pwm_not_used

LCD PWM not Used

通过占空比的不同，PWM 在 LCD 中常用于调节背光的亮度变化。这个参数设置相应值的对应含义为：

0：不使用 PWM

1：使用 PWM

3.3.2. lcd_pwm_ch

Lcd backlight PWM Chanel

该参数值只有在 lcd_pwm_not_used=0 时才有效。设置相应值的对应含义为：

0：使用 PWM0，从 PB2 送出

1：使用 PWM1，从 PI3 送出

sys_config.fex 的 lcd_pwm 要相应进行改动。注意，该参数与 lcd_pwm_freq lcd_pwm_pol 共同构成一路 PWM 信号的设置。

3.3.3. lcd_pwm_freq

Lcd backlight PWM Frequency

该参数值只有在 lcd_pwm_not_used=0 时才有效。这个参数配置 PWM 信号的频率，单位为 Hz。

3.3.4. lcd_pwm_pol

Lcd backlight PWM Polarity

该参数值只有在 lcd_pwm_not_used=0 时才有效。这个参数配置 PWM 信号的占空比的极性。

3.3.5. lcd_gamma_correction_en

Lcd Gamma Correction Enable

设置相应值的对应含义为：

0：TCON 的 Gamma 校正关闭

1：TCON 的 Gamma 校正打开

设置为 1 时，需要对 lcd_gamma_tbl [256]进行赋值。

3.3.6. lcd_gamma_tbl

Lcd Gamma Table

该参数为一个数组__u32 lcd_gamma_tbl[256]；

lcd_gamma_tbl[n] = rout<<16 | gout<<8 | bout<<0 表示：输入 r=n 时，输出 r=rout；输入 g=n 时，输出 g=gout；输入 b=n 时，输出 b=bout。

用户使用 Gamma 校正功能时，可以使用函数 lcd_gamma_gen(__panel_para_t * info)对其赋值，函数内容可自由实现。具体可参考“5.3.1”中的实例。

4. 操作指南

显示驱动按照操作系统和运行阶段不同分为两个部分:boot 显示驱动和 linux 显示驱动,因此 LCD 配置也对应两个部分.

配置 LCD 需要用户配置的最多有 5 个文件,用户可能会修改其中的一个或几个文件:

1,sys_config1.fex(boot 和 linux 阶段都会使用该文件的配置)

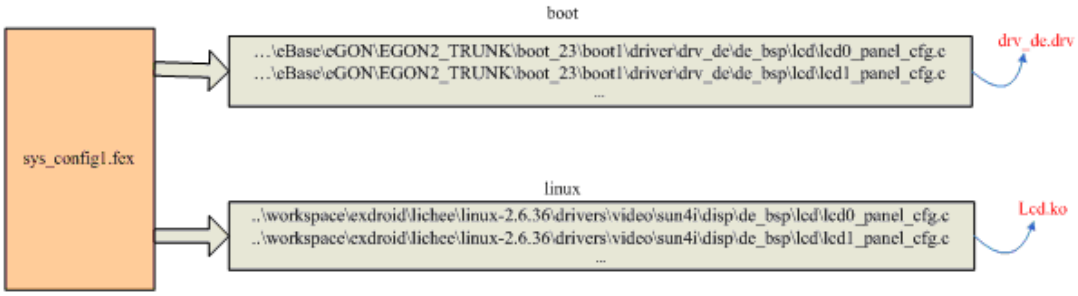
2,boot 阶段屏 0 配置文件: lcd0_panel_cfg.c

3,boot 阶段屏 1 配置文件: lcd1_panel_cfg.c

4,linux 阶段屏 0 配置文件: lcd0_panel_cfg.c

5,linux 阶段屏 1 配置文件: lcd1_panel_cfg.c

如下图所示:



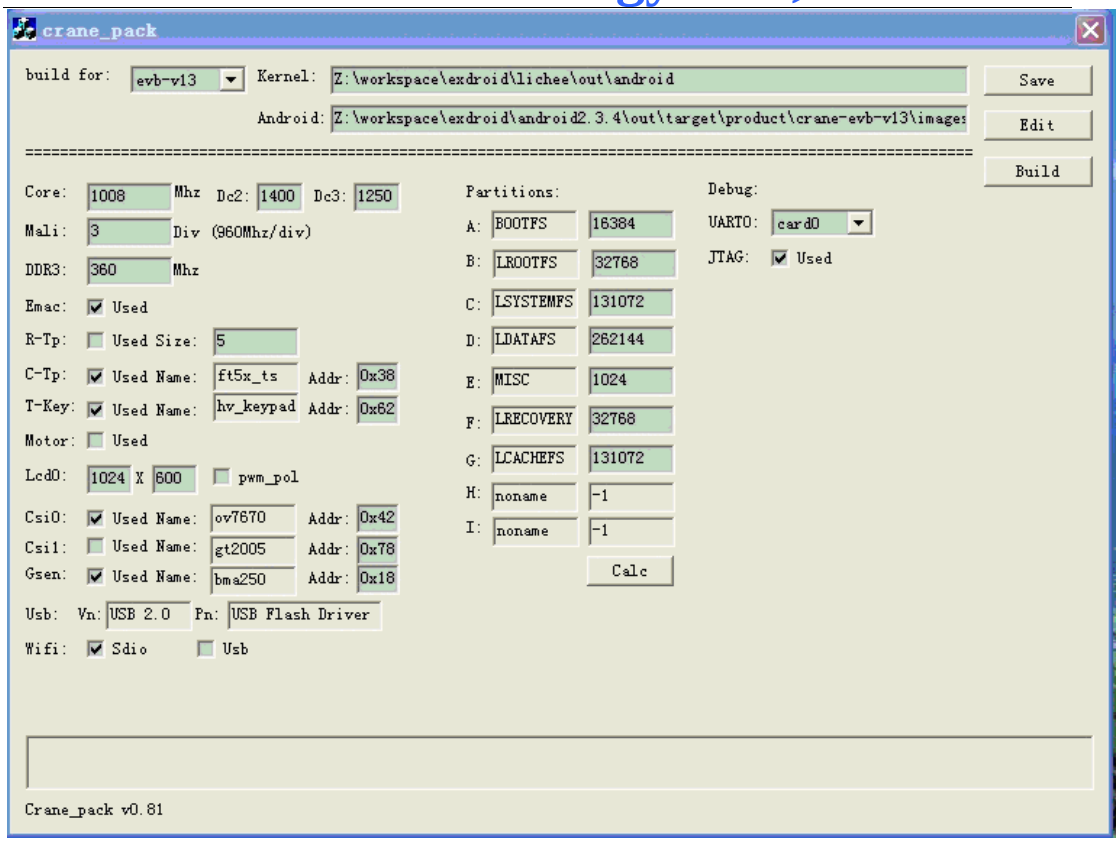
注意,我们尽量要求 boot 和 linux 阶段的对应两个文件内容完全相同,与操作系统相关的操作都会以一个封装好的,具有统一接口的形式提供.所以当一个系统的.c 文件调通以后,可以将对应的文件拷贝到另一个系统的目录里进行编译即可.

为了文件描述,这篇文档假定我们 linux 上述的目录为"..\workspace\exdroid\lichee\linux-2.6.36\",具体情况可能会与之不同, 将其改为你对应目录即可.

4.1. sys_config1.fex 配置

打开打包目录里的工具"crane_pack.exe", 选择好平台和路径后,点"Edit"编辑 sys_config1.fex 里的 LCD 相关配置,保存后点"Build"重新生成新的固件.

如下图所示:



4.2. boot 阶段 LCD 配置

eBase 代码下载路径:

B 网: svn://172.16.1.11/eBase

D 网: svn://192.168.200.11/eBase

C 文件路径:

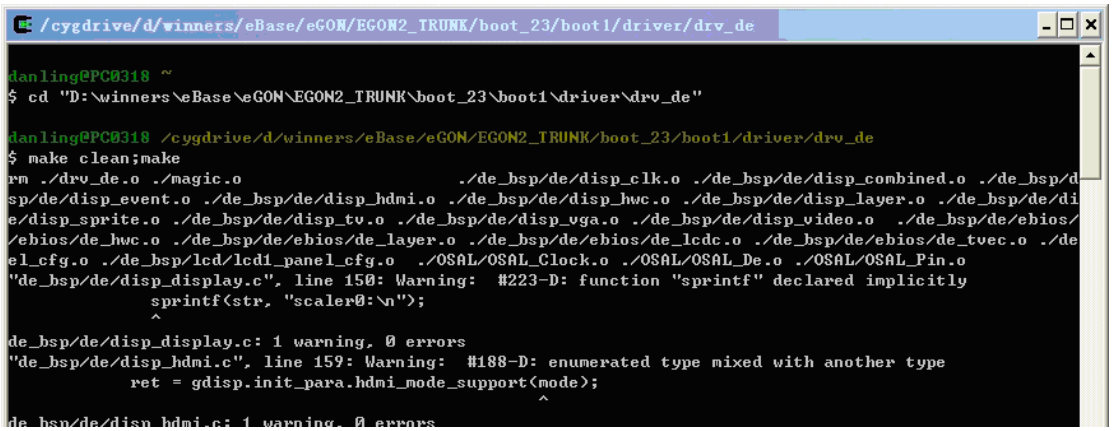
…\eBase\eGON\EGON2_TRUNK\boot_23\boot1\driver\drv_de\de_bsp\lcd\lcd0_panel_cfg.c

…\eBase\eGON\EGON2_TRUNK\boot_23\boot1\driver\drv_de\de_bsp\lcd\lcd1_panel_cfg.c

编译:

打开 cygwin，进入 drv_de 目录，然后调入命令”make clean;make”即可.

如下图所示。



编译出来的结果是 drv_de.drv,路径如下:

…\eBase\eGON\EGON2_TRUNK\boot_23\workspace\wboot\bootfs\drv_de.drv.

更新驱动：

如果要重新产生固件,可以将文件 drv_de.drv 文件拷贝到打包目录对应的地方,然后使用“crane_pack.exe”，点“Build”重新生成固件，再烧写。

路径：…\wboot\bootfs\drv_de.drv

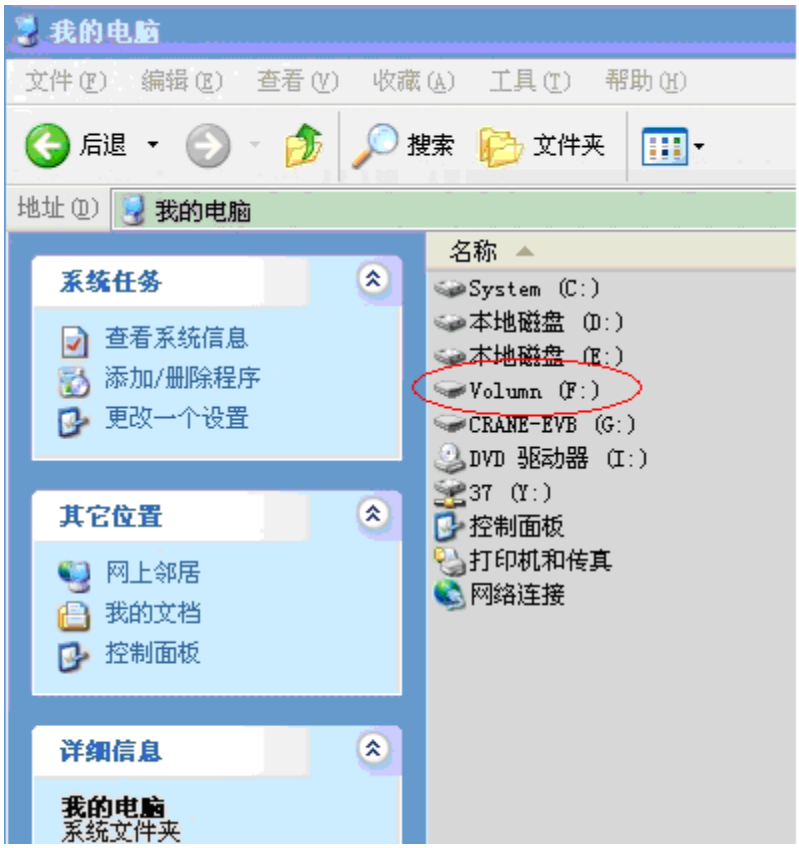
如果不想重新产生固件,可以直接将文件 drv_de.drv 拷贝到小机里。

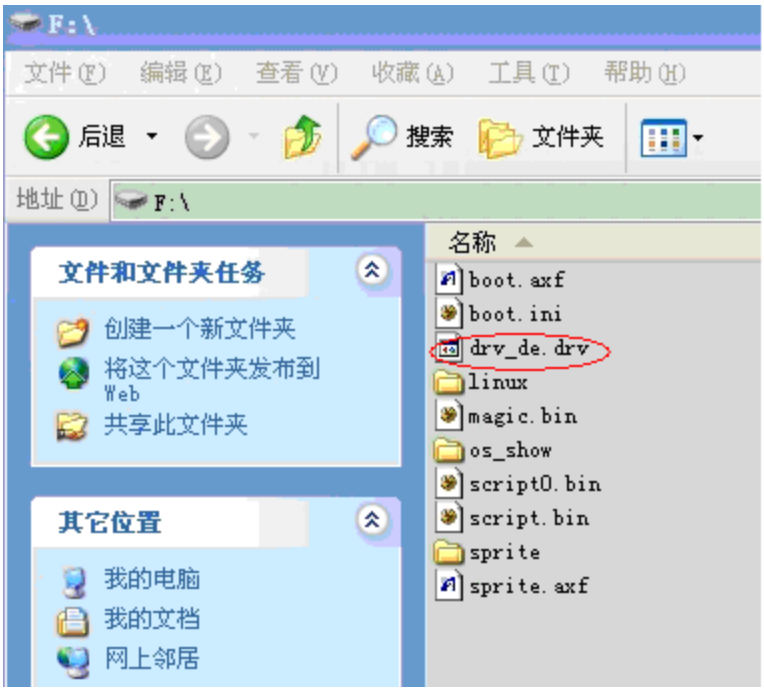
方法是先断电,接上串口，接上 USB 线，按住 PC 键盘的”1”然后开机，会进入 U 盘模式。

下图为串口打印信息：

```
HELLO! BOOT0 is starting!
boot0 version : 1.1.3
dram size =512
Succeed in opening nand flash.
Succeed in reading Boot1 file head.
The size of Boot1 is 0x00030000.
The file stored in 0X00000000 of block 2 is perfect.
Check is correct.
Ready to disable icache.
Succeed in loading Boot1.
Jump to Boot1.
[       0.161574] boot1 version : 1.1.5
[       0.161734] pmu type = 3
[       0.181853] axi:ahb:apb=2:1:1
[       0.181871] set dcdc2=1400, clock=1008 successed
[       0.184102] key
[       0.195942] no key found
[       0.195957] flash init start
[       0.216844] flash init finish
[       0.218536] fs init ok
[       0.219319] fs mount ok
[       0.223351] script finish
[       0.224262] power finish
[       0.228377] BootMain start
[       0.228403] 49
[       0.229342] part count = 2
[       0.232151] USB Device!!
[       0.234916] USB Connect!!
[       0.286088] uSuspend
```

再将 drv_de.drv 文件拷贝进名称为”Volumn”的磁盘里,如下图的 F 盘.





最后重启系统即可.

4.3. linux 阶段 LCD 配置

C 文件路径：

..\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i\disp\de_bsp\lcd\lcd0_panel_cfg.c

..\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i\disp\de_bsp\lcd\lcd1_panel_cfg.c

编译：

打开 putty，进入目录..\workspace\exdroid\lichee，再运行命令”./build.sh -p sun4i_crane”.

编译出来我们所需要的结果是 lcd.ko,路径如下:

..\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i\lcd\lcd.ko

更新驱动：

如果需要更新固件,进入目录” ..\workspace\exdroid\”,运行命令”mkimg”,然后使用”crane_pack.exe”,点”Build”重新生成固件,再烧写..

如果不需要更新固件,可以使用 adb 将 lcd.ko 拷贝到小机里.

方法是在进入 android 系统后，运行批处理文

件”..\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i\adb_push.bat”，它会将 disp.ko/hdmi.ko/lcd.ko 三个文件 push 到小机对应的目录(为什么要更新三个驱动而不是只更新 lcd.ko,是考虑到这三个文件有一定的关联,如果版本不一致的话容易出现一些问题,因此将三个文件一起更新以减少因版本不同而产生问题的机会,用户也可自行改写批处理只更新 lcd.ko 文件),然后重启系统即可.

批处理命令如下:

```
adb devices
adb shell mount -o remount,rw /dev/block/nandc /system
adb shell mount -o remount,rw /dev/root /
adb push disp/disp.ko /drv/disp.ko
adb shell chmod 777 /drv/disp.ko
adb shell sync
adb push lcd/lcd.ko /drv/lcd.ko
adb shell chmod 777 /drv/lcd.ko
adb shell sync
adb push hdmi/hdmi.ko /drv/hdmi.ko
adb shell chmod 777 /drv/hdmi.ko
adb shell sync
pause
```

运行批处理的界面如下:

```
Z:\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i>adb devices
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
List of devices attached
20080411        device


Z:\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i>adb shell mount -o remount,rw /dev/block/nandc /system

Z:\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i>adb shell mount -o remount,rw /dev/root /

Z:\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i>adb push disp/disp.ko /drv/disp.ko
4170 KB/s (3270044 bytes in 0.765s)

Z:\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i>adb shell chmod 777 /drv/disp.ko

Z:\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i>adb shell sync

Z:\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i>adb push lcd/lcd.ko /drv/lcd.ko
3594 KB/s (230072 bytes in 0.062s)

Z:\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i>adb shell chmod 777 /drv/lcd.ko

Z:\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i>adb shell sync

Z:\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i>adb push hdmi/hdmi.ko /drv/hdmi.ko
3564 KB/s (456303 bytes in 0.125s)

Z:\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i>adb shell chmod 777 /drv/hdmi.ko

Z:\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i>adb shell sync

Z:\workspace\exdroid\lichee\linux-2.6.36\drivers\video\sun4i>pause
请按任意键继续. . .
```

4.4. 调试

1、如果由于某些原因无串口打印, 可以用 adb 进行打印.

命令是: adb shell cat /proc/kmsg

2、如果你想先调试 linux 阶段的 LCD 显示,可以将 drv_de.drv 删除掉再进行调试,以免 boot 阶段的错误设置影响 llinux 阶段的显示.

5. 附录

5.1. 屏接口说明

5.1.1. HV RGB 同步屏接口

RGB 同步屏接口分为 Parallel RGB，Serial RGB，CCIR656 三种类型接口。常见的一些屏可同时支持 Serial RGB，CCIR656 两种，引脚定义一致。

图 5-1，图 5-2 是 RGB 同步屏的模组规格书的引脚，可供参考。

Pin No.	Symbol	I/O	Function
1	V_{LED-}	P	Power for LED backlight cathode
2	V_{LED+}	P	Power for LED backlight anode
3	GND	P	Power ground
4	V_{DD}	P	Power voltage
5	R0	I	Red data (LSB)
6	R1	I	Red data
7	R2	I	Red data
8	R3	I	Red data
9	R4	I	Red data
10	R5	I	Red data
11	R6	I	Red data
12	R7	I	Red data (MSB)
13	G0	I	Green data (LSB)
14	G1	I	Green data
15	G2	I	Green data
16	G3	I	Green data
17	G4	I	Green data
18	G5	I	Green data
19	G6	I	Green data
20	G7	I	Green data (MSB)
21	B0	I	Blue data (LSB)
22	B1	I	Blue data
23	B2	I	Blue data
24	B3	I	Blue data
25	B4	I	Blue data
26	B5	I	Blue data
27	B6	I	Blue data
28	B7	I	Blue data (MSB)
29	DGND	I	Digital ground
30	DCLK	I	Pixel clock
31	DISP	I	Display on/ off
32	HSYNC	I	Horizontal sync signal
33	VSYNC	I	Vertical sync signal
34	DE	I	Data enable
35	NC	-	No Connect
36	GND	P	Power ground
37	X1	I/O	Right electrode - differential analog
38	Y1	I/O	Bottom electrode - differential analog
39	X2	I/O	Left electrode - differential analog
40	Y2	I/O	Top electrode - differential analog

I/O: I: input, O: output, P: power

并行数据信号

同步控制信号

图 5-1 HV Parallel RGB 引脚参考图

PIN NO.	Symbol	Description
1	VCOM	Common electrode driving voltage
2	CS（SPENB）	Serial communication chip select
3	SDA（SPDA）	ial command data input
4	SCL（SPCK）	Serial command clock input
5	HSYNC（HSD）	Horizontal sync input
6	VSYNC（VSD）	Vertical sync input
7	DCLK（CLKIN）	Data clock input
8~15	D7~D0	Data bus
16	DGND	Ground for digital circuits.
17	VDD	Power setting
18	VDDIO	Power supply for digital interface
19	DVDD	Power setting capacitor connecting pins.
20	C1P	Capacitor connect pin for internal charge pump.
21	C1M	Capacitor connect pin for internal charge pump.
22	C2P	Capacitor connect pin for internal charge pump.
23	C2M	Capacitor connect pin for internal charge pump.
24	VINT1	Power setting capacitor connect pin.
25	C3P	Capacitor connect pin for internal charge pump.
26	C3M	Capacitor connect pin for internal charge pump.
27	VINT2	Power setting capacitor connect pin.
28	VINT3	Power setting capacitor connect pin.
29	C4P	Capacitor connect pin for internal charge pump.
30	C4M	Capacitor connect pin for internal charge pump.
31	VGH	Power setting capacitor connect pin

初始化串行接口

同步控制信号

串行数据信号

图 5-2 HV Serial RGB/CCIR656 引脚参考图

5.1.2. CPU/8080 屏接口

CPU 屏接口分为 Parallel RGB666（并行 18bit），Parallel RGB565（并行 16bit），Serial RGB666，Serial RGB565 四种类型。

图 5-3 是并行 18bitCPU 屏的模组规格书的引脚，可供参考。

PIN NO.	SYMBOL	DESCRIPTION
1	Y（U）	Touch panel YU
2	X（L）	Touch panel XL
3	Y（D）	Touch panel YD
4	X（R）	Touch panel XR
5	VSS	Ground
6	IOVCC	Connect to VCC .
7	VCC	Internal logic power: VCC=2.5V~3.3V, VCC>IOVCC.
8	FMARK	NC.
9	CS	Chip select
10	RS	Register select input pin RS = "H": DB0 to DB7 are display data RS = "L": DB0 to 读写控制信号 rol data
11	WR	Read/Write execution control R/W=" H ": read R/W=" L": write
12	RD	Read/Write execution control When /RD is " L ", D0 to D7 are in an output status.
13-30	DB0-DB17	18bit数据信号
31	IM1	NC
32	IM0	NC
33	ID	NC
34	RESET	This is an active low signal.
35	GND	Internal logic GND: GND=0V.
36	LED-A	LED 3.2V
37	LED-K1	LED1 GND
38	LED-K2	LED2 GND
39	LED-K3	LED3 GND
40	LED-K4	LED4 GND

图 5-3 CPU Parallel RGB666 引脚参考图

5.1.3. LVDS 屏接口

LVDS 屏接口分为 Single Link 和 Dual Link 两种接口。

在图 5-4 中，红色部分和蓝色部分分别对应独立的 2 个 link。若只有红色部分，即为 Single Link；若有红色和蓝色两部分，即为 Dual Link。

Single Link 和 Dual Link 都具有 24bit，18bit 两种色深。

在图 5-4 中，实线部分包含 3 data pair channel，虚线部分包含 1 对 data channel。在每一个 Link 中，若只有实线部分，即只有 3 data pair channel，即为 18bit，若具有实线和虚线共 4 data pair channel，即为 24bit。

NS mode 和 JEIDA mode 两种模式的定义见图 5-5。

Pin	Symbol	Description
1	GND	Ground
2	GND	Ground
3	AVDD	Power Supply, 3.3V Typ.
4	AVDD	Power Supply, 3.3V Typ.
5	AVDD	Power Supply, 3.3V Typ.
6	DVDD	Digital Power supply (3.3V Typ)
7	DVDD	Digital Power supply (3.3V Typ)
8	Clk EEDID	Two wire serial interface clock
9	DATA EEDID	Two wire serial interface data
10	RXinO0-	- LVDS differential data input, Chan 0-Odd
11	RXinO0+	+ LVDS differential data input, Chan 0-Odd
12	GND	Ground
13	RXinO1-	- LVDS differential data input, Chan 1-Odd
14	RXinO1+	+ LVDS differential data input, Chan 1-Odd
15	GND	Ground
16	RXinO2-	- LVDS differential data input, Chan 2-Odd
17	RXinO2+	+ LVDS differential data input, Chan 2-Odd
18	GND	Ground
19	RXOC-	- LVDS Differential Clock input (Odd)
20	RXOC+	+ LVDS Differential Clock input (Odd)
21	GND	Ground
22	RXinO3-	- LVDS differential data input, Chan 3-Odd
23	RXinO3+	+ LVDS differential data input, Chan 3-Odd
24	GND	Ground
25	RXinE0-	- LVDS differential data input, Chan 0-Even
26	RXinE0+	+ LVDS differential data input, Chan 0-Even
27	GND	Ground
28	RXinE1-	- LVDS differential data input, Chan 1-Even
29	RXinE1+	+ LVDS differential data input, Chan 1-Even
30	GND	Ground
31	RXinE2-	- LVDS differential data input, Chan 2-Even
32	RXinE2+	+ LVDS differential data input, Chan 2-Even
33	GND	Ground
34	RXEC-	- LVDS Differential Clock input (Even)
35	RXEC+	+ LVDS Differential Clock input (Even)
36	GND	Ground
37	RXinE3-	- LVDS differential data input, Chan 3-Even
38	RXinE3+	+ LVDS differential data input, Chan 3-Even
39	GND	Ground
40	NC	No connection

图 5-4 Dual Link 24bit LVDS

5.2. A10 与屏的连接说明

两组 LCD 口在 A10 中分配在 PD 和 PH 口。每组 LCD IO 口包括 4 个控制 IO（PD24、PD25、PD26、D27）和 24 个数据 IO（PD0-PD23）。

配置为不同的显示屏接口，IO 的定义不同，具体的定义见图 5-1，不同接口的屏与 A10 的 LCD IO 的连接也应参照图 5-1 进行。

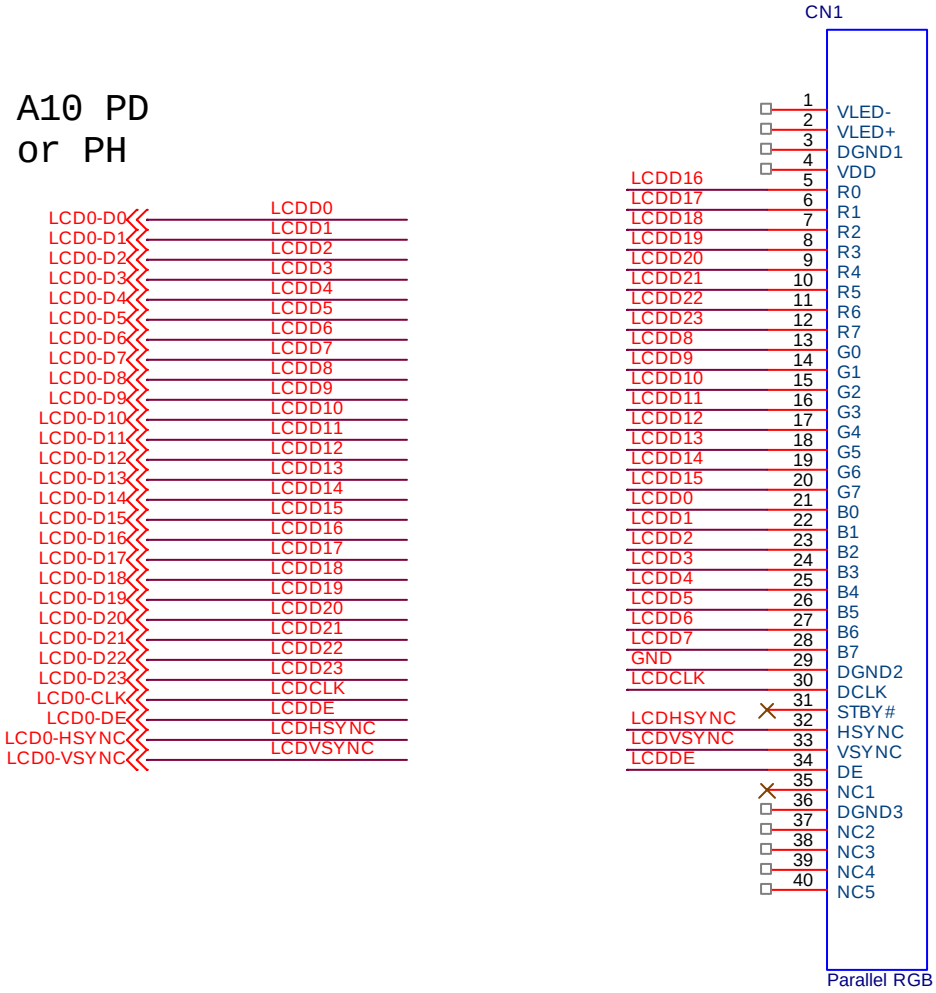
5.2.2 – 5.2.6 是常见的屏接口与 A10 LCD IO 的连接参考图。

5.2.1. LCD IO PORT 定义

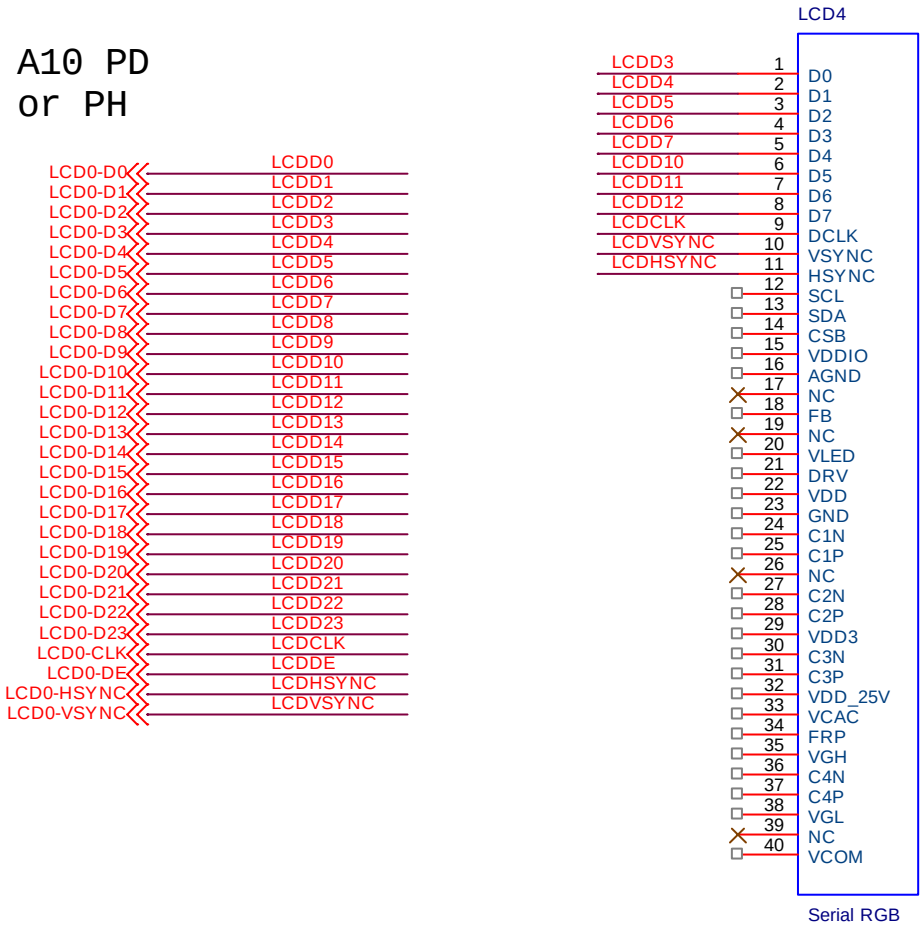
表 5-1 LCD IO PORT 定义

If	同步 RGB 接口					CPU/I80 接口						LVDS 接口	
	Para RGB	Serial RGB			CCIR 656	Para RGB 666	Para RGB 565	Serial RGB 666		Serial RGB 565		Sing Link	Dual Link
Cycle		1st	2nd	3rd				1st	2nd	1st	2nd		
IO0	VSYNC					CS							
IO1	HSYNC					RD							
IO2	DCLK					WR							
IO3	DE					RS							
D23	B7					R5	R4						
D22	B6					R4	R3						
D21	B5					R3	R2						
D20	B4					R2	R1						
D19	B3					R1	R0					1-VN3	E-VN3
D18	B2					R0	G5					1-VP3	E-VP3
D17	B1											1-VNC	E-VNC
D16	B0											1-VPC	E-VPC
D15	G7					G5	G4					1-VN2	E-VN2
D14	G6					G4	G3					1-VP2	E-VP2
D13	G5					G3						1-VN1	E-VN1
D12	G4	D17	D27	D37	D7	G2	G2	R5	G2	R4	G2	1-VP1	E-VP1
D11	G3	D16	D26	D36	D6	G1	G1	R4	G1	R3	G1	1-VN0	E-VN0
D10	G2	D15	D25	D35	D5	G0	G0	R3	G0	R2	G0	1-VP0	E-VP0
D9	G1											0-VN3	O-VN3
D8	G0											0-VP3	O-VP3
D7	B7	D14	D24	D34	D4	B5	B4	R2	B5	R1	B4	0-VNC	O-VNC
D6	B6	D13	D23	D33	D3	B4	B3	R1	B4	R0	B3	0-VPC	O-VPC
D5	B5	D12	D22	D32	D2	B3	B2	R0	B3	G5	B2	0-VN2	O-VN2
D4	B4	D11	D21	D31	D1	B2	B1	G5	B2	G4	B1	0-VP2	O-VP2
D3	B3	D10	D20	D30	D0	B1	B0	G4	B1	G3	B0	0-VN1	O-VN1
D2	B2					B0		G3	B0			0-VP1	O-VP1
D1	B1											0-VN0	O-VN0
D0	B0											0-VP0	O-VP0

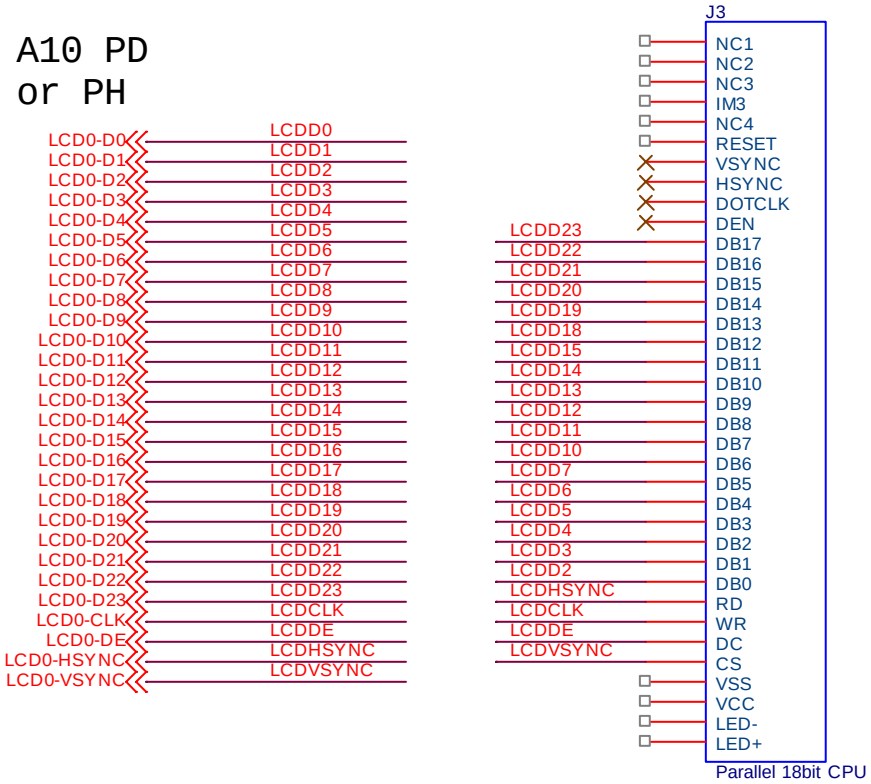
5.2.2. HV Parallel RGB 屏参考连接图



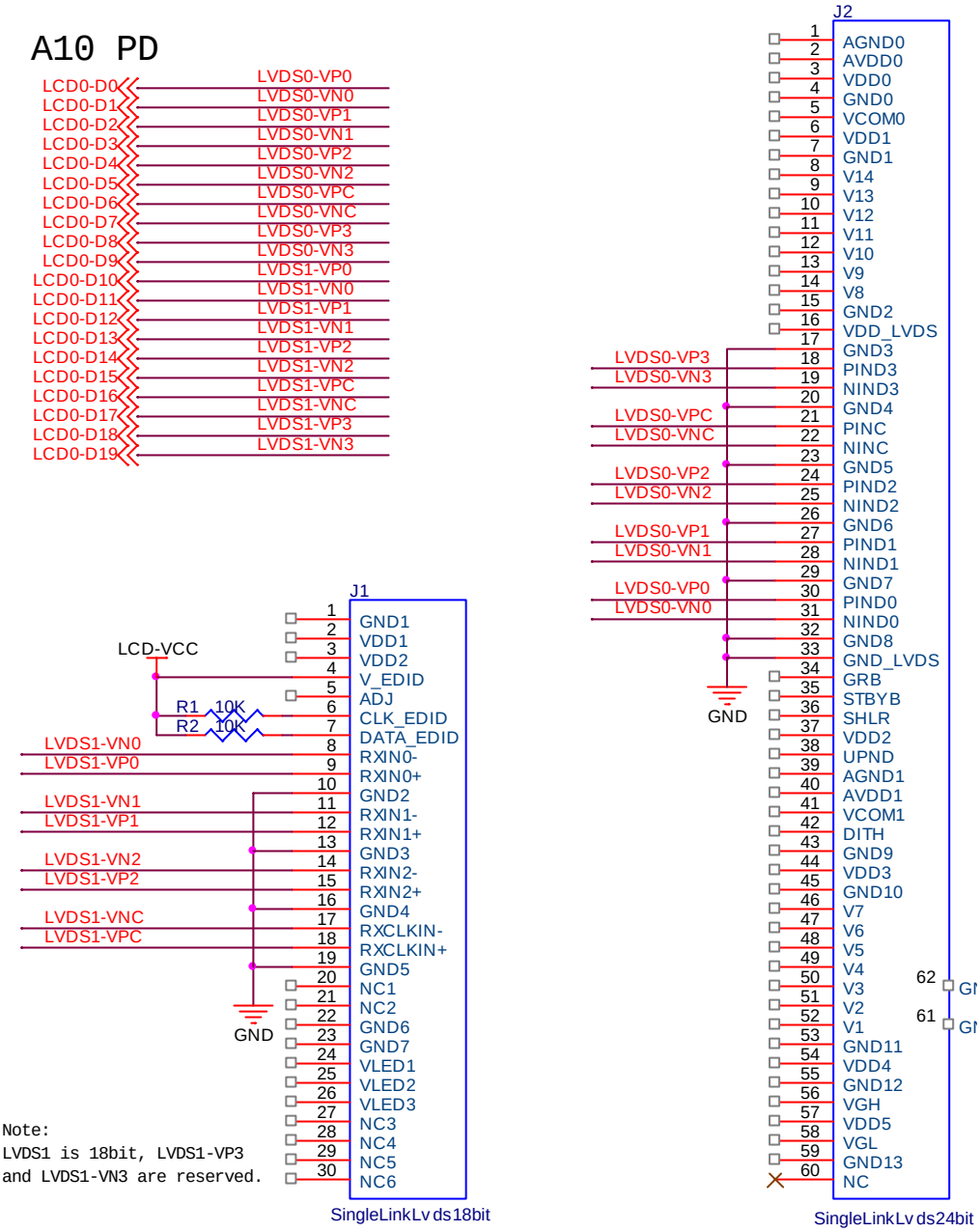
5.2.3. HV Serial RGB 屏参考连接图



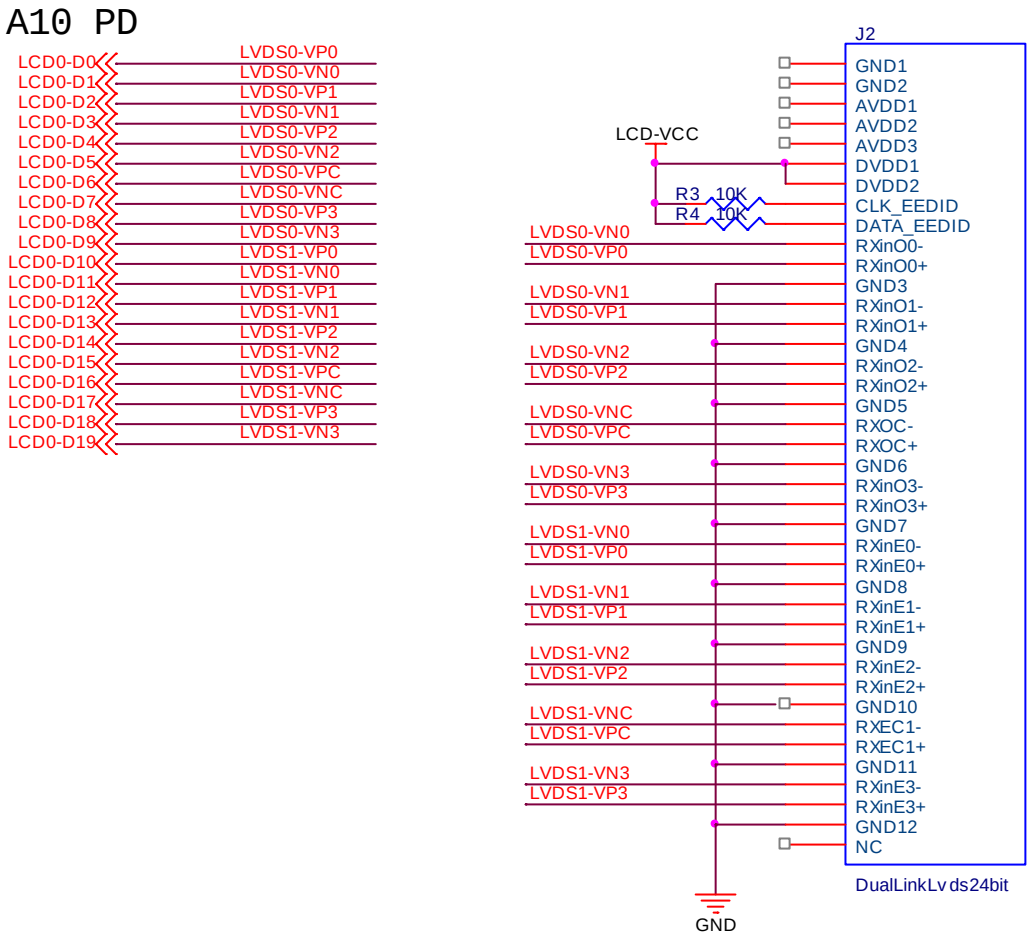
5.2.4. CPU Parallel RGB666 屏参考连接图



5.2.5. LVDS 2 Single Link 屏参考连接图



5.2.6. LVDS Dual Link 屏参考连接图



5.3. 屏文件实例

5.3.1 是 A10-EVB-1.1 开发板上的 sys_config1.fex 的 LCD 接口配置；

5.3.2 是 A10EVB 标配 5 寸屏的屏配置文件，HV 并行接口 800x480 分辨率；

5.3.3 是 4.3 寸屏 TD043 的屏配置文件；HV 并行接口 800x480 分辨率；该实例中有 IO 模拟串行接口，及插值 Gamma 表的实现；

5.3.4 是 10 寸屏 HSD100IFW1 的屏配置文件；LVDS Single Link 接口 1024x600 分辨率；

5.3.5 是 2.8 寸屏 KGM281I0 的屏配置文件；CPU 屏并行 18bit 接口。

5.3.1. sys_config1.fex

LCD 部分实例如下：

```
[lcd0_para]
lcd_used                = 1

lcd_power               = port:PH08<1><0><default><0>
;lcd_bl_en              = port:PH07<1><0><default><1>
lcd_pwm                 = port:PB02<2><default><default><default>

lcd_gpio_0              = port:PH07<1><0><default><1>
lcd_gpio_1              = port:PB10<1><0><default><1>
lcd_gpio_2              = port:PB11<1><0><default><1>
;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_para]
lcd_used                = 0
```

5.3.2. hv_800x480.c

```
/*****************************************************************************
 *
 *   hv_800x480.c
 *
 *****************************************************************************/

#include "lcd_panel_cfg.h"

//delete this line if you want to use the lcd para define in sys_config1.fex
#define LCD_PARA_USE_CONFIG

#ifdef LCD_PARA_USE_CONFIG

/*****************************************************************************
 *
 *   tcon parameters
 *
 *****************************************************************************/
static void LCD_cfg_panel_info(__panel_para_t * info)
{
    memset(info,0,sizeof(__panel_para_t));

     //interface
    info->lcd_if          = 0;      //0:hv;      1:cpu/8080;    2:reserved;    3:lvds
    info->lcd_hv_if       = 0;          //0:hv para;    1:hv serial;    2:ccir656

    //timing
    info->lcd_x           = 800;        //Hor Pixels
    info->lcd_y           = 480;        //Ver Pixels
    info->lcd_dclk_freq   = 33;         //Pixel Data Cycle,in MHz
    info->lcd_ht          = 1056;       //Hor Total Time
    info->lcd_hbp         = 216;        //Hor Back Porch
    info->lcd_vt          = 525*2;      //Ver Total Time*2
    info->lcd_vbp         = 35;         //Ver Back Porch
    info->lcd_hv_hspw     = 10;         //Hor Sync Time
    info->lcd_hv_vspw     = 10;         //Ver Sync Time
    info->lcd_io_cfg0     = 0x10000000; //Clock Phase

     //color
    info->lcd_frm         = 0;          //0: direct;    1: rgb666 dither;    2:rgb656 dither
```

```
	info->lcd_gamma_correction_en = 0;

	info->lcd_pwm_not_used   = 0;
	info->lcd_pwm_ch         = 0;
	info->lcd_pwm_freq       = 12500;        //Hz
	info->lcd_pwm_pol        = 0;
}

#endif


/*****************************************************************************
 *
 *   lcd flow function
 *    hv panel:first lcd_panel_init,than TCON_open
 *
 *****************************************************************************/
static __s32 LCD_open_flow(__u32 sel)
{
	LCD_OPEN_FUNC(sel, LCD_power_on,  50);    //open lcd power, than delay 50ms
	LCD_OPEN_FUNC(sel, TCON_open,             500);   //open lcd controller, than delay 500ms
	LCD_OPEN_FUNC(sel, LCD_bl_open,    0);      //open lcd backlight, than delay 0ms

	return 0;
}

static __s32 LCD_close_flow(__u32 sel)
{
	LCD_CLOSE_FUNC(sel, LCD_bl_close, 0);         //close lcd backlight, and delay 0ms
	LCD_CLOSE_FUNC(sel, TCON_close, 0);           //close lcd controller, and delay 0ms
	LCD_CLOSE_FUNC(sel, LCD_power_off, 1000);   //close lcd power, and delay 1000ms

	return 0;
}

/*****************************************************************************
 *
 *   lcd step function
 *
 *****************************************************************************/
static void LCD_power_on(__u32 sel)
{
	LCD_POWER_EN(sel, 1);
}
```

```
static void LCD_power_off(__u32 sel)
{
    LCD_POWER_EN(sel, 0);
}

static void LCD_bl_open(__u32 sel)
{
    LCD_PWM_EN(sel, 1);
    LCD_BL_EN(sel, 1);
}

static void LCD_bl_close(__u32 sel)
{
    LCD_BL_EN(sel, 0);
    LCD_PWM_EN(sel, 0);
}

/*****************************************************************************
 *
 *  user define function
 *
 *****************************************************************************/
static __s32 LCD_user_defined_func(__u32 sel, __u32 para1, __u32 para2, __u32 para3)
{
    return 0;
}

/*****************************************************************************
 *
 *  do not modify
 *
 *****************************************************************************/
void LCD_get_panel_funs_0(__lcd_panel_fun_t * fun)
{
#ifdef LCD_PARA_USE_CONFIG
    fun->cfg_panel_info = LCD_cfg_panel_info;
#endif
    fun->cfg_open_flow = LCD_open_flow;
    fun->cfg_close_flow = LCD_close_flow;
    fun->lcd_user_defined_func = LCD_user_defined_func;
}
```

5.3.3. hv_800x480_td043.c

```
/*****************************************************************************
 *
 *   hv_800x480_td043.c
 *
 *****************************************************************************/

#include "lcd_panel_cfg.h"

//delete this line if you want to use the lcd para define in sys_config1.fex
#define LCD_PARA_USE_CONFIG

#ifdef LCD_PARA_USE_CONFIG

/*****************************************************************************
 *
 *   tcon parameters
 *
 *****************************************************************************/
static void lcd_gamma_gen(__panel_para_t * info);
static void LCD_cfg_panel_info(__panel_para_t * info)
{
    memset(info,0,sizeof(__panel_para_t));

     //interface
    info->lcd_if              = 0;          //0:hv;          1:cpu/8080;    2:reserved;    3:lvds
    info->lcd_hv_if           = 0;              //0:hv para;     1:hv serial;     2:ccir656

    //timing
    info->lcd_x               = 800;        //Hor Pixels
    info->lcd_y               = 480;        //Ver Pixels
    info->lcd_dclk_freq       = 33;         //Pixel Data Cycle,in MHz
    info->lcd_ht              = 1056;       //Hor Total Time
    info->lcd_hbp             = 216;        //Hor Back Porch
    info->lcd_vt              = 525*2;      //Ver Total Time*2
    info->lcd_vbp             = 35;         //Ver Back Porch
    info->lcd_hv_hspw         = 10;         //Hor Sync Time
    info->lcd_hv_vspw         = 10;         //Ver Sync Time
    info->lcd_io_cfg0         = 0x10000000; //Clock Phase

     //color
    info->lcd_frm             = 0;          //0: direct;     1: rgb666 dither;     2:rgb656 dither
    info->lcd_gamma_correction_en = 0;      //Gamma Table enable
```

```
        lcd_gamma_gen(info);                    //Gamma Table Generation Func

        info->lcd_pwm_not_used   = 0;
        info->lcd_pwm_ch         = 0;
        info->lcd_pwm_freq       = 12500;       //Hz
        info->lcd_pwm_pol        = 0;
}

static void lcd_gamma_gen(__panel_para_t * info)
{
        const __u8 g_gamma_tbl[][2] =
        {
        //{input value, corrected value}
                {0, 0},
                {15, 15},
                {30, 30},
                {45, 45},
                {60, 60},
                {75, 75},
                {90, 90},
                {105, 105},
                {120, 120},
                {135, 135},
                {150, 150},
                {165, 165},
                {180, 180},
                {195, 195},
                {210, 210},
                {225, 225},
                {240, 240},
                {255, 255},
        };

        //insert value
        {
                __u32 items = sizeof(g_gamma_tbl)/2;
                __u32 i,j;
                for(i=0; i<items-1; i++)
                {
                        __u32 num = g_gamma_tbl[i+1][0] - g_gamma_tbl[i][0];
                        for(j=0; j<num; j++)
                        {
                                __u32 value = 0;
```

```
                        value = g_gamma_tbl[i][1]
                                + ((g_gamma_tbl[i+1][1] - g_gamma_tbl[i][1]) * j)/num;
                        info->lcd_gamma_tbl[g_gamma_tbl[i][0] + j]
                                = (value<<16) + (value<<8) + value;
                    }
                }
                info->lcd_gamma_tbl[255] = (g_gamma_tbl[items-1][1]<<16)
                                            + (g_gamma_tbl[items-1][1]<<8)
                                            + (g_gamma_tbl[items-1][1]);
        }

}
#endif


/*****************************************************************************
 *
 *   lcd flow function
 *    hv panel:first lcd_panel_init,than TCON_open
 *
 *****************************************************************************/
static __s32 LCD_open_flow(__u32 sel)
{
        LCD_OPEN_FUNC(sel, LCD_power_on,  50);     //open lcd power, than delay 50ms
        LCD_OPEN_FUNC(sel, LCD_panel_init,  50);     //lcd panel initial, than delay 50ms
        LCD_OPEN_FUNC(sel, TCON_open,             500);    //open lcd controller, than delay 500ms
        LCD_OPEN_FUNC(sel, LCD_bl_open,     0);       //open lcd backlight, than delay 0ms

        return 0;
}

static __s32 LCD_close_flow(__u32 sel)
{
        LCD_CLOSE_FUNC(sel, LCD_bl_close, 0);          //close lcd backlight, and delay 0ms
        LCD_CLOSE_FUNC(sel, TCON_close, 0);            //close lcd controller, and delay 0ms
        LCD_CLOSE_FUNC(sel, LCD_panel_exit, 0);       //lcd panel exit, and delay 0ms
        LCD_CLOSE_FUNC(sel, LCD_power_off, 1000);    //close lcd power, and delay 1000ms

        return 0;
}

/*****************************************************************************
 *
 *   lcd step function
```

```
 *
 ******************************************************************************/
static void LCD_power_on(__u32 sel)
{
    LCD_POWER_EN(sel, 1);
}

static void LCD_power_off(__u32 sel)
{
    LCD_POWER_EN(sel, 0);
}

static void LCD_bl_open(__u32 sel)
{
    LCD_PWM_EN(sel, 1);
    LCD_BL_EN(sel, 1);
}

static void LCD_bl_close(__u32 sel)
{
    LCD_BL_EN(sel, 0);
    LCD_PWM_EN(sel, 0);
}

/*****************************************************************************
 *
 *   lcd panel initial
 *    serial io initial
 *
 ******************************************************************************/
#define td043_spi_scen(sel,data)        LCD_GPIO_write(sel,2,data)
#define td043_spi_scl(sel,data)         LCD_GPIO_write(sel,1,data)
#define td043_spi_sda(sel,data)         LCD_GPIO_write(sel,0,data)

static void td043_spi_wr(__u32 sel,__u32 addr,__u32 value)
{
    __u32 i;
    __u32 data = (addr<<10 | value);
    td043_spi_scen(sel,1);
    td043_spi_scl(sel,0);
    td043_spi_scen(sel,0);
    for(i=0;i<16;i++)
    {
        if(data & 0x8000)
```

```
                td043_spi_sda(sel,1);
            else
                td043_spi_sda(sel,0);
            data <<= 1;
            LCD_delay_us(10);
            td043_spi_scl(sel,1);
            LCD_delay_us(10);
            td043_spi_scl(sel,0);
        }
        td043_spi_scen(sel,1);
}

static void td043_init(__u32 sel)
{
        td043_spi_wr(sel,0x02,0x07);
        td043_spi_wr(sel,0x03,0x5f);
        td043_spi_wr(sel,0x04,0x17);
        td043_spi_wr(sel,0x05,0x20);
        td043_spi_wr(sel,0x06,0x08);
        td043_spi_wr(sel,0x07,0x20);
        td043_spi_wr(sel,0x08,0x20);
        td043_spi_wr(sel,0x09,0x20);
        td043_spi_wr(sel,0x0a,0x20);
        td043_spi_wr(sel,0x0b,0x20);
        td043_spi_wr(sel,0x0c,0x20);
        td043_spi_wr(sel,0x0d,0x20);
        td043_spi_wr(sel,0x0e,0x10);
        td043_spi_wr(sel,0x0f,0x10);
        td043_spi_wr(sel,0x10,0x10);
        td043_spi_wr(sel,0x11,0x15);
        td043_spi_wr(sel,0x12,0xaa);
        td043_spi_wr(sel,0x13,0xff);
        td043_spi_wr(sel,0x14,0x86);
        td043_spi_wr(sel,0x15,0x8e);
        td043_spi_wr(sel,0x16,0xd6);
        td043_spi_wr(sel,0x17,0xfe);
        td043_spi_wr(sel,0x18,0x28);
        td043_spi_wr(sel,0x19,0x52);
        td043_spi_wr(sel,0x1a,0x7c);
        td043_spi_wr(sel,0x1b,0xe9);
        td043_spi_wr(sel,0x1c,0x42);
        td043_spi_wr(sel,0x1d,0x88);
        td043_spi_wr(sel,0x1e,0xb8);
        td043_spi_wr(sel,0x1f,0xff);
```

```
    td043_spi_wr(sel,0x20,0xf0);
}

static void LCD_panel_init(__u32 sel)
{
     td043_init(sel);
}

static void LCD_panel_exit(__u32 sel)
{

}

/*****************************************************************************
 *
 *   user define function
 *
 *****************************************************************************/
static __s32 LCD_user_defined_func(__u32 sel, __u32 para1, __u32 para2, __u32 para3)
{
    return 0;
}

/*****************************************************************************
 *
 *   do not modify
 *
 *****************************************************************************/
void LCD_get_panel_funs_0(__lcd_panel_fun_t * fun)
{
#ifdef LCD_PARA_USE_CONFIG
    fun->cfg_panel_info = LCD_cfg_panel_info;
#endif
    fun->cfg_open_flow = LCD_open_flow;
    fun->cfg_close_flow = LCD_close_flow;
    fun->lcd_user_defined_func = LCD_user_defined_func;
}
```

5.3.4. lvds_1024x600_hds100ifw1.c

```
/*****************************************************************************
 *
 *   lvds_1024x600_hds100ifw1.c
 *
 *****************************************************************************/

#include "lcd_panel_cfg.h"

//delete this line if you want to use the lcd para define in sys_config1.fex
#define LCD_PARA_USE_CONFIG

#ifdef LCD_PARA_USE_CONFIG

/*****************************************************************************
 *
 *   tcon parameters
 *
 *****************************************************************************/
static __u8 g_gamma_tbl[256];
static void LCD_cfg_panel_info(__panel_para_t * info)
{
    memset(info,0,sizeof(__panel_para_t));

     //interface
    info->lcd_if              = 3;      //0:hv;       1:cpu/8080;   2:reserved;   3:lvds
    info->lcd_lvds_ch         = 0;      //0:single link   1:dual link
    info->lcd_lvds_bitwidth = 1;        //0:24bit;    1:18bit;

    //timing
    info->lcd_x               = 1024;       //Hor Pixels
    info->lcd_y               = 600;        //Ver Pixels
    info->lcd_dclk_freq       = 52;         //Pixel Data Cycle,in MHz
    info->lcd_ht              = 1344;       //Hor Total Time
    info->lcd_hbp             = 20;         //Hor Back Porch
    info->lcd_vt              = 635*2;      //Ver Total Time*2
    info->lcd_vbp             = 20;         //Ver Back Porch

    info->lcd_hv_hspw         = 10;         //Hor Sync Time
    info->lcd_hv_vspw         = 10;         //Ver Sync Time
    info->lcd_io_cfg0         = 0x00000000;     //Clock Phase
```

```
    //color
    info->lcd_frm                = 1;          //0: direct;      1: rgb666 dither;      2:rgb656 dither
    info->lcd_gamma_correction_en = 0;

    info->lcd_pwm_not_used   = 0;
    info->lcd_pwm_ch         = 0;
    info->lcd_pwm_freq       = 12500;      //Hz
    info->lcd_pwm_pol        = 0;

}
#endif

/*****************************************************************************
 *
 *   lcd flow function
 *
 *****************************************************************************/
static __s32 LCD_open_flow(__u32 sel)
{
    LCD_OPEN_FUNC(sel, LCD_power_on,  50);    //open lcd power, than delay 50ms
//  LCD_OPEN_FUNC(sel, LCD_panel_init,  50);    //not in need
    LCD_OPEN_FUNC(sel, TCON_open,          500);   //open lcd controller, than delay 500ms
    LCD_OPEN_FUNC(sel, LCD_bl_open,    0);      //open lcd backlight, than delay 0ms

    return 0;
}

static __s32 LCD_close_flow(__u32 sel)
{
    LCD_CLOSE_FUNC(sel, LCD_bl_close, 0);         //close lcd backlight, and delay 0ms
    LCD_CLOSE_FUNC(sel, TCON_close, 0);           //close lcd controller, and delay 0ms
//  LCD_CLOSE_FUNC(sel, LCD_panel_exit, 0);      //not in need
    LCD_CLOSE_FUNC(sel, LCD_power_off, 1000);   //close lcd power, and delay 1000ms

    return 0;
}

/*****************************************************************************
 *
 *   lcd step function
 *
 *****************************************************************************/
static void LCD_power_on(__u32 sel)
```

```
{
    LCD_POWER_EN(sel, 1);
}

static void LCD_power_off(__u32 sel)
{
    LCD_POWER_EN(sel, 0);
}

static void LCD_bl_open(__u32 sel)
{
    LCD_PWM_EN(sel, 1);
    LCD_BL_EN(sel, 1);
}

static void LCD_bl_close(__u32 sel)
{
    LCD_BL_EN(sel, 0);
    LCD_PWM_EN(sel, 0);
}

/*****************************************************************************
 *
 *  user define function
 *
 *****************************************************************************/
static __s32 LCD_user_defined_func(__u32 sel, __u32 para1, __u32 para2, __u32 para3)
{
    return 0;
}
/*****************************************************************************
 *
 *  do not modify
 *
 *****************************************************************************/
void LCD_get_panel_funs_0(__lcd_panel_fun_t * fun)
{
#ifdef LCD_PARA_USE_CONFIG
    fun->cfg_panel_info = LCD_cfg_panel_info;
#endif
    fun->cfg_open_flow = LCD_open_flow;
    fun->cfg_close_flow = LCD_close_flow;
    fun->lcd_user_defined_func = LCD_user_defined_func;
```

5.3.5. cpu_320x240_kgm281i0.c

```
/*****************************************************************************
 *
 *   cpu_320x240_kgm281i0.c
 *
 *****************************************************************************/

#include "lcd_panel_cfg.h"

//delete this line if you want to use the lcd para define in sys_config1.fex
#define LCD_PARA_USE_CONFIG

#ifdef LCD_PARA_USE_CONFIG
/*****************************************************
 *
 *   tcon parameters
 *
 *****************************************************/
static __u8 g_gamma_tbl[256];
static void LCD_cfg_panel_info(__panel_para_t * info)
{
    memset(info,0,sizeof(__panel_para_t));

     //interface
    info->lcd_if              = 1;          //0:hv;          1:cpu/8080;    2:reserved;    3:lvds
    info->lcd_cpu_if          = 0;          //0:18bit        4:16bit

    //timing
    info->lcd_x               = 320;        //Hor Pixels
    info->lcd_y               = 240;        //Ver Pixels
    info->lcd_dclk_freq       = 6;          //Pixel Data Cycle
    info->lcd_ht              = 320+30;     //Hor Total Time
    info->lcd_hbp             = 20;               //Hor Back Porch
    info->lcd_vt              = (240+30)*2; //Ver Total Time*2
    info->lcd_vbp             = 20;         //Ver Back Porch
    info->lcd_hv_hspw         = 10;         //Hor Sync Time
    info->lcd_hv_vspw         = 10;         //Ver Sync Time
    info->lcd_io_cfg0         = 0x10000000; //Clock Phase

     //color
    info->lcd_frm             = 1;          //0: direct;     1: rgb666 dither;    2:rgb656 dither
    info->lcd_gamma_correction_en = 0;
```

```
        //backlight
        info->lcd_pwm_not_used   = 0;
        info->lcd_pwm_ch         = 0;
        info->lcd_pwm_freq       = 12500;        //Hz
        info->lcd_pwm_pol        = 0;
}
#endif

/*****************************************************
 *
 *   lcd flow function
 *     CPU Panel:first TCON_open,than lcd_panel_init
 *
 *****************************************************/
static __s32 LCD_open_flow(__u32 sel)
{
        LCD_OPEN_FUNC(sel, LCD_power_on,  50);    //open lcd power, than delay 50ms
        LCD_OPEN_FUNC(sel, TCON_open,     500);   //open lcd controller, than delay 500ms
        LCD_OPEN_FUNC(sel, LCD_panel_init,  50);  //lcd panel initial, than delay 50ms
        LCD_OPEN_FUNC(sel, LCD_bl_open,   0);     //open lcd backlight, than delay 0ms

        return 0;
}

static __s32 LCD_close_flow(__u32 sel)
{
        LCD_CLOSE_FUNC(sel, LCD_bl_close,   0);          //close lcd backlight, than delay 0ms
        LCD_CLOSE_FUNC(sel, LCD_panel_exit, 0);          //lcd panel exit, than delay 0ms
        LCD_CLOSE_FUNC(sel, TCON_close,     0);          //close lcd controller, than delay 0ms
        LCD_CLOSE_FUNC(sel, LCD_power_off, 1000);    //close lcd power, than delay 1000ms

        return 0;
}


/*****************************************************
 *
 *   lcd step function
 *
 *****************************************************/
static void LCD_power_on(__u32 sel)
{
    LCD_POWER_EN(sel, 1);
}
```

```
static void LCD_power_off(__u32 sel)
{
    LCD_POWER_EN(sel, 0);
}

static void LCD_bl_open(__u32 sel)
{
    LCD_PWM_EN(sel, 1);
    LCD_BL_EN(sel, 1);
}

static void LCD_bl_close(__u32 sel)
{
    LCD_BL_EN(sel, 0);
    LCD_PWM_EN(sel, 0);
}


/*********************************************************
 *
 *   lcd panel initial
 *    cpu 8080 bus initial
 *
 *********************************************************/
#define kgm281i0_rs(sel,data)  LCD_GPIO_write(sel,0,data)

static void kgm281i0_write_gram_origin(__u32 sel)
{
    LCD_CPU_WR(sel,0x0020, 0);        // GRAM horizontal Address
    LCD_CPU_WR(sel,0x0021, 319);     // GRAM Vertical Address
    LCD_CPU_WR_INDEX(sel,0x22);    // Write Memery Start
}

static void kgm281i0_init(__u32 sel)
{
    kgm281i0_rs(sel,1);
    LCD_delay_ms(50);
    kgm281i0_rs(sel,0);
    LCD_delay_ms(50);
    kgm281i0_rs(sel,1);

    LCD_CPU_WR(sel,0x0000, 0x0001);
    LCD_CPU_WR(sel,0x0001, 0x0100);
```

```
LCD_CPU_WR(sel,0x0002, 0x0400);
LCD_CPU_WR(sel,0x0003, 0x1018);
LCD_CPU_WR(sel,0x0004, 0x0000);
LCD_CPU_WR(sel,0x0008, 0x0202);
LCD_CPU_WR(sel,0x0009, 0x0000);
LCD_CPU_WR(sel,0x000A, 0x0000);
LCD_CPU_WR(sel,0x000C, 0x0000);
LCD_CPU_WR(sel,0x000D, 0x0000);
LCD_CPU_WR(sel,0x000F, 0x0000);
LCD_CPU_WR(sel,0x0010, 0x0000);
LCD_CPU_WR(sel,0x0011, 0x0007);
LCD_CPU_WR(sel,0x0012, 0x0000);
LCD_CPU_WR(sel,0x0013, 0x0000);
LCD_delay_ms(50);
LCD_CPU_WR(sel,0x0010, 0x17B0);
LCD_CPU_WR(sel,0x0011, 0x0001);
LCD_delay_ms(50);
LCD_CPU_WR(sel,0x0012, 0x013C);
LCD_delay_ms(50);
LCD_CPU_WR(sel,0x0013, 0x1300);
LCD_CPU_WR(sel,0x0029, 0x0012);
LCD_delay_ms(50);
LCD_CPU_WR(sel,0x0020, 0x0000);
LCD_CPU_WR(sel,0x0021, 0x0000);
LCD_CPU_WR(sel,0x002B, 0x0020);
LCD_CPU_WR(sel,0x0030, 0x0000);
LCD_CPU_WR(sel,0x0031, 0x0306);
LCD_CPU_WR(sel,0x0032, 0x0200);
LCD_CPU_WR(sel,0x0035, 0x0107);
LCD_CPU_WR(sel,0x0036, 0x0404);
LCD_CPU_WR(sel,0x0037, 0x0606);
LCD_CPU_WR(sel,0x0038, 0x0105);
LCD_CPU_WR(sel,0x0039, 0x0707);
LCD_CPU_WR(sel,0x003C, 0x0600);
LCD_CPU_WR(sel,0x003D, 0x0807);
LCD_CPU_WR(sel,0x0050, 0x0000);
LCD_CPU_WR(sel,0x0051, 0x00EF);
LCD_CPU_WR(sel,0x0052, 0x0000);
LCD_CPU_WR(sel,0x0053, 0x013F);
LCD_CPU_WR(sel,0x0060, 0x2700);
LCD_CPU_WR(sel,0x0061, 0x0001);
LCD_CPU_WR(sel,0x006A, 0x0000);
LCD_CPU_WR(sel,0x0080, 0x0000);
LCD_CPU_WR(sel,0x0081, 0x0000);
```

```
	LCD_CPU_WR(sel,0x0082, 0x0000);
	LCD_CPU_WR(sel,0x0083, 0x0000);
	LCD_CPU_WR(sel,0x0084, 0x0000);
	LCD_CPU_WR(sel,0x0085, 0x0000);
	LCD_CPU_WR(sel,0x0090, 0x0013);
	LCD_CPU_WR(sel,0x0092, 0x0000);
	LCD_CPU_WR(sel,0x0093, 0x0003);
	LCD_CPU_WR(sel,0x0095, 0x0110);
	LCD_CPU_WR(sel,0x0097, 0x0000);
	LCD_CPU_WR(sel,0x0098, 0x0000);
	LCD_CPU_WR(sel,0x0007, 0x0001);
	LCD_delay_ms(50);
	LCD_CPU_WR(sel,0x0007, 0x0021);
	LCD_CPU_WR(sel,0x0007, 0x0023);
	LCD_delay_ms(50);
	LCD_CPU_WR(sel,0x0007, 0x0173);
}

static void Lcd_cpuisr_proc(void)                    //irq func
{
	kgm281i0_write_gram_origin(0);
}

static void LCD_panel_init(__u32 sel)
{
	kgm281i0_init(sel);                              //initial lcd panel
	kgm281i0_write_gram_origin(sel);                 //set gram origin
	LCD_CPU_register_irq(sel,Lcd_cpuisr_proc);       //resgister cpu irq func
	LCD_CPU_AUTO_FLUSH(sel,1);                       //start sent gram data
}

static void LCD_panel_exit(__u32 sel)
{

}
/*************************************************************
 *
 *   user define function
 *
 *************************************************************/
static __s32 LCD_user_defined_func(__u32 sel, __u32 para1, __u32 para2, __u32 para3)
{
	return 0;
}
```

```
/*****************************************************
 *
 *   do not modify
 *
 *****************************************************/
void LCD_get_panel_funs_0(__lcd_panel_fun_t * fun)
{
#ifdef LCD_PARA_USE_CONFIG
        fun->cfg_panel_info = LCD_cfg_panel_info;
#endif
        fun->cfg_open_flow = LCD_open_flow;
        fun->cfg_close_flow = LCD_close_flow;
        fun->lcd_user_defined_func = LCD_user_defined_func;
}
```

5.4. LCD CHECK LIST

LCD CHECK LIST

方案：　　屏规格：　　日期：

<table>
<tr><td rowspan="3">电压检查</td><td></td><td>DVDD</td><td>AVDD</td><td>VHG</td><td>VGL</td><td>VCOM</td></tr>
<tr><td>开屏状态 □</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>关屏状态 □</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="2">效果检查</td><td>位置渐变图 □</td><td colspan="2">色深灰阶图□</td><td colspan="3">文字图 □</td></tr>
<tr><td>色块图 □</td><td colspan="2">亮度调节 □</td><td colspan="3">快慢开关屏□</td></tr>
<tr><td>签名</td><td colspan="6"></td></tr>
</table>