前言

文档简介

本文档为全志T系列的linuxSDK多屏配置指南,阐述了从uboot到的每个阶段中屏幕输出的选择以及应用。

目标读者

需要在T系列平台上进行多屏配置的开发人员。

适用范围

全志T系列产品。

主显默认输出配置说明

T3/A40i/T7 配置方法

对于T3/A40i/T7 linux sdk默认情况下，我们的主屏幕配的是1024x600的lvds输出。而如果想修改成tvout(cvbs)或者hdmi，又或者vga,
则需要做如下修改：

tools\pack\chips\sun8iw17p1\configs\t7-p1\sys_config.fex

```
;------------------------------------------------------------------------------
;boot disp init configuration
;output_disp    0:screen0;  1:screen1
;output_type    0:none; 1:lcd; 2:tvout 3:hdmi;
;output_mode    typically:(type:mode) ===> (1,4)~LCD;(2,14)~TVOUT;(3,9)~HDMI;(4,81)~VGA
;------------------------------------------------------------------------------
[boot_disp]
output_disp = 0
output_type = 1
output_mode = 4
[disp]
disp_init_enable       = 1
disp_mode              = 0
screen0_output_type    = 1
screen0_output_mode    = 4
```

（PS:若没有[boot_disp]字段的话，直接在文件最后追加即可）

解析：output_disp=0**表示主显，也可以理解为**screen0（T3/A40I/T7**可以同时驱动两个屏）。**
output_type=1**表示配的是**lcd**；**output_type=1=2**表示配的是**tvout**；**output_type=1=3**表示配成**hdmi**输出**
。

对应的代码段为：

brandy/u-boot-2014.07/board/sunxi/common/de_v2.c

```
if (fdt_getprop_u32(working_fdt, node, "output_type", (uint32_t*)&value) < 0) {
    printf("fetch script data boot_disp.output_type fail\n");
    err_count ++;
} else
    printf("boot_disp.output_type=%d\n", value);

if(value == 0)
{
    output_type = DISP_OUTPUT_TYPE_NONE;
}
else if(value == 1)
{
    output_type = DISP_OUTPUT_TYPE_LCD;
}
else if(value == 2)
{
    output_type = DISP_OUTPUT_TYPE_TV;
}
else if(value == 3)
{
    output_type = DISP_OUTPUT_TYPE_HDMI;
}
else if(value == 4)
{
    output_type = DISP_OUTPUT_TYPE_VGA;
}
```

因为针对每一种输出屏都有其对应的一些分辨率，这个就用output_mode**来表征，下面看一下这个结构体就清楚了**

结构体所在头文件位于"linux-xxx\include\video\sunxi_display2.h"

```
typedef enum
```

```
{
DISP_TV_MOD_480I          = 0,
DISP_TV_MOD_576I          = 1,
DISP_TV_MOD_480P          = 2,
DISP_TV_MOD_576P          = 3,
DISP_TV_MOD_720P_50HZ     = 4,
DISP_TV_MOD_720P_60HZ     = 5,
DISP_TV_MOD_1080I_50HZ    = 6,
DISP_TV_MOD_1080I_60HZ    = 7,
DISP_TV_MOD_1080P_24HZ    = 8,
DISP_TV_MOD_1080P_50HZ    = 9,   #对应(3,9)~HDMI
DISP_TV_MOD_1080P_60HZ    = 0xa,
DISP_TV_MOD_1080P_24HZ_3D_FP  = 0x17,
DISP_TV_MOD_720P_50HZ_3D_FP   = 0x18,
DISP_TV_MOD_720P_60HZ_3D_FP   = 0x19,
DISP_TV_MOD_1080P_25HZ    = 0x1a,
DISP_TV_MOD_1080P_30HZ    = 0x1b,
DISP_TV_MOD_PAL           = 0xb,
DISP_TV_MOD_PAL_SVIDEO    = 0xc,
DISP_TV_MOD_NTSC          = 0xe,  #对应(2,14)~TVOUT
DISP_TV_MOD_NTSC_SVIDEO   = 0xf,
DISP_TV_MOD_PAL_M         = 0x11,
DISP_TV_MOD_PAL_M_SVIDEO  = 0x12,
DISP_TV_MOD_PAL_NC        = 0x14,
DISP_TV_MOD_PAL_NC_SVIDEO = 0x15,
DISP_TV_MOD_3840_2160P_30HZ   = 0x1c,
DISP_TV_MOD_3840_2160P_25HZ   = 0x1d,
DISP_TV_MOD_3840_2160P_24HZ   = 0x1e,
/* vga */
  DISP_VGA_MOD_640_480P_60    = 0x50,
  DISP_VGA_MOD_800_600P_60    = 0x51,  #对应(4,81)~VGA
  DISP_VGA_MOD_1024_768P_60   = 0x52,
  DISP_VGA_MOD_1280_768P_60   = 0x53,
  DISP_VGA_MOD_1280_800P_60   = 0x54,
  DISP_VGA_MOD_1366_768P_60   = 0x55,
  DISP_VGA_MOD_1440_900P_60   = 0x56,
  DISP_VGA_MOD_1920_1080P_60  = 0x57,
  DISP_VGA_MOD_1280_720P_60   = 0x58,
  DISP_VGA_MOD_1920_1200P_60  = 0x5a,
  DISP_TV_MODE_NUM            = 0x5b,
}disp_tv_mode;
```

值得注意的是，对于输出是lcd的情况，output_mode没有任何意义，即如果output_type=1的话，系统

就认为是lcd了，output_mode是多少都无所谓。注释中写了常见的一些格式类型。

(type,mode) ===>
(1,4)　~LCD;　输出lcd
(2,14)　~TVOUT; 输出NTSC格式的tvout(cvbs)信号
(3,9)　~HDMI; 输出1080P50Hz的hdmi信号
(4,81)　~VGA　输出800x600分辨率的vga信号

（PS:T7没有HDMI和VGA）

所以，如果想默认改成cvbs 720x480分辨率输出的话(其他的hdmi/vga也类似)，可以改成这样：

```
[boot_disp]
output_disp = 0
output_type = 2
output_mode = 14
[disp]
disp_init_enable      = 1
disp_mode             = 0
screen0_output_type   = 2
screen0_output_mode   =14
```

T5 linux的lcd输出配置稍有不同，它被放在放在了
“ lichee\device\config\chips\t507\configs\demo2.0\board.dts ”中

```
disp: disp@01000000 {
disp_init_enable      = <1>;
disp_mode             = <0>;

screen0_output_type   = <1>;
screen0_output_mode   = <4>;
    ...
dev0_output_type      = <1>;
dev0_output_mode      = <4>;
```

需要注意的是，T5 除了改screen0_output_type/mode之外，dev0_output_type/mode也需要对应的做同步修改才能生效。

下面是把主显默认改成cvbs输出的配置：

```
disp: disp@01000000 {
disp_init_enable      = <1>;
disp_mode             = <0>;

screen0_output_type   = <2>;
```

```
screen0_output_mode     = <14>;
        ...
dev0_output_type        = <2>;
dev0_output_mode        = <14>;
        ...
fb0_format              = <0>;
fb0_width               = <720>;
fb0_height              = <480>;
```

量产卡的进度条显示输出也是跟boot_disp的配置走的。

显示节点参数

显示节点查看方法

```
cat /sys/class/disp/disp/attr/sys
```

示例如下：

```
# cat /sys/class/disp/disp/attr/sys
screen 0:
de_rate 432000000 Hz /* de 的时钟频率*/, ref_fps=50 /* 输出设备的参考刷新率*/
lcd output mode(0) fps:50.5 1280x 720
err:0 skip:54 irq:21494 vsync:0
BUF enable ch[0] lyr[0] z[0] prem[N] a[globl 255] fmt[ 1] fb[1920,1080;1920,1080;1920,1080] crop[ 0, 0,1920,1080] frame[ 32, 18,1216, 684]
addr[716da000, 0, 0] flags[0x 0] trd[0,0]
screen 1:
de_rate 432000000 Hz /* de 的时钟频率*/, ref_fps=50 /* 输出设备的参考刷新率*/
lcd output mode(0) fps:50.5 1280x 720
err:0 skip:54 irq:8372 vsync:0
BUF enable ch[0] lyr[0] z[0] prem[Y] a[globl 255] fmt[ 0] fb[ 720, 576; 720, 576; 720, 576] crop[ 0, 0, 1280, 720] frame[ 18, 15, 684, 546]
addr[739a8000, 0, 0] flags[0x 0] trd[0,0]
acquire: 225, 2.6 fps
release: 224, 2.6 fps
display: 201, 2.5 fps
```

图层各信息描述如下：

BUF: 图层类型，BUF/COLOR，一般为BUF，即图层是带BUFFER 的。COLOR 意思是显示一个纯色的画面，不带BUFFER。

enable: 显示处于enable 状态

ch[0]: 该图层处于blending 通道0

lyr[0]: 该图层处于当前blending 通道中的图层0

z[0]: 图层z 序，越小越在底部，可能会被z 序大的图层覆盖住
prem[Y]: 是否预乘格式，Y 是，N 否
a: alpha 参数， globl/pixel/; alpha 值
fmt: 图层格式，值64 以下为RGB 格式；以上为YUV 格式，常见的72 为YV12,76 为NV12
fb: 图层buffer 的size，width,height，三个分量
crop: 图像buffer 中的裁减区域， [x,y,w,h]
frame: 图层在屏幕上的显示区域，[x,y,w,h]
addr: 三个分量的地址
flags: 一般为0， 3D SS 时0x4, 3D TB 时为0x1, 3D FP 时为0x2;
trd: 是否3D 输出，3D 输出的类型（HDMI FP 输出时为1）各counter 描述如下：
err： de 缺数的次数，de 缺数可能会出现屏幕抖动，花屏的问题。de 缺数一般为带宽不足引起。
skip: 表示de 跳帧的次数，跳帧会出现卡顿问题。跳帧是指本次中断响应较慢，de 模块判断在本次中断已经接近或者超过了消隐区，将放弃本次更新图像的机会，选择继续显示原有的图像。
irq: 表示该通路上垂直消隐区中断执行的次数，一直增长表示该通道上的timing controller 正在运行当中。
vsync:表示显示模块往用户空间中发送的vsync 消息的数目，一直增长表示正在不断地发送中。

acquire/release/display 含义如下,只在android 方案中有效：
acquire： 是hw composer 传递给disp driver 的图像帧数以及帧率，帧率只要有在有图像更新时才有效，静止时的值是不准确的
release: 是disp driver 显示完成之后，返还给android 的图像帧数以及帧率，帧率只要有在有图像更新时才有效，静止时的值是不准确的
display： 是disp 显示到输出设备上的帧数以及帧率，帧率只要有在有图像更新时才有效，静止时的值是不准确的如果acquire 与release 不一致，说明disp 有部分图像帧仍在使用，未返还，差值在1~2 之间为正常值。二者不能相等，如果相等，说明图像帧全部返还，显示将会出现撕裂现象。如果display 与release 不一致，说明在disp 中存在丢帧情况，原因为在一个active 区内hwcomposer 传递多于一帧的图像帧下来

应用端副显同步输出

如果开机启动后没有自动运行sdktest或者CameraUI等应用的话，副显是处于关闭的状态，如果想要把主显的数据同步输出到副显tvout的话，需要用软件的方法来打开。
为什么运行了sdktest或者CameraUI等应用的话副显会被激活？那是因为这两个应用里调用了HwDisplay类的hwd_screen1_mode()方法：

函数原型int hwd_screen1_mode(int mode);
功能screen0和screen1同时显示FB0的内容。
参数mode,目前只支持下面几种：
DISP_TV_MOD_720P_60HZ = 5,
DISP_TV_MOD_1080P_50HZ = 9,
DISP_TV_MOD_PAL = 0xb,
DISP_TV_MOD_NTSC = 0xe,
DISP_VGA_MOD_640_480P_60 = 0x50,

DISP_VGA_MOD_800_600P_60 = 0x51,
DISP_VGA_MOD_1024_768P_60 = 0x52,
DISP_VGA_MOD_1280_768P_60 = 0x53,
其中,DISP_TV_MOD_720P_60HZ/DISP_TV_MOD_1080P_50HZ是hdmi输出
，DISP_TV_MOD_PAL/DISP_TV_MOD_NTSC是cvbs输出，剩下的是vga输出
返回值
调用说明在调用HwDisplay::getInstance成功后，即可调用该函数，使screen的数据同时输出到不同显示器上。
（PS:T7没有HDMI和VGA输出）
当然，如果用户不想用sdktest或者CameraUI，也可以把这个类合到自己的应用程序中,demo如下：

```
#include<stdlib.h>
#include "hwdisp2.h"
int main(int argc, char *argv[])
{
  android::HwDisplay *mcd = NULL;
  int mode = 0x0e;       //DISP_TV_MOD_NTSC=0x0e
    mcd = android::HwDisplay::getInstance();
    if(!mcd){
  printf( “ get mcd failed\n ” );
  return 0;
}
    mcd->hwd_screen1_mode(mode);//副显跟随主显输出
  return 0;
}
```

外部依赖：
此demo依赖于buildroot-201611\target\user_rootfs_misc目录下的
sdk_lib/libsdk_disp.so 以及HwDisplay类的头文件
sdk_lib/include/disp2/下的hwdisp2.h、typedef.h 和sunxi_display2.h
（以前版本的软件可能还依赖于sdk_lib/include/cutils和sdk_lib/include/utils）

注意事项：
1. HwDisplay类是直接控制硬件用的，所以不可以有多个实例，或者多个应用程序同时跑多个HwDisplay。建议单独写一个小应用程序，或者做成类似server的东西来管理。
2. 目前只做了主显FB0的映射，如果涉及到多个FB的情况，则需要根据具体场景来进行进一步的适配。