密级状态：绝密（　　）　秘密（　　）　内部（　）　公开（ √ ）

# 基于 RK FB 显示框架的常见问题

（图形显示平台中心）

| | | |
|---|---|---|
| 文件状态：<br>[　] 正在修改<br>[√] 正式发布 | 当前版本： | V1.1 |
| | 作　　者： | 黄家钗 |
| | 完成日期： | 2017-6-6 |
| | 审　　核： | 黄德胜、姚智情 |
| | 完成日期： | 2017-6-6 |

福州瑞芯微电子股份有限公司

Fuzhou　Rockchips　Semiconductor　Co . , Ltd

# 版 本 历 史

| 版本号 | 作者 | 修改日期 | 修改说明 | 备注 |
|---|---|---|---|---|
| V1.0 | 黄家钗 | 2017-6-6 | 初始版本 | |
| V1.1 | 黄家钗 | 2017-6-27 | 加入 RGB/TLL 输出硬件连接图 | |
| | | | | |
| | | | | |
| | | | | |

# 目 录

# 一、各平台显示接口最大支持分辨率

1. RV1108:

（1）RGB: 支持 RGB666/sRGB888 接口，最大支持 1280*720@60fps 输出；

（2）MIPI：支持 DSI v1.0,DCS v1.0, DPHY v1.0 协议标准，最大支持 1280*720@60fps 输出；

（3）HDMI：支持 HDMI 1.4a 协议标准，最大支持 1920*1080p@60fps 输出；

（4）CVBS: 支持 NTSC/PAL 标准输出；

2. RK312X:

（1）RGB: 支持 RGB666 接口，最大支持 1280*800@60fps 输出；

（2）LVDS: 支持 VESA 和 JEIDA LVDS 数据格式,最大支持 1280*800@60fps 输出；

（3）MIPI： 支持 DSI v1.0,DCS v1.0, DPHY v1.0 协议标准，最大支持 1280*800@60fps 输出；

（4）HDMI：支持 HDMI 1.4a 协议标准，最大支持 1920*1080p@60fps 输出(RK3128 支持 HDMI 输出接口，RK3126/RK3126B/RK3126C 不支持 HDMI 输出接口)；

（5）CVBS: 支持 NTSC/PAL 标准输出；

3. RK3288/RK3288W：

（1）RGB(RGB888/RGB666)：

VOP_BIG：1920*1080@60fps

VOP_LITTLE：1920*1080@60fps

（2）LVDS：

支持 VESA 和 JEIDA LVDS 数据格式，LVDS PHY 支持最大工作频率为 148.5M，所以每个通道支持的最大数据量为：148.5M * 7bit/per cycle = 1.03Gbps。

单通道 LVDS 屏最大支持的数据量为：1.03 Gbps* 4lane = 4.12Gbps，理论上最大可以支持：1920*1080@60fps，RK3288 目前实际用过最大的分辨率为：1280*800@60fps。

双通道 LVDS 屏最大支持的数据量为：1.03Gbps * 8lane = 8.24Gbps，理论上最大

可以支持：2560*1600@60fps，RK3288 目前实际用过最大的分辨率为：1920*1200@60fps

（3）MIPI：

支持 DSI v1.1，DCS v1.1，DPHY v1.1 协议标准，带宽为：1.5Gbps/lane。

单通道 MIPI 屏理论最大可以支持数据量为：1.5Gbps * 4 = 6 Gbps， 目前确认支持的最大分辨率为：1920*1200@60fps、1200*1080@90fps。

双通道 MIPI 屏理论最大可以支持数据量为：1.5Gbps * 8 = 12 Gbps，目前确认支持的最大分辨率为：2560*1600@60fps、1440*2560@75fps。

（4）eDP：

支持 DP1.2a 和 eDP1.3 协议标准，带宽为：2.7Gbps/lane 理论最大可以支持数据量为：2.7Gbps/lane＊4lane = 10.28Gbps，目前确认支持的最大分辨率：2560*1600@60fps。

（5）HDMI：

支持 HDMI 1.4a 和 2.0 协议标准。

VOP_BIG：最大支持 3840*2160@60fps 输出。

VOP_LITTLE：最大支持 1920*1080@60fps 输出。

4. RK3368/RK3368H:

（1）RGB: 支持 RGB888/RGB666 接口，最大支持 1920*1080@60fps 输出；

（2）LVDS: 支持 VESA 和 JEIDA LVDS 数据格式，目前使用过最大 1280*800@60fps 输出；

（3）MIPI：支持 DSI v1.0，DCS v1.0， DPHY v1.0 协议标准，最大支持 1920*1200@60fps 输出；

（4）eDP：支持 DP1.2a 和 eDP1.3 协议标准，带宽为：2.7Gbps/lane 理论最大可以支持数据量为：2.7Gbps/lane * 4lane = 10.28Gbps，目前确认支持的最大分辨率：2048*1536@60fps。

（5）HDMI：支持 HDMI 1.4a 和 2.0 协议标准，最大支持 4096*2160@60fps 输出；

5、RK3399:

（1）MIPI：

支持 DSI v1.1，DCS v1.1，DPHY v1.1 协议标准，带宽为：1.5Gbps/lane。

单通道 MIPI 屏理论最大可以支持数据量为：1.5Gbps * 4 = 6 Gbps， 目前确认支持的最大分辨率为：1920*1200@60fps、1200*1080@90fps。

双通道 MIPI 屏理论最大可以支持数据量为：1.5Gbps * 8 = 12 Gbps，目前确认支持的最大分辨率为：2560*1600@60fps、1440*2560@75fps。

（2）DP:

支持 DP 1.2 协议标准，带宽为：5.4Gbps/lane。

VOP_BIG：最大支持 4096*2160@60fps 输出。

VOP_LITTLE：最大支持 1920*1080@60fps 输出。

（3）eDP：

支持 DP1.2a 和 eDP1.3 协议标准，带宽为：5.4Gbps/lane 理论最大可以支持数据量为：5.4Gbps/lane＊4lane＝20.16Gbps，目前确认支持的最大分辨率：3840*2160@60fps。

（4）HDMI：

支持 HDMI 1.4a 和 2.0a 协议标准。

VOP_BIG：最大支持 3840*2160@60fps 输出。

VOP_LITTLE：最大支持 1920*1080@60fps 输出。

# 二、LCD 时序配置说明

1. 屏配置文件位置

3.10 版本内核： /kernel/arch/arm/boot/dts/lcd-xxx.dtsi

4.4 版本内核：/kernel/include/dt-bindings/display/screen-timing/lcd-xxx.dtsi

2. 属性说明

(1) screen-type: 屏的类型，比如 RGB 屏(SCREEN_RGB)，LVDS 屏(SCREEN_LVDS)，mipi 屏(SCREEN_MIPI)等；

(2) out-face: 输出接口，可以是 RGB888(OUT_P888)，RGB666(OUT_P666)，RGB888 高 6 位连接(OUT_D888_P666)等；

(3) clock-frequency: dclk 频率，比如 74.25M，clock-frequency = <74250000>;

(4) hactive: 水平有效像素;

(5) vactive: 垂直有效像素;

(6) hback-porch: 水平方向后消隐;

(7) hfront-porch: 水平方向前消隐;

(8) vback-porch: 垂直方向后消隐;

(9) vfront-porch: 垂直方向前消隐;

(10) hsync-len: 水平同步信号;

(11) vsync-len: 垂直同步信号;

(12) hsync-active: hsync 信号极性配置;

(13) vsync-active: vsync 信号极性配置;

(14) de-active: den 信号极性配置;

(15) pixelclk-active: dclk 信号极性配置;

(16) swap-rb: VOP 输出 rb 通道互换;

(17) swap-rg: VOP 输出 rg 通道互换;

(18) swap-gb: VOP 输出 gb 通道互换;

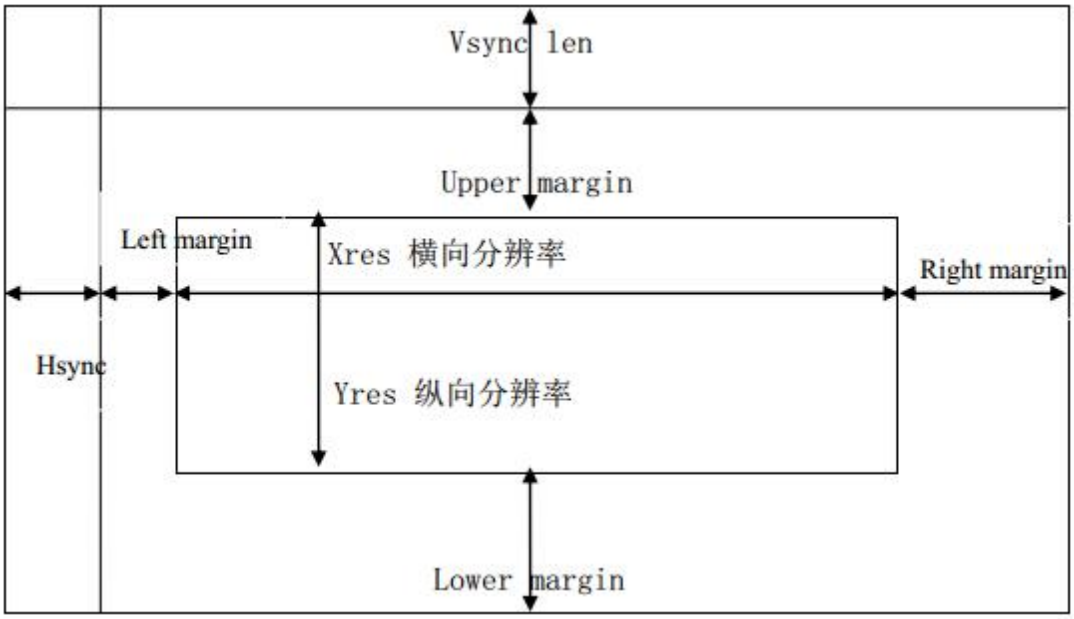
（19）screen-width: LCD 显示区域宽度，单位 mm;

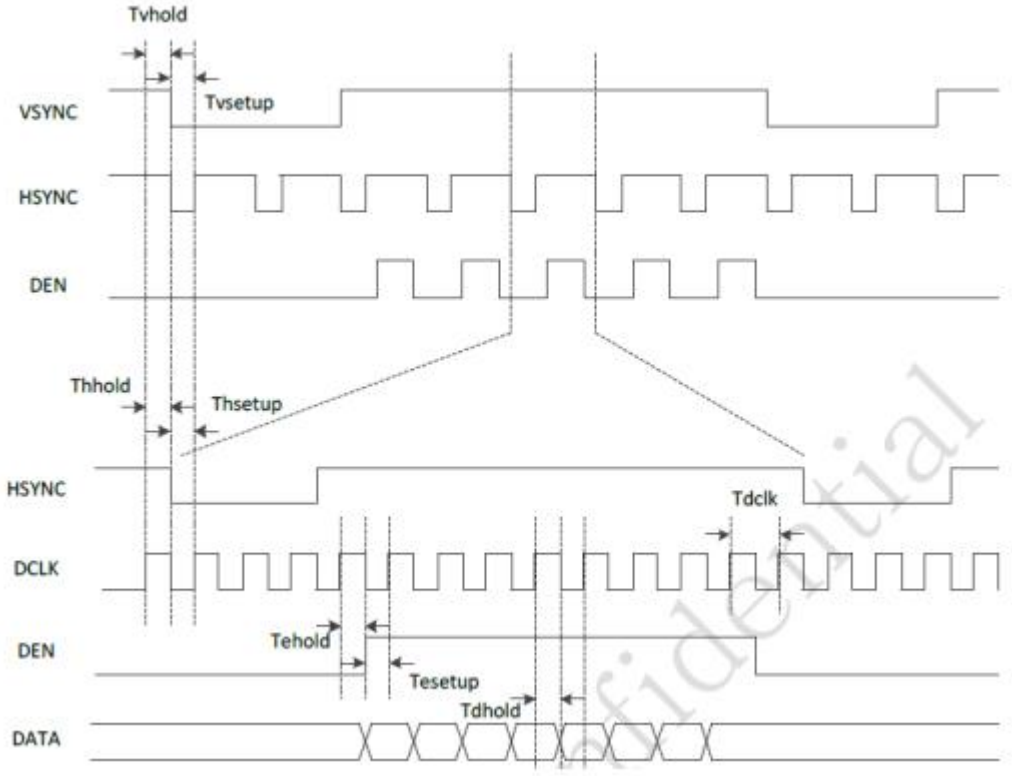
（20）screen-hight: LCD 显示区域高度，单位 mm;

（21）dsp-lut: 屏 gamma 调整映射曲线，如不需要此功能可以删去或者不要修改默认值;

（22）cabc-lut: CABC 功能屏 gamma 配置，目前就 RK3399/RK3288W 支持 CABC 功能，如不需要此功能可以删去或者不要修改默认值;

3. 时序图




4. VOP 刷新帧率确认

（1）可以通过节点 cat /sys/class/graphics/fbx/fps 知道当前 vop 的帧率；

（2）手动计算帧率 = dclk / ((hback-porch + hsync-len + hfront-porch) * (vback-porch + vsync-len + vfront-porch))

（3）debug 的时候可以通过命令修改帧率 echo value > /sys/class/graphics/fbx/fps,

val 值为需要设定的帧率，比如 60，这样修改可能会出现颜色不对的问题，特别是在一些 mipi 屏上。

（4）以上 fbx 是 vop 硬件对应的设备节点，具体哪个 vop 对应哪个节点每个平台不一样，总的来说，主显设备固定为 fb0，次显设备可以查看/sys/class/graphics/目录下有几个 fbx，除以 2 后就是次显设备的节点。

# 三、BCSH 使用说明

1. BCSH 是 Brightness、Contrast、Saturation 和 Hue 的简称，可以用于调节显示的亮度、对比度、饱和度和色度。

2. 操作说明：

（1）打开 bcsh： echo open > /sys/class/graphics/fb0/bcsh;

（2）配置亮度： echo "brightness 128" > /sys/class/graphics/fb0/bcsh，调节范围是[0, 255]，默认值是 128；

（3）配置对比度：echo "contrast 256" > /sys/class/graphics/fb0/bcsh，调节范围是[0, 510]，默认值是 256；

（4）配置饱和度： echo "sat_con 256" > /sys/class/graphics/fb0/bcsh，调节范围是[0, 1015]，默认值是 256；

（5）配置色度： echo "hue sin(a) cos(a)" > /sys/class/graphics/fb0/bcsh，a 的值为[-30°,30°]；

（6）关闭 bcsh： echo close > /sys/class/graphics/fb0/bcsh;

3. 调节正常的效果后应用可以参考类似背光亮度效果，将对应的值保存起来，每次系统起来后按上次配置的值配置到节点；

# 四、GAMMA 使用说明

1. 安装 gamma 调节应用 BizLineAdjustWithBcsh.apk；

2. 修改节点权限 chmod 777 /sys/class/graphics/fb0/dsp_lut

3. 打开应用调节效果；

4. 调节满意后将/data/data/com.rockchip.graphics/files/dsp_lut_bkp 文件的值填到屏的配置文件 lcd-xxx.dtsi 的 dsp_lut 中：


5.BizLineAdjustWithBcsh.apk 请从 FAE 获取。

# 五、CABC 使用说明

1. 从屏的 spec 获取屏的 gamma 值: gamma_val；

2. 将 rk3399_cabc_gamma 拷贝到 linux 环境执行：./rk3399_cabc_gamma gamma_val ；

3. 将 2 中得到的一组数字填到屏的 dtsi 文件 cabc-lut = <.....>中，可以参考 lcd-f402.dtsi 文件的配置；

4. 修改项目使用的 dts 文件，将背光的的 pwm 改成 vop0_pwm 或者 vop1_pwm，取决于屏要使用哪个 vop 显示；

5. 在使用的那个 vop 节点下面加入：rockchip,cabc_mode = <1>；

6. 开关 cabc：

关 CABC： echo "0 5 257 255 192" > /sys/class/graphics/fb0/cabc

开 CABC(normal)： echo "1 5 257 255 192" > /sys/class/graphics/fb0/cabc //调节力度较小， 建议在普通场景使用；

开 CABC(low power)： echo "1 8 260 252 180" > /sys/class/graphics/fb0/cabc //调节力度较大，建议在视频场景使用；

7. 目前支持 CABC 的平台有：RK3288W, RK3399，详情请参考从 FAE 获取的《CABC 使用说明》文档。

## 六、dump VOP 正在显示的 buffer 和让 VOP 显示 buffer 的方法

1. dump VOP 当前正在显示的 buffer

echo bin > /sys/class/graphics/fb0/dummp_buf 或者

echo bmp > /sys/class/graphics/fb0/dump_buf，执行命令后会把 VOP 当前正在显示的 buffer 写成一个文件保存到/data/dmp_buf 目录中， 如果得到的 bin 文件，可以在 PC 上安装软件 7yuv 查看，确认源数据是否正常，更多的命令可以 cat /sys/class/graphics/fb0/dump_buf 查看：

```
root@rk3399_stbvr:/ # cat /sys/class/graphics/fb
fb0/ fb1/ fb2/ fb3/ fb4/ fb5/ fb6/ fb7/ fb8/ fb9/
at /sys/class/graphics/fb0/dump_buf
bmp         -- dump buffer to bmp image
               can't support dump to single file
bin         -- dump buffer to bin image
multi       -- each dump will create new file
               only works on trace context
win=num     -- mask win to dump, default mask all
               win=1, will dump win1 buffer
               win=23, will dump win2 area3 buffer
trace=num -- trace num frames buffer dump
               this option will block buffer switch
               so recommend use with bin and win=xx

Example:
echo bmp > dump_buf;  -- dump current buf to bmp file
echo bin > dump_buf;  -- dump current buf to bin file
echo trace=50:win=1:win=23 > dump_buf
          -- dump 50 frames, dump win1 and win2 area3
          -- dump all buffer to single file
You can found dump files at /data/dmp_buf
```

2. 让 vop 显示固定的 buffer

（1）将需要显示图片的 bin 文件放到/data/fb0.bin；

（2）输入命令 echo "num xsize ysize format" > /sys/class/graphics/fb0/dsp_buf

（3）命令参数解析： num 为帧数， xzie 和 ysize 为图片的大小，format 为图片的格式，

具体什么格式对应什么值请参考/kernel/include/linux/rk_fb.h 中 HAL_PIXEL_FORMAT_XXX 的定义。

（4）更多的使用说明可以 cat /sys/class/graphics/fb0/dsp_buf 查看：

```
1|root@rk3399_stbvr:/ # cat /sys/class/graphics/fb0/dsp_buf
you can display a picture store in /data/fb0.bin use the following cmd:
echo n xsize ysize format > dsp_buf
n: picture numberxsize: picture horizontal size
ysize: picture vertical size
format:
    RGBA=1,RGBX=2,RGB=3,YUV420SP=17root@rk3399_stbvr:/ #
```

# 七、MCU 屏使用说明

1. 目前确实支持可以使用 MCU 屏的平台有 RK3188；

2. MCU 时序的配置：

假定 dclk 设置为 150M， 屏的分辨率为 480x800，最大刷新率为 55fps，Twr 为向屏发送一个 pixel 所用的时间，那么 Twr = 1000000000/(55 * 800 *480) = 47ns。

（1）Twr = mcu_total * Tcycle = mcu_total * 1000000000 / dclk

（2）screen->mcu_wrperiod = Twr

（3）dclk = 150M

通过（1）（2）（3）可以算出：

mcu_total = (screen->mcu_wrperiod*150*1000)/1000000 = 7，参考 MCU timing 配置时序：

mcu_csstart = 1；

mcu_csend = 6；

mcu_rwstart = 2;

mcu_rwend = 5;

原则是，cs start 和 end 需要在 7 个 cycle 内，rw start 和 end 需要在 cs start 和 cs end 的范围内。将上面获得的几个参数配置到 kernel/driver/video/rockchip/lcdc/rk3188_lcdc.c 函数 rk3188_load_screen 的对应参数上；

3. 具体说明请参考从 FAE 获取的《RK3188_MCU 屏使用说明》文档。

# 八、LVDS 输出说明

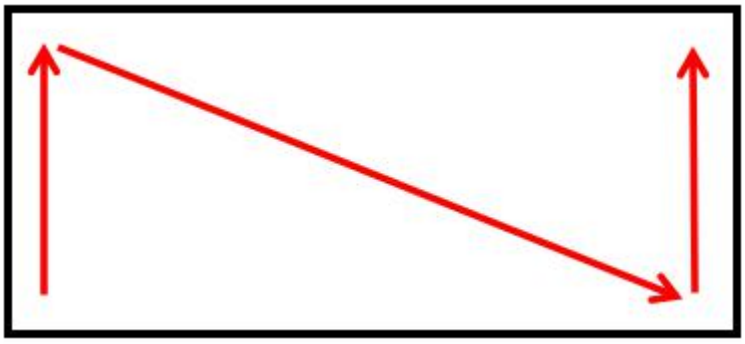
1. LVDS 输出分为 6bit/8bit/10bit 的 VESA 和 JEIDA 两个标准，每种格式的映射关系可以查看/kernel/include/dt-bindings/rkfb/rk_fb.h 中 lvds connect config 的说明；

2. 如果使用 RK3288 双 LVDS 输出，因为硬件上将两个通道接反导致显示异常的，可以使用下面的修改让主控输出前做下通道互换：

```
--- a/drivers/video/rockchip/transmitter/rk32_lvds.c
+++ b/drivers/video/rockchip/transmitter/rk32_lvds.c
@@ -89,6 +89,7 @@ static int rk32_lvds_en(void)
     val |= (screen->pin_dclk << 8) | (screen->pin_hsync << 9) |
         (screen->pin_den << 10);
     val |= (0xffff << 16);
+    val |= LVDS_FMT_1;
     grf_writel(val, RK3288_GRF_SOC_CON7);
```

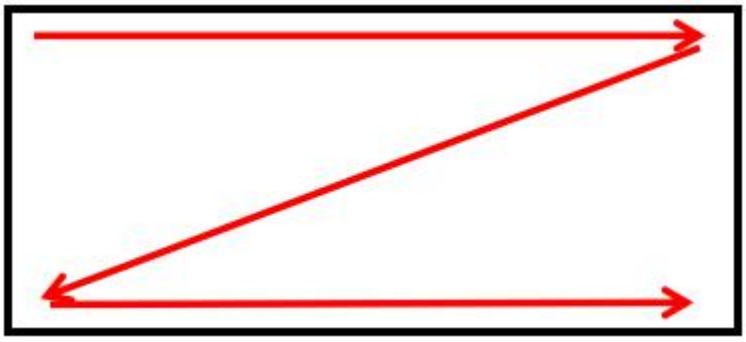
# 九、VR 产品显示配置说明

屏布局方式一：



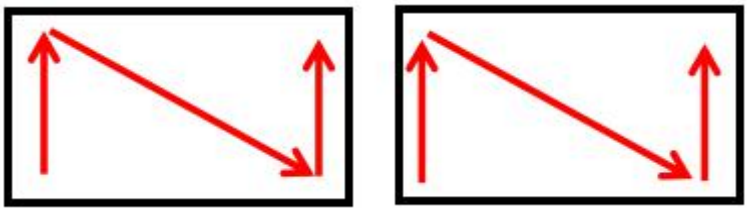
软件配置说明： 设置 dts 文件中 vop 节点下的属性 rockchip,dsp_mode = <DEFAULT_MODE>;

屏布局方式二：



软件配置说明：设置 dts 文件中 vop 节点下的属性 rockchip,dsp_mode = <DEFAULT_MODE>;

屏布局方式三：



软件配置说明：

（1）要求 2 个 mipi 屏的分辨率和扫描时序一致，最好是同一型号的两块屏；

（2）设置 dts 文件中 vop 节点下的属性 rockchip,dsp_mode = <ONE_VOP_DUAL_MIPI_VER_SCAN>;

（3）把两个单通道的 mipi 屏拼成一起看成一个双通道的 mipi，所以屏的 dtsi 文件按照双 MIPI 屏配置，主要注意以下属性配置：

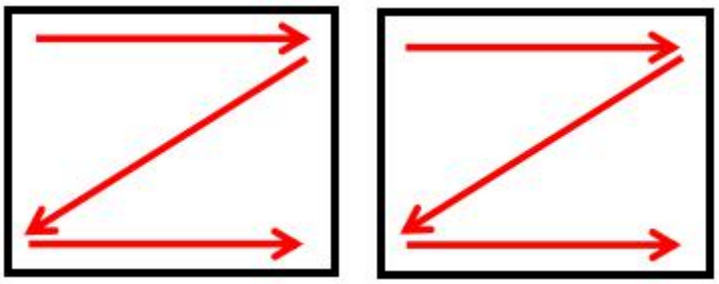
rockchip,mipi_dsi_num = <2>;

rockchip,dsi_id = <2>;

screen-type = <SCREEN_DUAL_MIPI>;

clock-frequency、hactive、hsync-len、hback-porch、hfront-porch 在屏 spec 指定的配置参数上乘以 2；

屏布局方式四：



软件配置说明：

（1）要求 2 个 mipi 屏的分辨率和扫描时序一致，最好是同一型号的两块屏；

（2）设置 dts 文件中 vop 节点下的属性 rockchip,dsp_mode = <ONE_VOP_DUAL_MIPI_HOR_SCAN>;

（3）把两个单通道的 mipi 屏拼成一起看成一个双通道的 mipi，所以屏的 dtsi 文件按照双 MIPI 屏配置，主要注意以下属性配置：

rockchip,mipi_dsi_num = <2>;

rockchip,dsi_id = <2>;

screen-type = <SCREEN_DUAL_MIPI>;

clock-frequency、hactive、hsync-len、hback-porch、hfront-porch 在屏 spec 指定的配置参数上乘以 2；

# 十、如何看 VOP 当前的图层信息

1. 输入命令 cat /sys/class/graphics/fbx/disp_info 可以看到如下信息：

```
z-order:
  win[3]
  win[2]
  win[1]
  win[0]
win0:
  state:1, fmt:ARGB888
  y_vir: 720, uv_vir:   0, x_act  :  720, y_act  : 1280, dsp_x   : 1920, dsp_y   : 1080
  x_st :   0, y_st  :   0, y_h_fac:24537, y_v_fac: 4853, uv_h_fac: 4096, uv_v_fac: 4096
  y_addr:0x13002000,    uv_addr:0x00000000
win1:
  state:0, fmt:ARGB888
  y_vir:   0, uv_vir:   0, x_act  :    1, y_act  :    1, dsp_x   :    1, dsp_y   :    1
  x_st :   0, y_st  :   0, y_h_fac:    0, y_v_fac:    0, uv_h_fac:    0, uv_v_fac:    0
  y_addr:0x00000000,    uv_addr:0x00000000
win2:
  state:1
  area0: state:1, fmt:ARGB888, dsp_x:1152, dsp_y:2048, x_st:   0, y_st:   0, y_addr:0x10000000
  area1: state:0, fmt:ARGB888, dsp_x: 320, dsp_y: 240, x_st:   0, y_st:   0, y_addr:0x00000000
  area2: state:0, fmt:ARGB888, dsp_x: 320, dsp_y: 240, x_st:   0, y_st:   0, y_addr:0x00000000
  area3: state:0, fmt:ARGB888, dsp_x: 320, dsp_y: 240, x_st:   0, y_st:   0, y_addr:0x00000000
win3:
  state:0
  area0: state:0, fmt:ARGB888, dsp_x:   1, dsp_y:   1, x_st:   0, y_st:   0, y_addr:0x00000000
  area1: state:0, fmt:ARGB888, dsp_x:   1, dsp_y:   1, x_st:   0, y_st:   0, y_addr:0x00000000
  area2: state:0, fmt:ARGB888, dsp_x:   1, dsp_y:   1, x_st:   0, y_st:   0, y_addr:0x00000000
  area3: state:0, fmt:ARGB888, dsp_x:   1, dsp_y:   1, x_st:   0, y_st:   0, y_addr:0x00000000
```

2. fbx 是 vop 对应的设备节点，对应关系请查看本文档第三点中的描述；

3. disp_info 信息说明：

z-order：表示几个图层的叠加关系，从前面 dump 的信息看，四个图层从下到上依次是 win0->win1->win2->win3；

每个图层下基本都有：

state： 表示当前图层的开关状态，0 表示图层关闭，1 表示图层打开；

fmt： 当前图层访问数据的格式；

y_vir: buffer 的虚宽，单位是 word；

uv_vir：如果是 yuv 的数据，还有 uv 的虚宽，单位是 word；

xact，yact： vop 访问数据的实宽和实高，单位是 pixel；

dsp_x，dsp_y：当前图层输出的宽和高，单位是 pixel；

x_st，y_st：当前图层显示的起点位置；

y_h_fac，y_v_fac：水平和垂直方向的缩放倍数；

uv_h_fac， uv_v_fac： uv 分量水平和垂直方向的缩放倍数；

# 十一、IO 命令使用说明

1. io 命令是最常用的硬件调试命令之一，经常被用于修改硬件的寄存器配置，在串口中输入 io 后可以看到下面的使用说明：

```
Raw memory i/o utility - $Revision: 1.5 $

io -v -1|2|4 -r|w [-l <len>] [-f <file>] <addr> [<value>]

    -v          Verbose, asks for confirmation
    -1|2|4      Sets memory access size in bytes (default byte)
    -l <len>    Length in bytes of area to access (defaults to
                one access, or whole file length)
    -r|w        Read from or Write to memory (default read)
    -f <file>   File to write on memory read, or
                to read on memory write
    <addr>      The memory address to access
    <val>       The value to write (implies -w)

Examples:
    io 0x1000                   Reads one byte from 0x1000
    io 0x1000 0x12              Writes 0x12 to location 0x1000
    io -2 -l 8 0x1000           Reads 8 words from 0x1000
    io -r -f dmp -l 100 200     Reads 100 bytes from addr 200 to file
    io -w -f img 0x10000        Writes the whole of file to memory

Note access size (-1|2|4) does not apply to file based accesses.
```

2. 经常会用到的有：

（1）读一个寄存器的值： io -4 0xff900000

（2）读 100 个寄存器的值：

```
1|root@rk3399_stbvr:/ # io -4 -r -l 100 0xff900000
ff900000:  00000000 03058896 20051800 0003f000
ff900010:  00000000 0000e442 20010200 00711c08
ff900020:  ed000000 00000000 00000000 00000000
ff900030:  3a000040 00000000 00000000 01400140
ff900040:  00000000 00000000 00ef013f 00ef013f
ff900050:  000a000a 10001000 10001000 00000000
ff900060:  00000000
```

（3）将 0xff900004 的 bit0 设置为 1

io -4 0xff900004 0x03058897

（4）从 0xff900000 读一段数据长度为 0x100 到指定的文件/data/fb0.bin

io -r -f /data/fb0.bin -l 0x100 0xff900000

（5）把文件/data/fb0.bin 写到 0xff900000 中

io -w -f /data/fb0.bin 0xff900000

# 十二、如何让 VOP 显示固定颜色

VOP 有一个背景色可以根据需要输出固定颜色，背景层只有在所有图层关闭后才可以看到，所以要 vop 输出背景层的固定颜色需要先关闭 vop 的所有图层，可以按如下步骤操作：

1. 将系统停住防止应用不停的刷新显示，如 android 系统下输入 stop 命令；

2. 查看 TRM 文档 VOP 章节关闭图层，找到类似 win0_en, win1_en, win2_en, win3_en 的寄存器，将对应位配置成 0，同时查找 REG_CFG_DONE 寄存器配置成 1 前面的配置生效；

3. cat disp_info 确认是否所有图层已经关闭，如果没有关闭回到步骤 2 确认 关闭关不已经关闭了，此时看到的全黑色就是背景色；

4. 查找 VOP 章节中 DSP_BG 寄存器，修改该寄存器的值同时写 REG_CFG_DONE 让配置生效就可以看到背景色改变；

# 十三、LVDS/RGB 输出驱动强度调节问题

1. RV1108

TTL(RGB)输出：

TTL(RGB)输出： DCLK/DEN/HSYNC/VSYNC 以及 DATA0~DATA17 通过配置成 VALUE_DRV_2MA、VALUE_DRV_4MA、VALUE_DRV_6MA、VALUE_DRV_8MA 调节驱动强度：

```
index 9a47093..7261580 100644
--- a/arch/arm/boot/dts/rv1108.dtsi
+++ b/arch/arm/boot/dts/rv1108.dtsi
@@ -1807,6 +1807,7 @@
                                                <1 GPIO_B4 RK_FUNC_1 &pcfg_pull_none>,//D15
                                                <1 GPIO_B3 RK_FUNC_1 &pcfg_pull_none>,//D16
                                                <1 GPIO_B2 RK_FUNC_1 &pcfg_pull_none>;//D17
+                               rockchip,drive = <VALUE_DRV_8MA>;
                        };

                        lcdc_mipi_gpio: lcdc-mipi-gpio {
```

2. RK312X 系列(RK3126/RK3128/RK3126B/RK3126C)

（1）TTL(RGB)输出：只能通过 LVDS PHY 寄存器调节 DATA0~DATA9 驱动强度，0x00 驱动强度最小，0xff 驱动强度最大，DATA10~DATA17 以及 DCLK、HSYNC、VSYNC、DEN 信号不可调。

```
index 500c87f..fca7e67 100755
--- a/drivers/video/rockchip/transmitter/rk31xx_lvds.c
+++ b/drivers/video/rockchip/transmitter/rk31xx_lvds.c
@@ -325,6 +325,7 @@ static void rk31xx_output_lvttl(struct rk_lvds_device *lvds,
        val = v_LVDS_MODE_EN(0) | v_TTL_MODE_EN(1) | v_MIPI_MODE_EN(0) |
                v_MSB_SEL(1) | v_DIG_INTER_RST(1);
        lvds_writel(lvds, MIPIPHY_REGE0, val);
+       lvds_writel(lvds, MIPIPHY_REGE5, 0xff);

        rk31xx_lvds_pwr_on();

diff --git a/drivers/video/rockchip/transmitter/rk31xx_lvds.h b/drivers/video/rockchip/transmitter/rk31xx_lvds.h
index 53adcc2..2dd7d90 100755
--- a/drivers/video/rockchip/transmitter/rk31xx_lvds.h
+++ b/drivers/video/rockchip/transmitter/rk31xx_lvds.h
@@ -103,6 +103,7 @@ enum {

 #define v_VOCM(x)              BITS_MASK(x, 3, 4)
 #define v_DIFF_V(x)            BITS_MASK(x, 3, 6)
+#define MIPIPHY_REGE5          0x0394

 #define MIPIPHY_REGE8          0x03a0
```

（2）LVDS 输出：共模电压和差模电压调整方法：

```
--- a/drivers/video/rockchip/transmitter/rk31xx_lvds.c
+++ b/drivers/video/rockchip/transmitter/rk31xx_lvds.c
@@ -106,7 +106,7 @@ static int rk31xx_lvds_pwr_on(void)
        if (lvds->screen.type == SCREEN_LVDS) {
                /* set VOCM 900 mv and V-DIFF 350 mv */
                lvds_msk_reg(lvds, MIPIPHY_REGE4, m_VOCM | m_DIFF_V,
-                            v_VOCM(0) | v_DIFF_V(2));
+                            v_VOCM(3) | v_DIFF_V(3));

                /* power up lvds pll and ldo */
                lvds_msk_reg(lvds, MIPIPHY_REG1,
```

3. RK3288 系列(RK3288/RK3288W)

（1）TTL(RGB)输出：DCLK/HYSNC/VSYNC/DEN 驱动强度调节：

```
--- a/arch/arm/boot/dts/rk3288-pinctrl.dtsi
+++ b/arch/arm/boot/dts/rk3288-pinctrl.dtsi
@@ -627,7 +627,7 @@
                                                <LCDC0_HSYNC_GPIO1D>,
                                                <LCDC0_VSYNC_GPIO1D>;
                                rockchip,pull = <VALUE_PULL_DISABLE>;
-                               rockchip,drive = <VALUE_DRV_DEFAULT>;
+                               rockchip,drive = <VALUE_DRV_8MA>;
                        };
```

DATA0~DATA23 驱动强度调节，0xff 驱动强度最大，0x11 驱动强度最小：

```
--- a/drivers/video/rockchip/transmitter/rk32_lvds.c
+++ b/drivers/video/rockchip/transmitter/rk32_lvds.c
@@ -102,6 +102,7 @@ static int rk32_lvds_en(void)
                lvds_writel(lvds, LVDS_CH0_REG_4, 0x3f);
                lvds_writel(lvds, LVDS_CH0_REG_5, 0x3f);
                lvds_writel(lvds, LVDS_CH0_REG_3, 0x46);
+               lvds_writel(lvds, LVDS_CH0_REG_9, 0xff);
                lvds_writel(lvds, LVDS_CH0_REG_d, 0x0a);
                lvds_writel(lvds, LVDS_CH0_REG_20,0x44);/* 44:LSB  45:MSB*/
                writel_relaxed(0x00, lvds->regs + LVDS_CFG_REG_c); /*eanble pll*/
```

（2）LVDS 输出：差模电压：LVDS_CH0_REG_8 bit[3,2]，00~11 从小到大

```
--- a/drivers/video/rockchip/transmitter/rk32_lvds.c
+++ b/drivers/video/rockchip/transmitter/rk32_lvds.c
@@ -120,6 +120,7 @@ static int rk32_lvds_en(void)
                lvds_writel(lvds, LVDS_CH0_REG_2, 0xfe);
                lvds_writel(lvds, LVDS_CH0_REG_3, 0x46);
                lvds_writel(lvds, LVDS_CH0_REG_4, 0x00);
+               lvds_writel(lvds, LVDS_CH0_REG_8, 0xf6);
                lvds_writel(lvds, LVDS_CH0_REG_d, 0x0a);
                lvds_writel(lvds, LVDS_CH0_REG_20,0x44);/* 44:LSB  45:MSB*/
                writel_relaxed(0x00, lvds->regs + LVDS_CFG_REG_c); /*eanble pll*/
diff --git a/drivers/video/rockchip/transmitter/rk32_lvds.h b/drivers/video/rockchip/transmitter/rk32_lvds.h
index ca424a7..ff2a844 100755
--- a/drivers/video/rockchip/transmitter/rk32_lvds.h
+++ b/drivers/video/rockchip/transmitter/rk32_lvds.h
@@ -7,6 +7,7 @@
 #define LVDS_CH0_REG_3                 0x0c
 #define LVDS_CH0_REG_4                 0x10
 #define LVDS_CH0_REG_5                 0x14
+#define LVDS_CH0_REG_8                 0x20
 #define LVDS_CH0_REG_9                 0x24
 #define LVDS_CFG_REG_c                 0x30
 #define LVDS_CH0_REG_d                 0x34
```

共模电压：LVDS_CH0_REG_9 bit[7,6]，配置为 00：Vcm=900mv，01：Vcm=885mv，11：Vcm=915mv。

```
index 3e8394f..237ebd6 100755
--- a/drivers/video/rockchip/transmitter/rk32_lvds.c
+++ b/drivers/video/rockchip/transmitter/rk32_lvds.c
@@ -120,6 +120,7 @@ static int rk32_lvds_en(void)
                lvds_writel(lvds, LVDS_CH0_REG_2, 0xfe);
                lvds_writel(lvds, LVDS_CH0_REG_3, 0x46);
                lvds_writel(lvds, LVDS_CH0_REG_4, 0x00);
+               lvds_writel(lvds, LVDS_CH0_REG_9, 0xc0);
                lvds_writel(lvds, LVDS_CH0_REG_d, 0x0a);
                lvds_writel(lvds, LVDS_CH0_REG_20,0x44);/* 44:LSB  45:MSB*/
                writel_relaxed(0x00, lvds->regs + LVDS_CFG_REG_c); /*eanble pll*/
```

4. RK336X 系列(RK3368/RK3368H)

（1）TTL(RGB)输出：DCLK/HYSNC/VSYNC/DEN 以及 DATA10~DATA23 驱动强度调节：

```
--- a/arch/arm64/boot/dts/rk3368.dtsi
+++ b/arch/arm64/boot/dts/rk3368.dtsi
@@ -2074,6 +2074,7 @@
                                        <0 GPIO_D6 RK_FUNC_1 &pcfg_pull_none>,//DEN
                                        <0 GPIO_D4 RK_FUNC_1 &pcfg_pull_none>,//HSYNC
                                        <0 GPIO_D5 RK_FUNC_1 &pcfg_pull_none>;//VSYN
+                       rockchip,drive = <VALUE_DRV_8MA>;
                };

                lcdc_gpio: lcdc-gpio {
```

DATA0~DATA9 通过 PHY 寄存器调节驱动强度，0x00 驱动强度最小，0xff 驱动强度最大：

```
index 500c87f..fca7e67 100755
--- a/drivers/video/rockchip/transmitter/rk31xx_lvds.c
+++ b/drivers/video/rockchip/transmitter/rk31xx_lvds.c
@@ -325,6 +325,7 @@ static void rk31xx_output_lvttl(struct rk_lvds_device *lvds,
         val = v_LVDS_MODE_EN(0) | v_TTL_MODE_EN(1) | v_MIPI_MODE_EN(0) |
                 v_MSB_SEL(1) | v_DIG_INTER_RST(1);
         lvds_writel(lvds, MIPIPHY_REGE0, val);
+        lvds_writel(lvds, MIPIPHY_REGE5, 0xff);

         rk31xx_lvds_pwr_on();

diff --git a/drivers/video/rockchip/transmitter/rk31xx_lvds.h b/drivers/video/rockchip/transmitter/rk31xx_lvds
index 53adcc2..2dd7d90 100755
--- a/drivers/video/rockchip/transmitter/rk31xx_lvds.h
+++ b/drivers/video/rockchip/transmitter/rk31xx_lvds.h
@@ -103,6 +103,7 @@ enum {

 #define v_VOCM(x)               BITS_MASK(x, 3, 4)
 #define v_DIFF_V(x)             BITS_MASK(x, 3, 6)
+#define MIPIPHY_REGE5           0x0304

 #define MIPIPHY_REGE8            0x03a0
```

LVDS 输出：共模电压和差模电压调整方法：

```
--- a/drivers/video/rockchip/transmitter/rk31xx_lvds.c
+++ b/drivers/video/rockchip/transmitter/rk31xx_lvds.c
@@ -106,7 +106,7 @@ static int rk31xx_lvds_pwr_on(void)
         if (lvds->screen.type == SCREEN_LVDS) {
                 /* set VOCM 900 mv and V-DIFF 350 mv */
                 lvds_msk_reg(lvds, MIPIPHY_REGE4, m_VOCM | m_DIFF_V,
-                             v_VOCM(0) | v_DIFF_V(2));
+                             v_VOCM(3) | v_DIFF_V(3));

                 /* power up lvds pll and ldo */
                 lvds_msk_reg(lvds, MIPIPHY_REG1,
```

# 十四、Uboot logo 和 kernel logo 配置说明

1. 要求 logo.bmp 和 logo_kernel.bmp 的大小一致，即图片分辨率一致；

2. 图片的宽高需要按偶数像素对齐；

3. 支持 bmp 24bit/8bit 的图片，带 alpha 通道的 32bit bmp 图片需要工具转换成 24bit。

注：使用 windwos 自带的画图软件打开 32bit 的 bmp 图片后按 24bit 格式另存图片后其实存下来的还是 32bit 的图片无法使用。可以使用 photoshop 软件打开图片后在图像->模式里选择 RGB 颜色 8bit/通道，然后另存图片并把 alpha 通道打勾的地方去掉，这样保存出来的图片就是 24bit 的 bmp 图片：


4. 如果图片太大，可以使用下面的命令做下压缩 ，压缩后图片质量会有一定的损失，可以用于图像比较简单的 logo 图片，对于颜色比较丰富的 logo 图片可能不太适用：

```
convert -compress rle -colors 256 logo_old.bmp logo_new.bmp
```

5. 将得到的 logo.bmp 和 logo_kernel.bmp 放到 kernel/目录下，可以用 file logo.bmp 确认图片的分辨率和格式：

```
hjc@gdiv~/work/develop/kernel
$file logo.bmp
logo.bmp: PC bitmap, Windows 3.x format, 1280 x 720 x 24
hjc@gdiv~/work/develop/kernel
$file logo_kernel.bmp
logo_kernel.bmp: PC bitmap, Windows 3.x format, 1280 x 720 x 24
```

# 十五、OLED 屏调试问题

1. 上电闪屏问题

原因分析：一些 mipi 屏的背光不是采用 pwm 控制，而是通过发送 dcs cmd 0x51 控制背光，一般屏厂为了调试方便会在初始化代码里同时把背光打开，这样会照成在初始化的过程中屏就是亮的，看到上电闪屏现象；

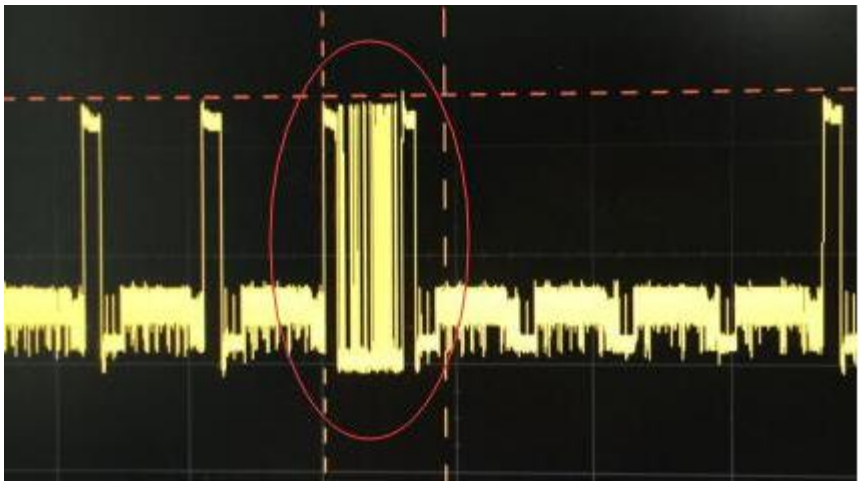
处理方法：屏的初始化代码将背光设置为 0，在背光驱动里面调用 dsi_send_packet 发送 cmd 命令打开背光，保证软件加载顺序为 vop->mipi->backlight 的顺序；

2. 调背光闪屏问题

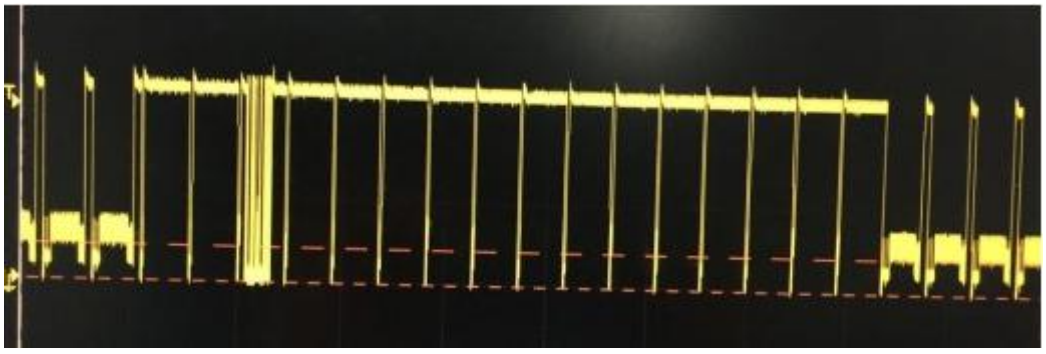
原因分析：背光调节的命令需要通过 lane0 向屏发送，而且背光的调节是随机，导致 lane0 上传输的同步信号或者有效数据被破坏造成显示花屏或者卡顿现象；

处理方法： 在背光调节前调用 rk_fb_poll_wait_frame_complete()，同时调用 local_irq_save()保证不会其他中断打断，保证在场消隐期发送背光调节的 cmd；

在显示区域发送调节背光 cmd，对后面几行数据造成影响的波形：

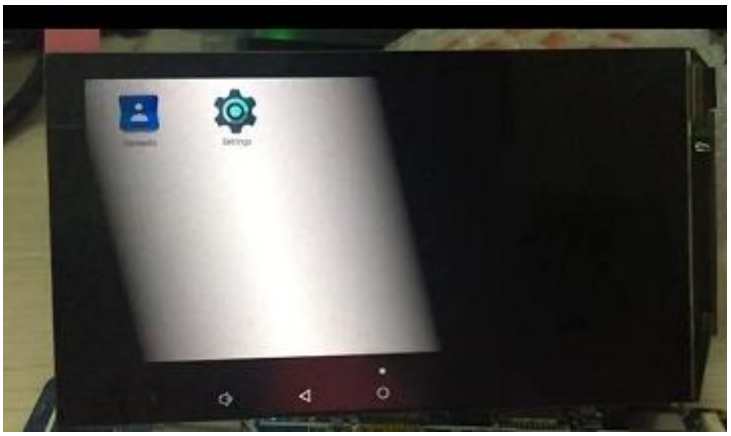


加入上面的处理方法后可以保证在场消隐期发送调节背光的 cmd：



3. 余晖时间确认

传统的 LCD 屏是全余晖的，而 OLED 一大优势就是低余晖，这可以给 VR 产品带来更舒适的体验。一般 OLED 屏都可以调整余晖值，在没有高速 DV 的情况下可以使用 iphone 手机的慢动作模式拍摄屏幕，根据亮度区间和全屏幕的比例关系初步估算出余晖时间，如下面这张图，如果屏按 60fps 扫描的话，这个时候亮部区域差不多占全屏区域一半，这样可以估算余晖时间大概为 8ms 左右。



# 十六、RGB/TLL 输出硬件连接图

| *Display mode* | *RGB Parallel 24-bit* | *RGB Parallel 18-bit* | *RGB Parallel 18-bit* | *RGB Parallel 16-bit* | *RGB Parallel 16-bit* | *ITU656 Mode0* | *ITU656 Mode1* | *ITU656 Mode2* | *MCU mode* |
|---|---|---|---|---|---|---|---|---|---|
| **Screen_face** | OUT_P888 | OUT_D888_P666 | OUT_P666 | OUT_D888_P565 | OUT_P565 | OUT_S888/OUT_S888DUMY | OUT_S888/OUT_S888DUMY | OUT_S888/OUT_S888DUMY | OUT_P888 |
| **DCLK** | DCLK | DCLK | DCLK | DCLK | DCLK | DCLK | DCLK | DCLK | RS |
| **VSYNC** | VSYNC | VSYNC | VSYNC | VSYNC | VSYNC | | | | CS |
| **HSYNC** | HSYNC | HSYNC | HSYNC | HSYNC | HSYNC | | | | WEN |
| **DEN** | DEN | DEN | DEN | DEN | DEN | | | | REN |
| **DATA** | DATA[23:0] | DATA[23:18] DATA[15:10] DATA[7:2] | DATA[17:0] | DATA[23:19] DATA[15:10] DATA[7:3] | DATA[15:0] | DATA[7:0] | DATA[15:8] | DATA[14:7] | |
| **D23** | R7 | R5 | - | R4 | - | - | - | - | D23 |
| **D22** | R6 | R4 | - | R3 | - | - | - | - | D22 |
| **D21** | R5 | R3 | - | R2 | - | - | - | - | D21 |
| **D20** | R4 | R2 | - | R1 | - | - | - | - | D20 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **D19** | R3 | R1 | - | R0 | - | - | - | - | D19 |
| **D18** | R2 | R0 | - | - | - | - | - | - | D18 |
| **D17** | R1 | - | R5 | - | - | - | - | - | D17 |
| **D16** | R0 | - | R4 | - | - | - | - | - | D16 |
| **D15** | G7 | G5 | R3 | G5 | R4 | - | D7 | - | D15 |
| **D14** | G6 | G4 | R2 | G4 | R3 | - | D6 | D7 | D14 |
| **D13** | G5 | G3 | R1 | G3 | R2 | - | D5 | D6 | D13 |
| **D12** | G4 | G2 | R0 | G2 | R1 | - | D4 | D5 | D12 |
| **D11** | G3 | G1 | G5 | G1 | R0 | - | D3 | D4 | D11 |
| **D10** | G2 | G0 | G4 | G0 | G5 | - | D2 | D3 | D10 |
| **D9** | G1 | - | G3 | - | G4 | - | D1 | D2 | D9 |
| **D8** | G0 | - | G2 | - | G3 | - | D0 | D1 | D8 |
| **D7** | B7 | B5 | G1 | B4 | G2 | D7 | - | D0 | D7 |
| **D6** | B6 | B4 | G0 | B3 | G1 | D6 | - | - | D6 |
| **D5** | B5 | B3 | B5 | B2 | G0 | D5 | - | - | D5 |
| **D4** | B4 | B2 | B4 | B1 | B4 | D4 | - | - | D4 |
| **D3** | B3 | B1 | B3 | B0 | B3 | D3 | - | - | D3 |
| **D2** | B2 | B0 | B2 | - | B2 | D2 | - | - | D2 |
| **D1** | B1 | - | B1 | - | B1 | D1 | - | - | D1 |
| **D0** | B0 | - | B0 | - | B0 | D0 | - | - | D0 |