

密级状态：绝密( ) 秘密( ) 内部( ) 公开( √ )

# Rockchip 平台 Linux 3.10 上 MIPI 驱动介绍和调试说明

(技术部，系统产品一部)

文件状态：  [ ] 正在修改  [√] 正式发布	当前版本：	V1.0
	作 者：	XBL, GLN, OEH, LCS
	完成日期：	2015-12-16
	审 核：	ZY, ZYY, ZXZ
	完成日期：	2015-12-25

福州瑞芯微电子有限公司

Fuzhou Rockchips Electronics Co . , Ltd

(版本所有, 翻版必究)

## 版 本 历 史

版本号	作者	修改日期	修改说明	备注
V1.0	XBL, GLN, OEH, LCS	2015.12.16	初稿	

## 目录

1	代码目录说明 .....	3
2	代码结构框图 .....	3
3	Menuconfig 配置 .....	5
4	屏参文件配置与说明 .....	6
4.1	MIPI Host 配置 .....	7
4.2	屏电源控制配置 .....	7
4.3	屏初始化序列 .....	9
4.4	屏参 .....	10
5	板级文件配置 .....	11
6	MIPI 屏调试流程 .....	12
7	常见问题以及解决 .....	15

## 1 代码目录说明

Rockchip MIPI 驱动相关文件以及说明如下所示：

drivers/video/rockchip/transmitter/

  |\_ rk32\_mipi\_dsi.c /\* MIPI 驱动主体文件 \*/

  |\_ rk32\_mipi\_dsi.h /\* 寄存器以及结构体的定义 \*/

  |\_ mipi\_dsi.c /\* 封装的函数指针接口函数，供 lcd\_mipi.c 调用，函数的具体实现  
在 rk32\_mipi\_dsi.c 中 \*/

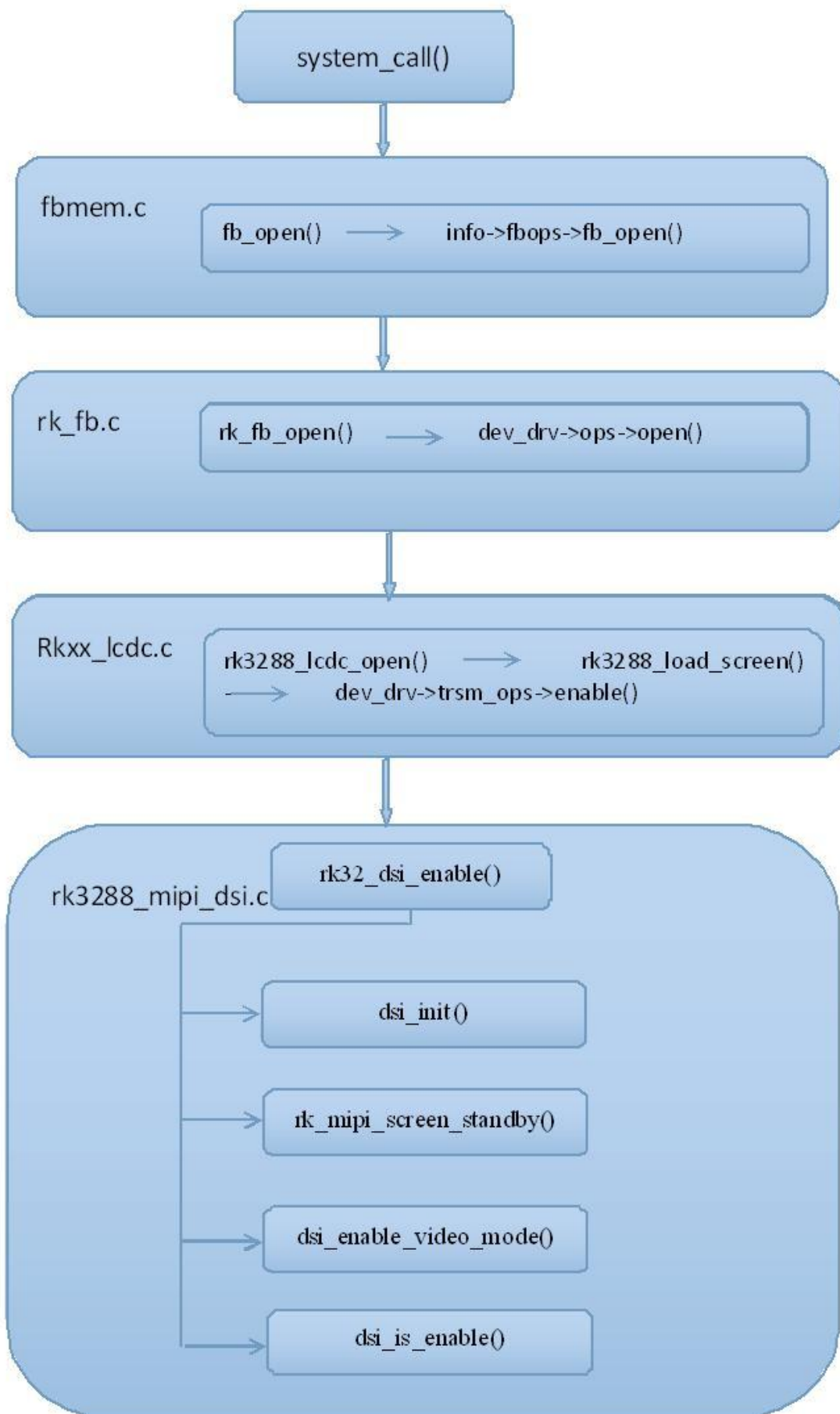
  |\_ mipi\_dsi.h /\* mipi 协议相关的宏定义以及函数指针结构体定义 \*/

drivers/video/rockchip/screen/

  |\_ lcd\_mipi.c /\* 屏参 dtsi 文件的解析 \*/

## 2 代码结构框图

无论是正常的开机流程以及休眠唤醒的流程，显示相关的模块都是要和 fb 以及 vop(lcdc)交互的。rk32\_dsi\_enable()和 rk32\_dsi\_disable()函数作为 MIPI 和 vop 之间的交互的总入口函数。另外还有一对 rk32\_mipi\_power\_up\_DDR() 和 rk32\_mipi\_power\_down\_DDR()函数是单独供 ddr 变频的时候使用，正常的开机以及休眠唤醒流程不走这两个函数。开机流程图如下图所示：



**dsi\_init()**：该函数主要实现 mipi host 和 phy 的上电以及初始化工作。

**rk\_mipi\_screen\_standby()**：屏的供电以及屏的初始化工作（屏初始化命令的发送）。

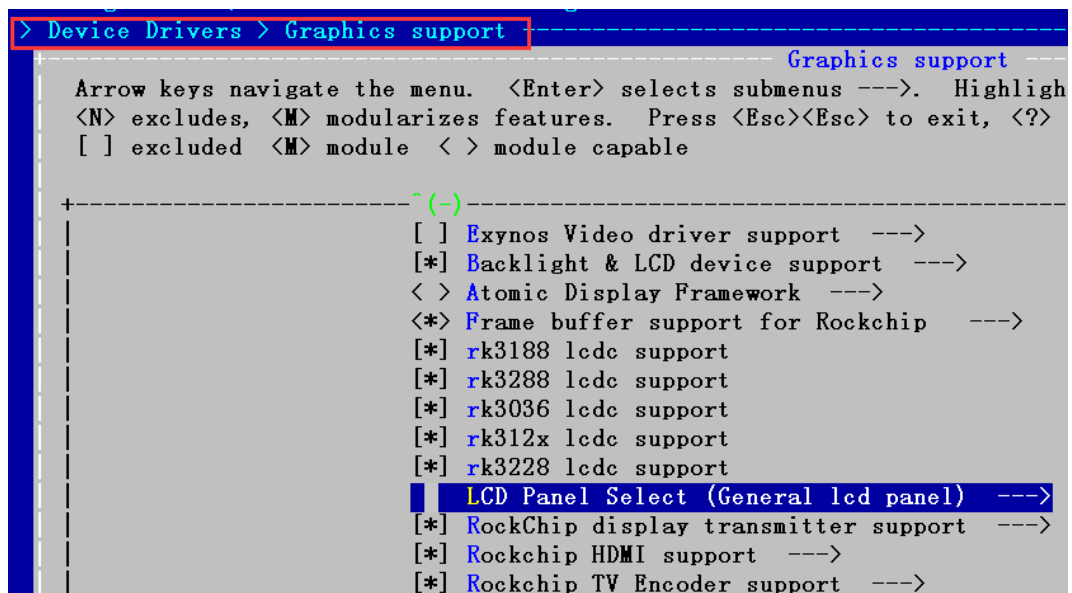
**dsi\_enable\_video\_mode()**：command 模式和 video 模式的切换，发送命令一般是在 command 模式下，正常数据传输是在 video 模式下。

**dsi\_is\_enable()**：mipi host 的关闭与开启。

除了开机，休眠唤醒的在 lcdc 中的调用是 rk3368\_lcdc\_early\_suspend() 和 rk3368\_lcdc\_early\_resume()。

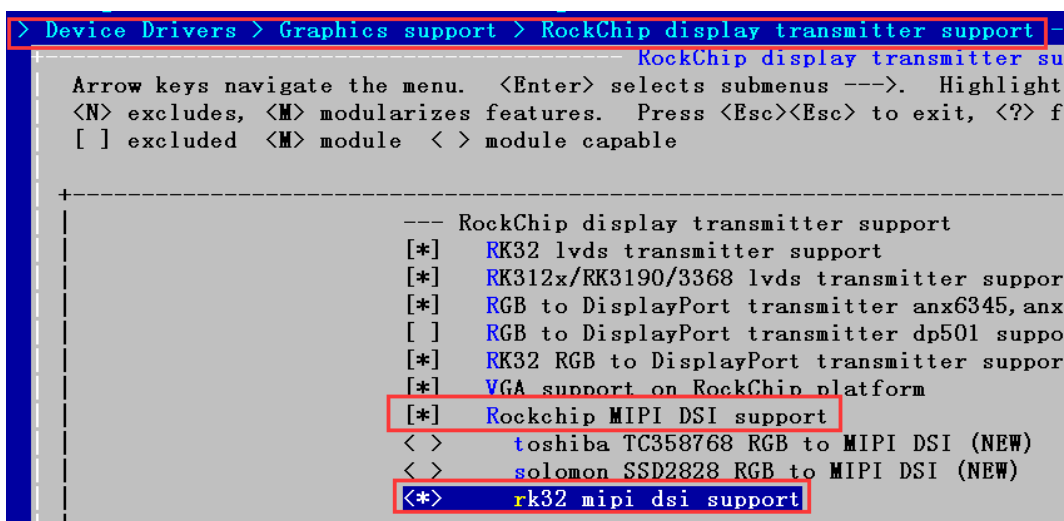
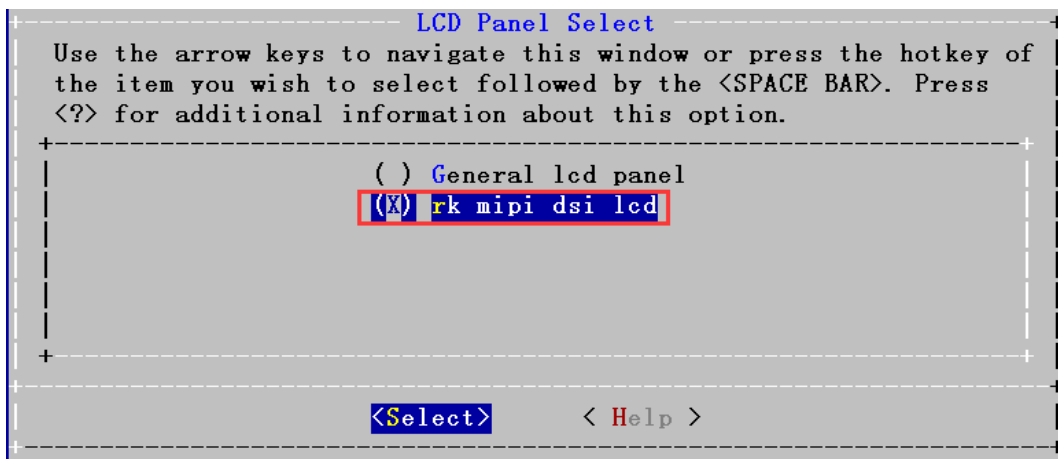
### 3 Menuconfig 配置

要调用到 MIPI 驱动需要在 menuconfig 里面进行配置，目前在开发分支里面已经默认配置好，开启和关闭由板级 dts 文件来控制。配置如下列图所示（红色标记表示配置路径和需要配置的选项）：



```
> Device Drivers > Graphics support -----
Graphics support
Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highligh
<N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?>
[ ] excluded <M> module < > module capable

-----~(-)-----
[ ] Exynos Video driver support --->
[*] Backlight & LCD device support --->
< > Atomic Display Framework --->
<*> Frame buffer support for Rockchip --->
[*] rk3188 lcdc support
[*] rk3288 lcdc support
[*] rk3036 lcdc support
[*] rk312x lcdc support
[*] rk3228 lcdc support
[ ] LCD Panel Select (General lcd panel) --->
[*] RockChip display transmitter support --->
[*] Rockchip HDMI support --->
[*] Rockchip TV Encoder support --->
```



## 4 屏参文件配置与说明

屏参文件包含四个部分：mipi host 配置、屏电源控制配置、屏初始化序列和屏参。

屏参文件的解析：mipi host 配置、屏电源控制配置、屏初始化序列三部分是在 drivers/video/rockchip/screen/lcd\_mipi.c 中解析的，最后的屏参是在 drivers/video/of\_display\_timing.c 中解析。因为该部分信息 mipi/edp/lvds/hdmi 之类显示设备都存在，所以在统一的地方进行解析。

屏参文件所在目录为 arch/arm/boot/dts/中。

## 4.1 MIPI Host 配置

Mipi host 配置结构如下所示：

```
disp_mipi_init: mipi_dsi_init{
    compatible = "rockchip,mipi_dsi_init";
    rockchip,screen_init    = <0>;
    rockchip,dsi_lane       = <4>;
    rockchip,dsi_hs_clk     = <1000>;
    rockchip,mipi_dsi_num   = <1>;
};
```

**screen\_init**：表示屏是否需要初始化，如果需要则置为 1。

**dsi\_lane**：mipi 数据传输需要几条数据 lane，这个一般根据原理图和 mipi 屏的规格书来配置。这个指的是每个 mipi 的数据 lane 数。譬如如果是双 mipi，每个 mipi 为 4 lane。那么此处仍然设置为 4。

**dsi\_hs\_clk**：屏 ddr clk，表示一条数据 lane 的传输速率，单位为 Mb/s。有个大概的计算公式： $100 + H\_total * V\_total * fps * 3 * 8 / lanes$ 。

H\_total, V\_total 包括 active, bp, fp 和 sync-len 的和；

fps 为帧率，刚调试一款屏时，fps 为 50 多帧就好，然后慢慢抬高；

3 为一个像素点为 rgb 3 个字节；

8 为 8 bits；

lanes 为  $(dsi\_lane * mipi\_dsi\_num)$ ；

100 为实际的结果要比理论值大 100M 左右。

上面计算得到的值只是大概值并非精确的值，但是对于一般的屏都适用，对于部分屏需要微调该值。

**mipi\_dsi\_num**：单 mipi 还是双 mipi，也是根据原理图和屏幕规格书来配置的。如果是双 mipi 则置为 2。单 MIPI 的屏 SDK 代码默认设置的是 MIPI\_TX，所以单 MIPI 是接 MIPI\_TX 这一路，双 MIPI 接法：MIPI\_TX 这路接左屏，MIPI\_TX/RX 这一路接右屏。

## 4.2 屏电源控制配置

屏的电源控制配置可以放在 dtsi 中的 lcdc 模块中去配置，而且默认放在 lcdc 中去配置。原因有二：一、我们 mipi/edp/lvds 电源控制部分是一致的，放在 lcdc 中便于兼容；二、客户一般都会基于我们的硬件参考资料去布板，所以屏的电源控制部分基本都和我们



SDK 一致。但是如果客户的电源控制部分和我们 SDK 有差异，那么可以在屏参文件单独配置。屏电源控制配置的结构如下：

```
disp_mipi_power_ctr: mipi_power_ctr {
    compatible="rockchip,mipi_power_ctr";
    mipi_lcd_rst:mipi_lcd_rst{
        compatible = "rockchip,lcd_rst";
        rockchip,gpios = <&gpio3 GPIO_D6 GPIO_ACTIVE_HIGH>;
        rockchip,delay = <10>;
    };
    mipi_lcd_en:mipi_lcd_en {
        compatible = "rockchip,lcd_en";
        rockchip,gpios = <&gpio7 GPIO_A3 GPIO_ACTIVE_HIGH>;
        rockchip,delay = <10>;
    };
    mipi_lcd_cs:mipi_lcd_cs {
        compatible = "rockchip,lcd_cs";
        rockchip,gpios = <&gpio7 GPIO_A4 GPIO_ACTIVE_HIGH>;
        rockchip,delay = <10>;
    };
};
```

电源配置的 gpios 需要根据原理图来配置 lcd\_en 等各对应哪路 gpio，另外还可能存在不需要三路电源控制的情况。

delay 部分是操作完后的延迟时间，这个关系到 mipi 的上电时序。在需要的时候要对延时时间进行调整。

对应的操作函数：driver/video/rockchip/screen/lcd\_mipi.c 的 rk\_mipi\_screen\_pwr\_enable(), rk\_mipi\_screen\_pwr\_disable()。

uboot 中对应的操作也是这两个函数。

dtsi 文件中 lcdc 的配置如下：

```
&lcdc0 {
    status = "okay";
    rockchip,mirror = <NO_MIRROR>;
    rockchip,cabc_mode = <0>;
    power_ctr: power_ctr {
        rockchip,debug = <0>;
        lcd_en:lcd_en {
            rockchip,power_type = <GPIO>;
            gpios = <&gpio7 GPIO_A3 GPIO_ACTIVE_HIGH>;
            rockchip,delay = <10>;
        };
    };
};
```

```
lcd_cs:lcd_cs {
    rockchip,power_type = <GPIO>;
    gpios = <&gpio7 GPIO_A4 GPIO_ACTIVE_HIGH>;
    rockchip,delay = <10>;
};

lcd_rst:lcd_rst {
    rockchip,power_type = <GPIO>;
    gpios = <&gpio3 GPIO_D6 GPIO_ACTIVE_HIGH>;
    rockchip,delay = <5>;
};
};
```

对应的操作函数：drivers\video\rockchip\rk\_fb.c 里的 rk\_disp\_pwr\_enable()和 rk\_disp\_pwr\_disable()。

uboot 中对应的操作函数是：drivers\video\rockchip\_fb.c 里的 rk\_fb\_pwr\_enable()，被 lcd\_ctrl\_init()函数调用。

## 4.3 屏初始化序列

屏是否需要初始化要根据屏的规格书来确定，如果需要初始化，可以像屏厂要提供初始化序列。

屏初始化命令发送在 driver/video/rockchip/screen/lcd\_mipi.c 的 rk\_mipi\_screen\_cmd\_init()中完成。

屏初始化序列的结构如下所示：

```
disp_mipi_init_cmds: screen-on-cmds {
    compatible = "rockchip,screen-on-cmds";
    rockchip,cmd_debug = <1>;
    rockchip,on-cmds1 {
        compatible = "rockchip,on-cmds";
        rockchip,cmd_type = <HSDT>;
        rockchip,dsi_id = <2>;
        rockchip,cmd = <0xb0 0x02>;
        rockchip,cmd_delay = <0>;
    };
};
```

**rockchip,on-cmds1 结构**：便是一条初始化命令的结构，如果有多条初始化命令，那

么需要多个命令结构。

**cmd\_type** : 命令是在 low power(LPDT)还是 high speed(HSDT)下发送。

**dsi\_id** : 命令通过哪个 mipi 发送。0 表示在 mipi0 发送, 1 表示在 mipi1 发送, 2 表示双 mipi 同时发送。因为很少出现单独使用 mipi1 的情况, 所以对于单 mipi, 这个值默认是 0; 对于双 mipi, 这个值是 2。

**cmd** : 初始化命令。格式: 命令类型 (如 0x05/0x15/0x39) + 命令 + 参数。具体细节见 mipi 协议文档。

**cmd\_delay** : 命令发完后延迟时间, 这个根据屏厂给的初始化序列来配置。

另外有一些特殊标准命令 (0x22,0x32), 某些特殊屏在有可能需要, 需要修改控制器来发送, 屏厂确认需要可以联系我们这里修改。

如果屏幕不需要初始化, 也就是 rockchip,screen\_init 为 0 的情况。不需要初始化并不是表示没有发送命令。在不需要初始化的情况下, 我们等 mipi host, phy 供电初始化完以及屏的供电结束以后会按照 mipi 协议发送 exit\_sleep\_mode 和 set\_display\_on 命令。

Exit\_sleep\_mode 表示退出 sleep 模式。

Set\_display\_on 表示告诉显示设备 (屏) 可以开始显示图像数据了。

## 4.4 屏参

屏参文件包括屏幕的格式, dclk, 时序等信息。如下所示, 其中红色标记的是需要重点关注的参数:

```
disp_timings: display-timings {
    native-mode = <&timing0>;
    compatible = "rockchip,display-timings";
    timing0: timing0 {
        screen-type = <SCREEN_MIPI>;
        lvds-format = <LVDS_8BIT_2>;
        out-face    = <OUT_P888>;
        clock-frequency = <145000000>;
        hactive = <1920>;
        vactive = <1200>;
        hback-porch = <16>;
        hfront-porch = <24>;
        vback-porch = <10>;
        vfront-porch = <16>;
        hsync-len = <10>;
```

```
vsync-len = <3>;
hsync-active = <0>;
vsync-active = <0>;
de-active = <0>;
pixelclk-active = <0>;
swap-rb = <0>;
swap-rg = <0>;
swap-gb = <0>;
}
}
```

screen-type : 屏幕类型，对于 mipi 屏来说有两种选择：单 mipi (SCREEN\_MIPI);  
双 mipi (SCREEN\_DUAL\_MIPI);

lvds-format : lvds 数据格式。对于 mipi 来说是无效参数，不用配置

out-face : 屏幕接线格式

上述三个参数的取值在 include/dt-bindings/rkfb/rk\_fb.h 中定义。

clock-frequency : dclk 频率，单位为 HZ，一般屏的规格书中有，如果没有可以通过公式计算： $H*V(\text{包括同步信号})*fps$

Hactive : 水平有效像素

Vactive : 垂直有效像素

hback-porch/hfront-porch/hsync-len : 水平同步信号

vback-porch/vfront-porch/vsync-len : 垂直同步信号

hsync-active、vsync-active、de-active、pixelclk-active : 分别为 hync, vsync, DEN, dclk 的极性控制。置 1 将对极性进行翻转。

swap-rb、swap-rg、swap-gb : 设 1 将对对应的颜色进行翻转。

## 5 板级文件配置

屏参文件配置好以后，需要在板级文件中包含这个屏参文件：

```
#include "lcd-4ZB02M01-mipi.dtsi"
```

因为在芯片 dtsi 中默认是关闭的，所以需要在板级板级文件中开启 mipi host：

```
&dsihost0 {
    status = "okay";
};
&dsihost1 {
```

```
status = "okay";  
};
```

如果是单 mipi，则只需开启对应的一个即可，双 mipi 两个都需要开启。

关于各个芯片之间的差异如下表所示：

芯片平台	是否内部集成	单/双 MIPI	是否支持长包屏	其他注意事项
RK29xx/ 30xx/31 88	外接 RK618 等转换芯片	单	否	
RK3288	内部集成	双	是	双 mipi 屏：phy0(rx) 接左半屏，phy1 (rx/tx)接右半屏
RK312x	内部集成	单	是	
RK3368	内部集成	单	是	

## 6 MIPI 屏调试流程

- (1) 根据原理图和屏的规格书配置屏参文件，具体细节见上面第 4 节。
- (2) 板级文件中包含屏参文件并开启 mipi dsi host，具体见上面第 5 节。
- (3) menuconfig 里面配置打开 mipi，具体见上面第 3 节。

上面三步骤是软件上最基本的配置，部分 mipi 屏只需要配置上面三步就可以点亮，如果屏没有点亮，那么可以按接下来的步骤尝试：

- (4) 查看背光是否有亮，如果背光没亮，检查背光电源是否正常，PWM 通道是否配置正确。
- (5) 测量 mipi 的供电是否正常，譬如 3368 中给 mipi 的三路供电分别为 1.0V，1.8V 和 3.3V。
- (6) 对于有初始化命令的屏，可以检查一下命令是否发送成功。检查的方法可以用示波器测量和测量屏端电压值。初始化命令发送成功后，屏驱动 IC 上某些电压会升高（具体哪几个电压需要屏驱动 IC 确认）。
- (7) uboot-logo-on 开启或者关闭测试，因为 uboot-logo-on 开启和关闭在开机过程中走的流程不一样，开启时加载的是 uboot 中的 mipi 驱动，不开启是加载的是 kernel 中的驱

动；另外两者的时序也会有差异。

(8) 如果开机没有显示，可以看 log 等系统起来以后测试一下休眠唤醒，以断定是否为时序问题。因为开机的时序和休眠唤醒时序会有差异，有时候会因为这个差异导致开机没显示，但是休眠唤醒可以显示的情况。此时就需要校正开机的时序。

(9) 开启 mipi debug 接口：

driver/video/rockchip/transmitter/rk32\_mipi\_dsi.c 中的：

```
#define MIPI_DBG(x...) printk(KERN_INFO x)。
```

driver/video/rockchip/screen/lcd\_mipi.c 中的：

```
#define MIPI_SCREEN_DBG(x...) printk(KERN_ERR x)。
```

看看 log 中是否有异常，譬如 probe 函数是否正常；是否有调用到 rk32\_dsi\_enable() 函数，该函数为 lcdc 调用 mipi 的入口函数；初始化 mipi 的过程中是否有报错等等。

(10) 可以查看一下 fb 中是否有数据：

3368 的命令为

```
echo bmp > sys/class/graphics/fb0/dump_buf 或者
```

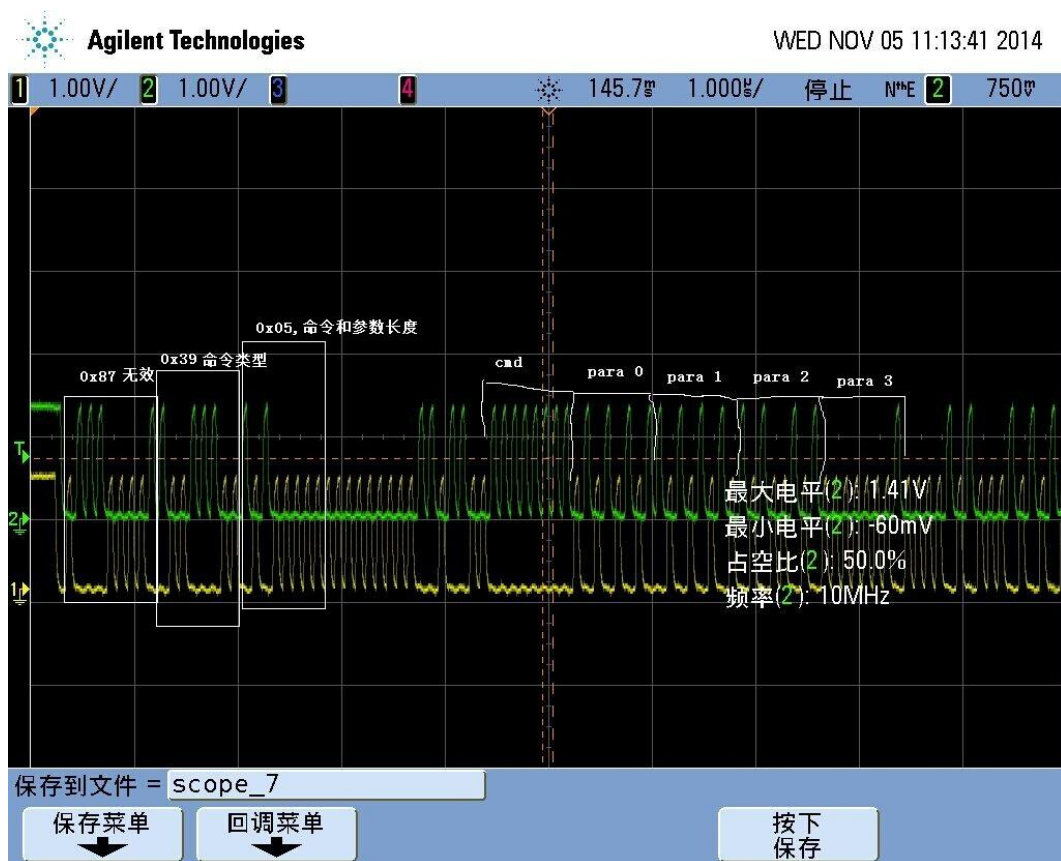
```
echo bin > sys/class/graphics/fb0/dump_buf
```

312x 的命令为 `cat /dev/graphics/fb0 > /data/fb0.yuv`

通过软件，譬如 7YUV 读取输出的 bmp 文件或者 bin 文件，如果有软件能正常显示画面，说明是 mipi 问题，如果软件不能正常显示画面，说明 fb 刷下来的数据有问题

(11) 如果上述方式都验证过了，那么就需要用示波器来测量了，示波器的带宽最少 500M。

- i. 先测量上电时序是否符合屏幕规格书中的定义。
- ii. 测量 data 和 clk 的波形是否正常。
- iii. 如果也正常，那就需要确认 mipi phy 是否把初始化命令正确发送出来。用差分探头在单端模式下抓 mipi phy 的 lane0N 和 lane0P。



上面是长包的波形:

第一个字节 0x87(固定是 0x87)丢弃;第二个字节 0x39 为命令的 type;第三个字节 0x05 为命令和参数的长度;第四,第五字节为校验,不管;第六字节 cmd 为命令;后面跟的是命令的参数。

如果是短包:第一个字节 0x87 丢弃;第二个字节 0x15 为命令的 type;第三字节为命令,第四字节为参数,第五字节为校验。

※目前 RK 的芯片中,只有 RK618, RK3288, RK3368 支持长包, RK312X ECO 之前不支持长包, ECO 之后的芯片支持长包。

(12) 如果命令也确认过是正常的,但是屏还是不亮,这个时候就要分析 mipi 协议了。

主要是以下几个时间是否符合屏的要求:

HS Entry: DATA TLPX

HS Entry: DATA TX THS-PREPARE+THS-ZERO

HS Exit: DATA TX THS-TRAIL

HS Exit: DATA TX TREOT

HS Exit: DATA TX TEOT

HS Exit: DATA TX THS-EXIT

这些参数不建议随意修改，务必先测量这些时序段后确认需要修改在调整，调整的代码在 rk32\_phy\_init 接口中。如 THS-PREPARE（调试中发现此参数修改较多），修改在 rk32\_mipi\_dsi.c 文件中：

test\_data[0] = 0x80 | 85; 调整 85 这个值。

rk32\_dwc\_phy\_test\_wr(dsi,code\_hstxdatalanepreparestatetime,test\_data,1);

## 7 常见问题以及解决

① uboot 开机有显示，kernel 起来后，休眠唤醒后没有显示：

- A. 确认 kernel 里的开机时序和 uboot 是否一样。
- B. 注释一下两处。

```
904      /* enable non-continued function */
905      /* rk32_dsi_set_bits(dsi, 1, auto_clklane_ctrl); */
906      /*
```

```
880      /* enable send command in low power mode */
881      /*
882      rk32_dsi_set_bits(dsi, 4, outvact_lpcmd_time);
883      rk32_dsi_set_bits(dsi, 4, invact_lpcmd_time);
884      rk32_dsi_set_bits(dsi, 1, lp_cmd_en);
885      */
```

② 开机显示正常，休眠唤醒概率性有背光无显示

- A. 测量给 mipi 的供电电压是否精准，譬如 1.8V，测量出来的值是否偏差很小。如果偏差较大那么需要 pmic 那边校准。
- B. 抬高 logic 电压。
- C. 部分屏对 dsi\_hs\_clk 要求较高，需要微调 dsi\_hs\_clk 的值。

③ 双 mipi 屏屏幕中间位置显示异常：可能原因是屏 overlay(overlap)的值出错，可以去屏的规格书中去寻找，也可以直接咨询屏厂。

④ 加强抗静电能力：在屏支持非连续模式的前提下，可以选择非连续模式。在 rk32\_mipi\_dsi.c 文件中

```
/* enable non-continued function */
rk32_dsi_set_bits(dsi, 1, auto_clklane_ctrl);
```

⑤ 使用 HX8389A/B 的屏开机显示正常，但唤醒黑屏；相同机器 HX8389C 开机和唤醒均正常。



HX8389C 屏 IC 默认有屏蔽 shut down (22h) 指令, HX8389B 则可能没有, 修改:

drivers/video/rockchip/transmitter/rk32\_mipi\_dsi.c

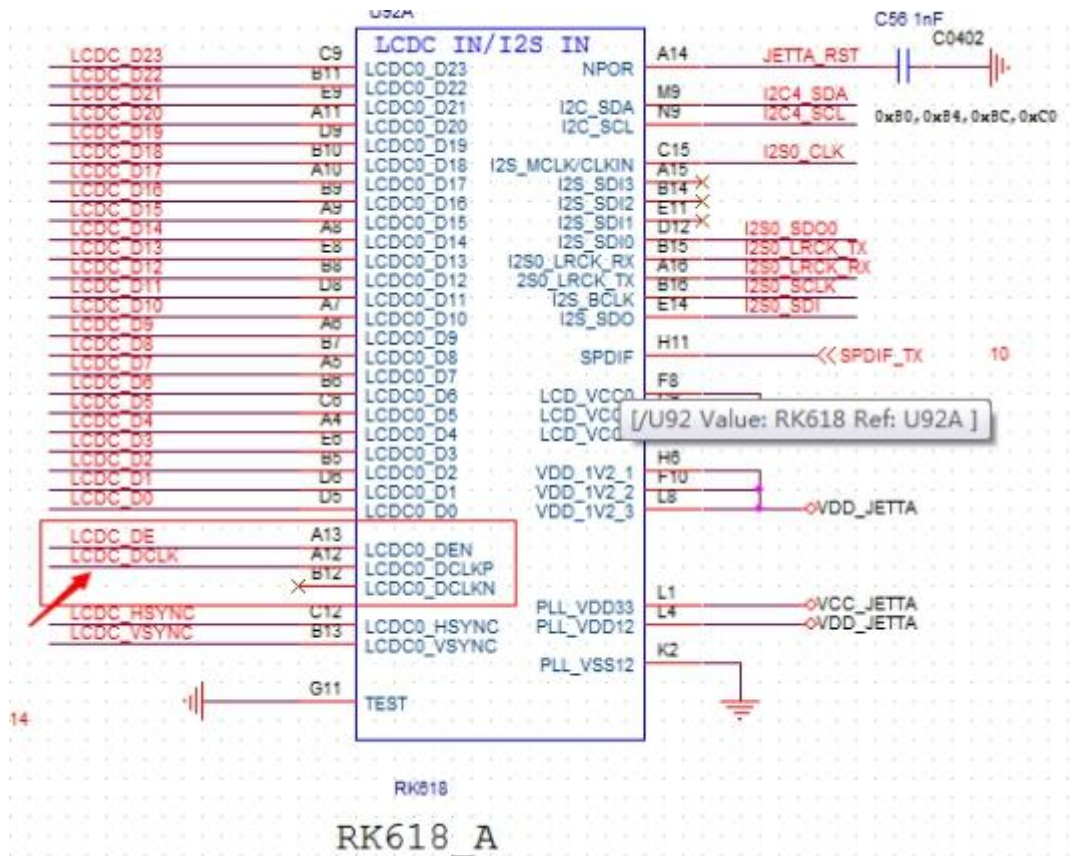
```
rk32_dsi_set_bits(dsi, 1, colorm_active_low);
rk32_dsi_set_bits(dsi, 1, shutd_active_low);
```

这会发送 mipi 指令 0x22 及 0x12

改为去除:

```
rk32_dsi_set_bits(dsi, 0, colorm_active_low);
rk32_dsi_set_bits(dsi, 0, shutd_active_low);
```

⑥ 对于 RK3188 +rk618 配置点 mipi 屏这样的方案, 如果按照上述调试步骤仍然不能点亮屏, 并且测量发现屏的数据线上没有数据波形, 请确认下 RK618 输入的时钟引脚是不是正确, 例如下图可能导致屏不亮。



公版参考设计

