屏幕对接

使用指南

文档版本　04

发布日期　2019-09-12

商标声明

 、HISILICON、海思和其他海思商标均为海思技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

上海海思技术有限公司

地址：深圳市龙岗区坂田华为总部办公楼　　邮编：518129

网址：http://www.hisilicon.com/cn/

客户服务邮箱：support@hisilicon.com

前　言

概述

本文档介绍如何在海思芯片解决方案上开发调试 RGB LCD&MIPI LCD 屏，以帮助客户有序快速开发 RGB LCD&MIPI LCD 业务。

注意：关于屏幕的称呼差异较大，本文中 RGB LCD 屏指的是以 RGB 为接口的 LCD 屏幕，一般指 RGB 屏。MIPI LCD 指的是以 MIPI DSI 为接口的 LCD 屏，一般指 MIPI 屏。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3556A	V100
Hi3559A	V100
Hi3559C	V100
Hi3519A	V100
Hi3516C	V500
Hi3516D	V300
Hi3559	V200
Hi3556	V200
Hi3516A	V300
Hi3516E	V200
Hi3516E	V300
Hi3518E	V300
Hi3516D	V200

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

文档版本 04 (2019-09-12)

1.4.5、2.2.1 和 2.4.12 小节涉及修改

文档版本 03 (2019-07-30)

2.4.6 小节涉及修改

新增 2.4.11、2.4.14 小节。

文档版本 02(2019-06-20)

2.2.2、2.2.4、2.3.3、2.3.8 小节涉及修改。

文档版本 01(2019-05-25)

2.2.4 和 2.3.3 小节涉及修改

文档版本 00B01(2019-04-28)

第一次临时版本发布。

目 录

前 言 i

1 RGB LCD 屏幕对接指南 1

1.1 环境准备 1
1.1.1 RGB 屏幕接口介绍 1
1.1.2 硬件连线确认 3
1.1.3 控制接口使能确认 3
1.2 配置步骤 7
1.2.1 配置管脚复用和驱动能力 9
1.2.2 配置 RGB LCD 屏复位 9
1.2.3 配置背光 10
1.2.4 配置 RGB LCD 屏初始化序列 10
1.2.5 配置 VO 输出序列 10
1.2.6 配置 VO 输出时钟 13
1.3 屏幕验证与调试 16
1.3.1 管脚复用与驱动能力确认 16
1.3.2 复位操作流程确认 17
1.3.3 背光控制确认 18
1.3.4 控制命令通路确认 18
1.3.5 VO 输出时钟和时序确认 18
1.3.6 VO Colorbar 调试 19
1.4 显示屏调试 FAQ 21
1.4.1 屏幕全黑，毫无反应 21
1.4.2 屏幕显示位置错误 21
1.4.3 屏幕显示裂屏 22
1.4.4 屏幕显示大小错误 22
1.4.5 屏幕显示颜色异常 22
1.4.6 屏幕显示效果不够满意 24
1.4.7 屏幕画面转换时出现残影 24
1.4.8 显示帧率低/画面卡顿 24
1.4.9 显示画面较暗，但背光正常 25

2 MIPI LCD 屏对接指引 1
2.1 环境准备 1
2.1.1 MIPI DSI 屏幕接口介绍 1
2.1.2 硬件连线确认 2
2.2 配置步骤 2
2.2.1 配置管脚复用和驱动能力 4
2.2.2 配置 MIPI 屏复位 4
2.2.3 配置背光 4
2.2.4 配置 MIPI 屏幕 4
2.2.5 配置 VO 输出序列 14
2.2.6 配置 VO 输出时钟 18
2.3 配置验证与调试 20
2.3.1 管脚复用与驱动能力确认 20
2.3.2 复位操作流程确认 20
2.3.3 屏幕控制命令通路验证 22
2.3.4 背光控制 23
2.3.5 VO 输出时钟和时序确认 23
2.3.6 VO COLORBAR 25
2.3.7 MIPI 配置属性查看方法 26
2.3.8 MIPI Tx colorbar 调试方法 28
2.4 显示屏调试 FAQ 30
2.4.1 屏幕全黑，毫无反应 30
2.4.2 屏幕显示位置错误 31
2.4.3 屏幕显示裂屏 31
2.4.4 屏幕显示图像出现错乱 32
2.4.5 屏幕显示大小错误 32
2.4.6 屏幕显示颜色异常 32
2.4.7 屏幕显示效果不够满意 33
2.4.8 屏幕画面转换时出现残影 34
2.4.9 显示帧率低/画面卡顿 34
2.4.10 显示画面较暗，但背光正常 34
2.4.11 退出显示业务后重新启动，RGB 屏无法显示 35
2.4.12 MIPI 检测不到前端时序行信息 35
2.4.13 设置 MIPI 设备属性 phy_data_rate 与实际检测出来的差别较大 35
2.4.14 MIPI D-PHY 时序与屏幕端匹配确认 36

图目录

图 1-1 RGB 屏幕接口示意图 2
图 1-2 LINUX 端 SPI/I2C 控制器部署示意图 4
图 1-3 Huawei LiteOS 端 SPI/I2C 控制器部署示意图 5
图 1-4 增加 SPI0 声明示意图 6
图 1-5 增加 SPI0 定义示意图 6
图 1-6 Huawei LiteOS 端增加 SPI0 示意图 7
图 1-7 Huawei LiteOS 端增加 I2C0 示意图 7
图 1-8 海思芯片对接 RGB LCD 屏幕基本框架图 8
图 1-9 RGB 屏配置时序图 9
图 1-10 屏的初始化序列示意图 10
图 1-11 某 RGB 屏时序配置示意图 12
图 1-12 VO User 时序下的 LCD 像素区域示意图 12
图 1-13 时钟源时钟频率与接口时钟之间的关系 15
图 1-14 LCDCLK 复用关系示意图 16
图 1-15 GPIO 口基地址示意图 17
图 1-16 GPIO 方向控制寄存器示意图 17
图 1-17 部分 VO 的 proc 信息 19
图 1-18 VO 设备中断信息 19
图 1-19 VDP 水平 colorbar 样式 20
图 1-20 VDP 垂直 colorbar 样式 20
图 2-1 MIPI DSI 接口连线示意图 2
图 2-2 对接 MIPI 屏基本框架图 3
图 2-3 MIPI 屏配置流程图 3
图 2-4 MIPI DSI 协议下 MIPI 像素区域示意图 6
图 2-5 MIPI DSI 协议下 MIPI 像素区域示意图 15

图 2-6 VO User 时序下的 LCD 像素区域示意图 16
图 2-7 时钟源时钟频率与接口时钟之间的关系 19
图 2-8 LCDCLK 复用关系示意图 20
图 2-9 GPIO 口基地址示意图 21
图 2-10 GPIO 方向控制寄存器示意图 21
图 2-11 部分 VO 的 proc 信息 24
图 2-12 VO 设备中断信息 24
图 2-13 MIPI Tx 的 proc 信息 25
图 2-14 VDP 水平 colorbar 样式： 26
图 2-15 VDP 垂直 colorbar 样式： 26
图 2-16 proc 显示参数 27
图 2-17 寄存器配置信息 27
图 2-18 寄存器配置信息 27
图 2-19 MIPI 垂直样式 29
图 2-20 MIPI 水平样式 30

表目录

表 1-1 VO_INTF_TYPE_E 结构体赋值示意表 11
表 1-2 VO_INTF_SYNC_E 11
表 1-3 VO_SYNC_INFO_S 结构体赋值示意表 13
表 2-1 combo_dev_cfg_t 结构体赋值示意表 6
表 2-2 cmd_info_t 结构体赋值示意表 9
表 2-3 VO_INTF_TYPE_E 结构体赋值示意表 14
表 2-4 VO_INTF_SYNC_E 14
表 2-5 VO_SYNC_INFO_S 结构体赋值示意表 16

1 RGB LCD 屏幕对接指南

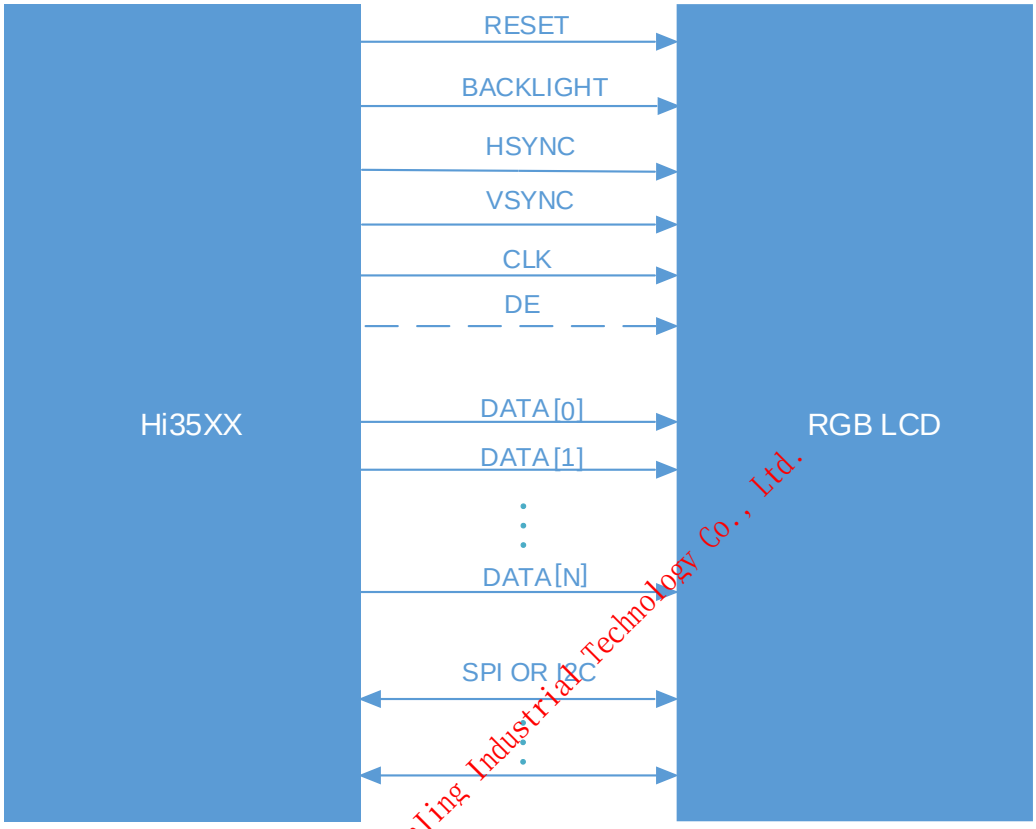
1.1 环境准备

1.1.1 RGB 屏幕接口介绍

板端与屏幕一般需要四种连线，各执行不同的功能，如图 1-1 所示。

- 控制命令传输接口(对应图中 SPI OR I2C)
- 图像数据传输接口（对应图中 HSYNC VSYNC CLK DATA0~DATAN DE）
- 背光控制（对应图中 BACKLIGHT）
- 硬件复位接口（对应图中 RESET）

图1-1 RGB 屏幕接口示意图



1.1.1.1 控制命令传输接口

控制命令接口可以用来更改或者获取屏幕的寄存器配置。

阅读 RGB LCD&MIPI LCD 屏的说明书，可以知道控制命令传输的接口类型，一般为 SPI 或者 I2C，也有一些屏幕不需配置，可能没有控制命令传输接口。

1.1.1.2 数据传输接口

板端利用该接口向 LCD 屏发送图像数据，俗称 RGB 接口，接口有 HSYNC VSYNC CLK DATA0~DATAN 等信号构成，而 DE 信号是否需要则取决于屏幕。一般 RGB 接口又被根据 RGB 信号是否经由 DATA 线分时复用发送，分为串行 RGB 和并行 RGB，串行 RGB 即 R、G、B 分量在不同时刻传输，并行 RGB 的 R、G、B 颜色分量在同一时刻传输。

时钟线 HSYNC、VSYNC、CLK 用来保证 RGB 数据按正确时序由主芯片端向 LCD 传输。

注意

有关 DE 模式说明：

- DE 模式是通过多接一个 Hi35xx 输出到 RGB LCD 上的信号线来准确通知 RGB LCD 屏行有效区的起始和结束点，即图 1-2 中的 DE 线。
- 因此当选择 RGB LCD 屏 DE 模式时，并确保硬件的 DE 管脚正确连接到 RGB LCD 屏上，水平消隐区对屏来说是不重要的。
- SDK 包默认配置为 DE 模式，故当 RGB LCD 屏选择 DE 模式，Hi35xx 这边无需做相关配置；若 RGB LCD 屏选择非 DE 模式，也无需再 Hi35xx 做开关 DE 模式配置，因为 RGB LCD 屏已忽略 DE 信号。

复位线 RESET，用来进行屏幕复位。

数据线，用来传输 RGB 数据。数据位传输有 6 位、8 位、16 位、18 位、24 位等。

1.1.1.3 背光调整接口

用于调整背光亮度的接口。

一般是通过 PWM 动态调整输出电流，继而控制屏幕背光亮度；关于 PWM 的设置可以参考《Hi35xx xx Camera SoC 用户指南》中的 PWM 配置章节，或者使用电阻配死无法动态调整，只能开关背光。

1.1.2 硬件连线确认

- 确认硬件设计按照《Hi35xx 硬件设计用户指南》的要求，且连线无异常。
- 确认硬件设计按照《Hi35xx LCD 输出说明》的要求，配置正确的管脚，否则会导致错误管脚无数据输出，显示异常。

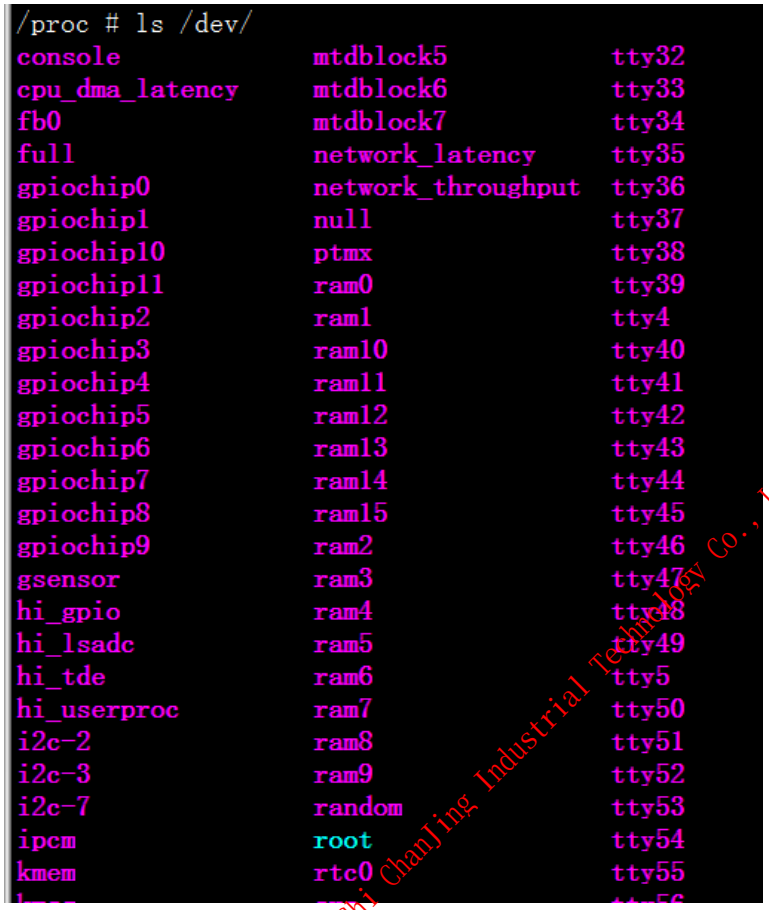
1.1.3 控制接口使能确认

对接 RGB LCD 屏幕时，需要确认控制线 I2C 或者 SPI 节点是否部署，时钟是否打开。

步骤 1 查询 I2C/SPI 控制器部署状态

- 查询 LINUX 端部署情况，可以在串口 ls /dev，如图 1-2 所示。

图1-2 LINUX 端 SPI/I2C 控制器部署示意图



以上图为例，在 LINUX 端部署了 I2C-2，I2C-3，I2C-7，并没有部署 SPI 设备，所以在 LINUX 端使用 SPI 设备或者没有部署的 I2C 设备就会有异常。

查询 Huawei LiteOS 端部署情况，可以在 Huawei LiteOS 端串口 ls /dev，如图 1-3 所示。

图1-3 Huawei LiteOS 端 SPI/I2C 控制器部署示意图

```
acodec          0
adec            0
aenc            0
ai              0
aio             0
ao              0
console1        0
gpio            0
hi_mipi         0
hi_mipi_tx      0
hi_rtc          0
i2c-0           0
i2c-1           0
i2c-4           0
ipcm            0
isp_dev         0
logmpp          0
mem             0
mmz_userdev     0
n3_dev          0
pm              0
pwm             0
rgn             0
serial          0
spidev0.0       0
spidev1.0       0
spidev2.0       0
spidev2.1       0
```


以上图为例，在 Huawei LiteOS 端部署了 I2C-0，I2C-1，I2C-4，部署了 SPI0 片选 0，SPI1 片选 0，SPI2 片选 0 和片选 1，所以在 Huawei LiteOS 端使用没有部署的 SPI/I2C 设备就会有异常。

注意

当两端同时部署和使用相同的 I2C/SPI 控制器的时候，就会发生不可预知的异常。所以建议两端分别部署 SPI/I2C 控制器。

步骤 2 改变 I2C/SPI 控制器部署状态

当 SPI/I2C 控制器部署关系，不符合业务需求。如屏幕驱动部署在 LINUX 或 Huawei LiteOS 端，需要 SPI2，但是 LINUX 或 Huawei LiteOS 端并没有部署 SPI2 的时候，就需要更改控制器的部署关系。

- 以在 LINUX 端增加 SPI0 的部署为例：
  - 首先，打开 osdrv/opensource/kernel/linux-x.x.x/arch/arm/boot/drs/hi35xx.dtsi 文件，在 aliases 结构体中，增加 spi0=&spi_bus0;

图1-4 增加 SPI0 声明示意图

```
#include <dt-bindings/clock/hi3556v200-clock.h>
/ {
    aliases {
        serial0 = &uart0;
        i2c3 = &i2c_bus3;
        i2c7 = &i2c_bus7;
        i2c2 = &i2c_bus2;
#ifndef CONFIG_ARCH_HISI_BVT_AMP
        i2c0 = &i2c_bus0;
        i2c1 = &i2c_bus1;
#endif
        i2c5 = &i2c_bus5;
        i2c6 = &i2c_bus6;
#ifndef CONFIG_ARCH_HISI_BVT_AMP

        spi1 = &spi_bus1;
        spi2 = &spi_bus2;
#endif
        spi0 = &spi_bus0;
        gpio0 = &gpio_chip0;
        gpio1 = &gpio_chip1;
        gpio2 = &gpio_chip2;
        gpio3 = &gpio_chip3;
        gpio4 = &gpio_chip4;
        gpio5 = &gpio_chip5;
        gpio6 = &gpio_chip6;
        gpio7 = &gpio_chip7;
        gpio8 = &gpio_chip8;
        gpio9 = &gpio_chip9;
        gpio10 = &gpio_chip10;
        gpio11 = &gpio_chip11;
    };
```

- 其次，在本文件下方 spi_bus0 的定义中，将状态从 disabled 改成 okey。

图1-5 增加 SPI0 定义示意图

```
        spi_bus0: spi@120c0000 {
            compatible = "arm,pl022", "arm,primecell";
            arm,primecell-periphid = <0x00800022>;
            reg = <0x120c0000 0x1000>;
            interrupts = <0 68 4>;
            clocks = <&clock HI3556V200_SPI0_CLK>;
            clock-names = "apb_pclk";
            #address-cells = <1>;
            #size-cells = <0>;
#ifdef CONFIG_HIEDMACV310
            dmas = <&hiedmacv310_0 27 27>, <&hiedmacv310_0 26 26>;
            dma-names = "tx","rx";
#endif
            status = "okey";
        };
```

- 最后重新编译和烧写内核。

- 以在 Huawei LiteOS 端增加 SPI0 和 I2C0 的部署为例：

  增加 SPI0，在 osdrv/platform/liteos/platform/bsp/borad/hi35xx/include/hisoc/目录下，打开 spi.h 文件，在#define SPI0_ENABLE 后置 1，如图 1-6 所示。

图1-6 Huawei LiteOS 端增加 SPI0 示意图

```
#define SPI0_ENABLE 1
#define SPI1_ENABLE 1
#define SPI2_ENABLE 1
#define SPI3_ENABLE 0
#define SPI4_ENABLE 0
```

  增加 I2C0，在 osdrv/platform/liteos/platform/bsp/borad/hi35xx/目录下，打开 borad.c 文件，在相应位置增加#define ENABLE_I2C0，如图 1-7 所示。

图1-7 Huawei LiteOS 端增加 I2C0 示意图

```
#ifdef LOSCFG_DRIVERS_I2C
#define I2C_NUM 5
#define ENABLE_I2C0
#define ENABLE_I2C1
#define ENABLE_I2C2
#define ENABLE_I2C3
#define ENABLE_I2C4
#define CLK_LIMIT_DEFAULT 400000
```

步骤 3 配置 I2C/SPI 管脚复用和时钟门控

- 首先，先配置 I2C/SPI 的管脚复用状态，将对应的 PIN 脚复用成 I2C/SPI，参考《Hi35XX_PINOUT_CN》进行修改。
- 其次，打开 I2C/SPI 的时钟，参考《Hi35xx xx Camera SOC 用户指南》的“系统”时钟章节，找到对应的 I2C/SPI 时钟门控所在的寄存器和位，配置对应的使能。

----结束

1.2 配置步骤

在这一章中将对屏幕对接时在软件方面需要进行的配置进行说明。

图 1-8 介绍了海思芯片对接 RGB LCD 屏幕的基本框架，客户主要需要操作的部分为用户程序配置层的内容和屏幕驱动。

图1-8 海思芯片对接 RGB LCD 屏幕基本框架图

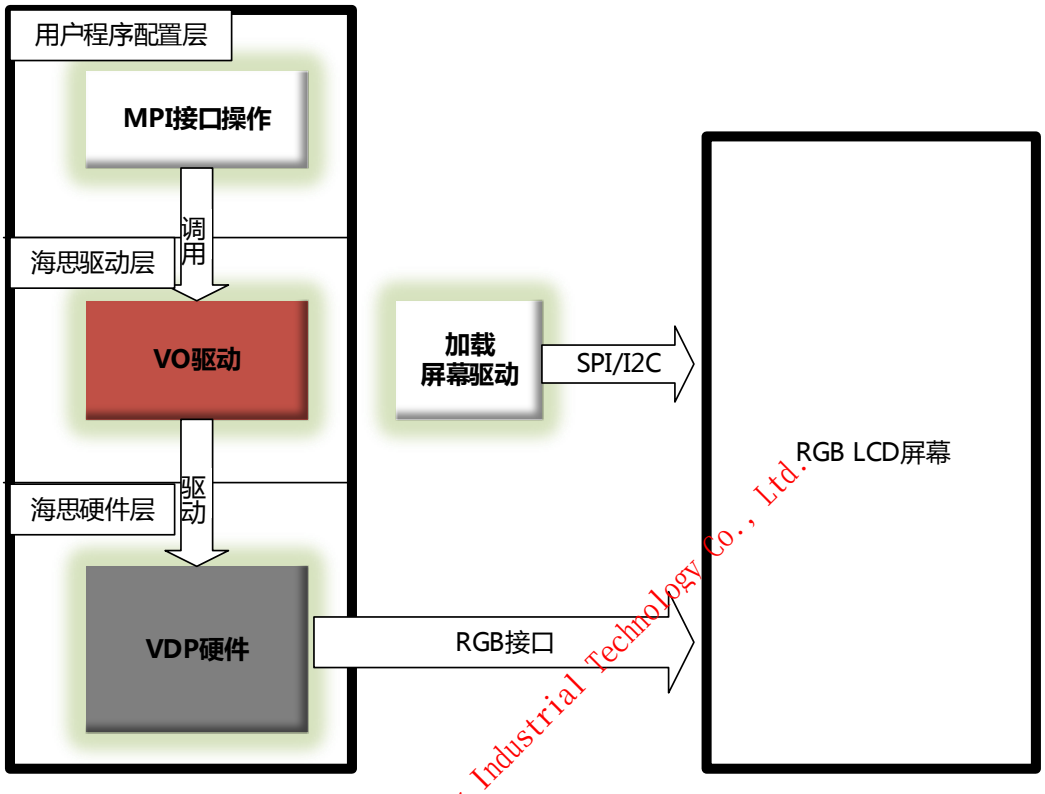
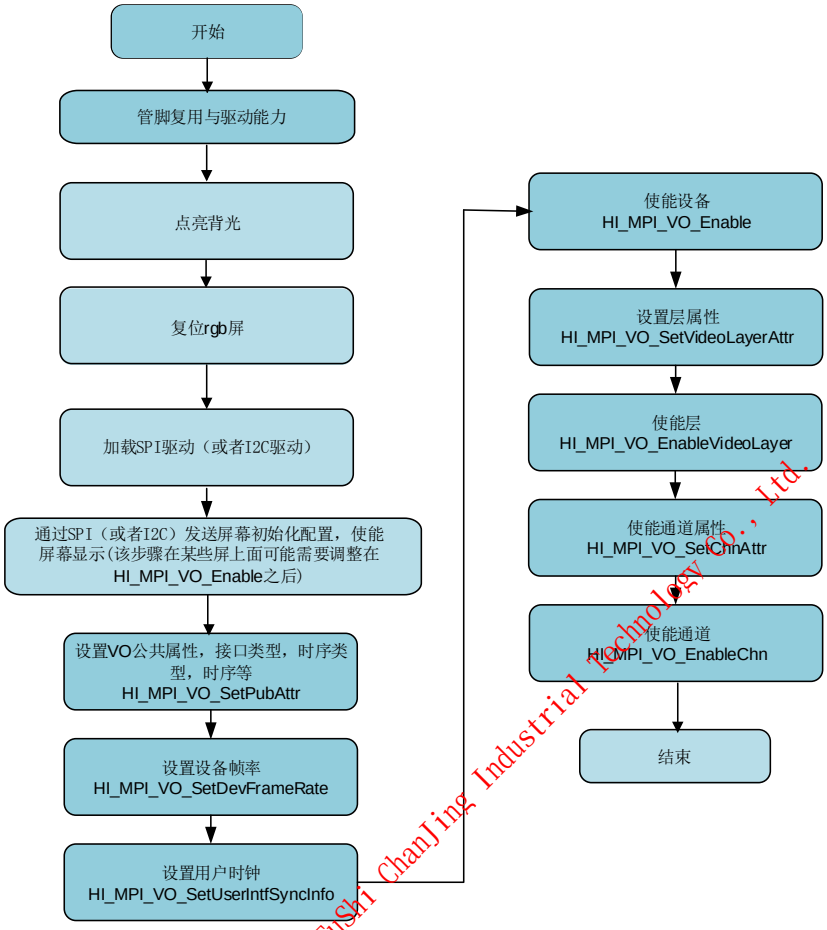


图 1-9 为 RGB 屏配置时序图，提供各操作流程进行的先后次序。

图1-9 RGB 屏配置时序图



1.2.1 配置管脚复用和驱动能力

对于 RGB 屏，可通过硬件原理图找到显示屏与主芯片的 I2C 或者 SPI 接口管脚，以及其数据交互管脚；然后通过查找主芯片的管脚信息文档《Hi35xx_PINOUT_CN》，使用配置寄存器的形式，配置管脚复用和驱动能力。

注意

管脚驱动能力配置取决于管脚对接外设，如果配的过大可能会导致时序波形有较大过冲，从而导致管脚受损，配的太小可能导致波形无法达到时序要求，因此，建议将驱动能力从小到大调试到符合其管脚对应时序要求。

1.2.2 配置 RGB LCD 屏复位

由于 RGB LCD 屏一般复位管脚用的是 GPIO 口。所以需要对 GPIO 口进行配置，同时进行屏的复位操作。

查询硬件连接图，获取复位的 GPIO 管脚。

配置复位所用的 GPIO 方向为输出（输出输入，是相对于主芯片来说的）。

屏的复位操作方法需要参考器件的说明书，若无复位操作，屏幕可能会无法点亮。

1.2.3 配置背光

一般是通过 PWM 动态调整输出电流，继而控制屏幕背光亮度；关于 PWM 的设置可以参考《Hi35xx xx Camera SoC 用户指南》中的 PWM 配置章节。

或者使用电阻固定电流、无法动态调整，只能开关背光。

1.2.4 配置 RGB LCD 屏初始化序列

屏幕一般会有初始化的过程，即 RGB LCD 屏一般通过 I2C 或者 SPI 发送数据或者命令来进行屏幕初始化，初始化的过程即向屏幕发送配置的过程。

初始化序列由屏厂提供，图 1-10 为某款 RGB LCD 屏厂提供的初始化序列的一部分，可以得到以下信息。

这是一款通过 SPI 通信的屏幕；

SPI 传输时，常见的屏幕单次传输 8bit、9bit 或者 16bit 之分，需要注意 spi 配置。

其它屏的初始化序列形式一般大同小异，区别为数据传输方式以及传输值。

图1-10 屏的初始化序列示意图

```
SPI_WriteComm(0x11);

Delay(120);              //Delay 120ms

SPI_WriteComm(0x36);
SPI_WriteData(0x00);

SPI_WriteComm(0x3a);
SPI_WriteData(0x66);
```


屏的初始化序列一般包括像素格式、数据刷新方向、Gamma 配置等，初始化序列中每一个指令具体含义，请在屏幕驱动 IC 的说明书中查找。

1.2.5 配置 VO 输出序列

1.2.5.1 User 时序配置

当对接一款屏幕时，请使用 User 时序接口的方式。相关接口的使用方法在《HiMPP V4.0 媒体处理软件开发参考》中“视频输出”章节中可以查到。配置用户时序接口参数类型 VO_PUB_ATTR_S，结构体如下所示。

```
typedef struct hiVO_PUB_ATTR_S
{
    HI_U32                u32BgColor;
    VO_INTF_TYPE_E         enIntfType;
```



```
    VO_INTF_SYNC_E          enIntfSync;
    VO_SYNC_INFO_S          stSyncInfo;
} VO_PUB_ATTR_S;
```

- 确认用户时序接口类型

  enIntfType 的值由屏本身决定。一般所配项如表 1-1 所示。

表1-1 VO_INTF_TYPE_E 结构体赋值示意表

成员名称	描述
VO_INTF_LCD_6BIT	RGB666 或者 RGB565 串行
VO_INTF_LCD_8BIT	RGB888 串行
VO_INTF_LCD_16BIT	RGB565 并行
VO_INTF_LCD_18BIT	RGB666 并行
VO_INTF_LCD_24BIT	RGB888 并行

- 确认用户时序类型

  enIntfSync 的值表示时序类型，选择 VO_OUTPUT_USER。

表1-2 VO_INTF_SYNC_E

成员名称	描述
VO_OUTPUT_USER	用户时序类型

- 填充用户时序信息

  一般来说，如图 1-11 所示，某典型 RGB 屏的 SPEC 上有相关的信息，而对于 VO 来说，User 时序对应的 LCD 像素区域示意图则如图 1-12 所示，用户时序结构体的数据类型为 VO_SYNC_INFO_S。


```
typedef struct hiVO_SYNC_INFO_S {
    HI_BOOL bSynm;
    HI_BOOL bIop;
    HI_U8 u8Intfb;

    HI_U16 u16Vact;
    HI_U16 u16Vbb;
    HI_U16 u16Vfb;

    HI_U16 u16Hact;
    HI_U16 u16Hbb;
    HI_U16 u16Hfb;
```

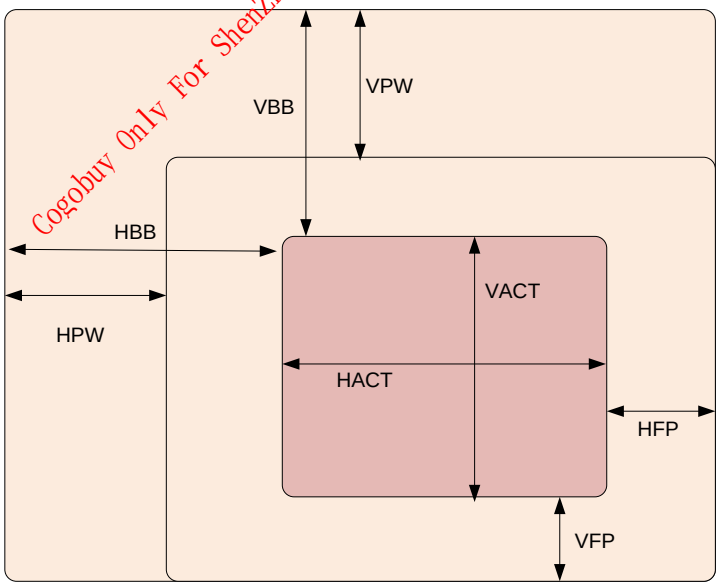


```
    HI_U16 u16Hmid;
    HI_U16 u16Bvact;
    HI_U16 u16Bvbb;
    HI_U16 u16Bvfb;

    HI_U16 u16Hpw;
    HI_U16 u16Vpw;

    HI_BOOL bIdv;
    HI_BOOL bIhs;
    HI_BOOL bIvs;
} VO_SYNC_INFO_S;
```

图1-11 某 RGB 屏时序配置示意图

Parameter	Symbol	Min.	Typ.	Max.	Unit
Horizontal Sync. Width	hpw	2	10	hpw+hbp=31	Clock
Horizontal Sync. Back Porch	hbp	4	10		Clock
Horizontal Sync.Front Proch	hfp	2	38		Clock
Vertical Sync. Width	vsa	1	4	vsa+vbp=127	Line
Vertical Sync. Back Porch	vbp	1	4		Line
Vertical Sync.Front Proch	vfp	1	8		Line

图1-12 VO User 时序下的 LCD 像素区域示意图



- 图 1-11 中 hpw+hbp 对应图 1-12 中的 HBB

- 图 1-11 中 hfp 对应图 1-12 中的 HFP
- 图 1-11 中 vsa+vbp 对应图 1-12 中的 VBB
- 图 1-11 中 vfp 对应图 1-12 中的 VFP

表1-3 VO_SYNC_INFO_S 结构体赋值示意表

成员名称	描述
bSynm	同步模式。参数无意义配 0。
bIop	0 为隔行时序，1 为逐行时序。MIPI 和 LCD 配置为 1。
u8Intfb	输出接口位宽。参数无意义配 0。
u16Vact	垂直有效区，单位为行。根据屏幕手册可知。
u16Vbb	垂直消隐后肩，单位为行。根据屏幕手册可知(VSA+VBP)。
u16Vfb	垂直消隐前肩，单位为行。根据屏幕手册可知。
u16Hact	水平有效区，单位为像素。根据屏幕手册可知。
u16Hbb	水平消隐后肩，单位为像素。根据屏幕手册可知（HSA+HBP）
u16Hfb	水平消隐前肩，单位为像素。根据屏幕手册可知。
u16Hmid	底场垂直同步有效像素值。逐行时序时无意义，置为 1。
u16Bvact	底场垂直有效区，逐行时有效，单位为行。逐行时序时无意义，置为 1。
u16Bvbb	底场垂直消隐后肩，逐行时有效，单位为行。逐行时序时无意义，置为 1。
u16Bvfb	底场垂直消隐前肩，逐行时有效，单位为行。逐行时序时无意义，置为 1。
u16Hpw	水平同步信号的宽度，单位为像素。
u16Vpw	垂直同步信号的宽度，单位为行。
bIdv	数据有效信号的极性。0 为高有效，1 为低有效。
bIhs	水平有效信号的极性，0 为高有效，1 为低有效。
bIvs	垂直有效信号的极性，0 为高有效，1 为低有效。

1.2.6 配置 VO 输出时钟

当配好 User 时序后，用户还需要配置 VO 的时钟频率。VO 接口输出的时钟频率即为用户要求设置的时钟频率。

具体频率值可以时序和帧率信息根据公式计算通过《RGB_MIPI 屏幕时钟时序计算器》表格自动生成。VO 接口输出时钟频率为时钟源输出时钟除以分频后得到。因此，为了得到接口输出时钟，需要先确定分频比，然后确认时钟源时钟频率。之后再调用 HI_MPI_VO_SetUserIntfSyncInfo 函数配置 VO_USER_INTFSYNC_INFO_S 结构体配置时钟。相关接口的使用方法在《HiMPP V4.0 媒体处理软件开发参考》中“视频输出”章节中可以查到。

说明

《RGB_MIPI 屏幕时钟时序计算器》表格暂不支持 Hi3516EV200/Hi3516EV300/Hi3518EV300/Hi3516DV200。

用户时序信息

```
typedef struct hiVO_USER_INTFSYNC_INFO_S {
    VO_USER_INTFSYNC_ATTR_S stUserIntfSyncAttr;
    HI_U32                  u32PreDiv;
    HI_U32                  u32DevDiv;
    HI_BOOL                 bClkReverse;
} VO_USER_INTFSYNC_INFO_S;
```

时钟源属性

```
typedef struct hiVO_USER_INTFSYNC_ATTR_S {
    VO_CLK_SOURCE_E enClkSource;
    union {
        VO_USER_INTFSYNC_PLL_S stUserSyncPll;
        HI_U32 u32LcdMClkDiv;
    };
} VO_USER_INTFSYNC_ATTR_S;
```

时钟源选择

```
typedef enum hiVO_CLK_SOURCE_E {
    VO_CLK_SOURCE_PLL,
    VO_CLK_SOURCE_LCDMCLK,
    VO_CLK_SOURCE_BUTT
} VO_CLK_SOURCE_E;
```

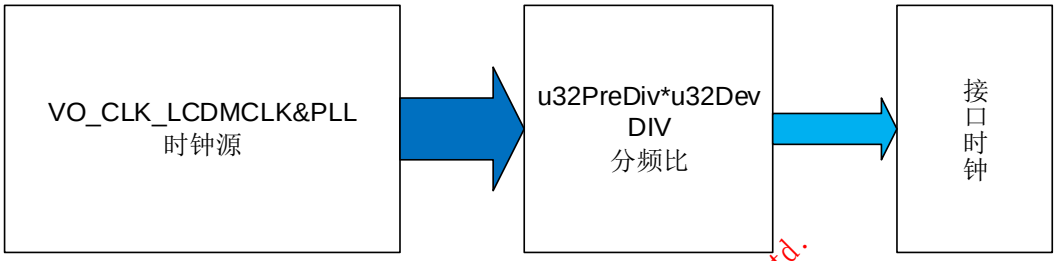
PLL 参数配置

```
typedef struct hiVO_USER_INTFSYNC_PLL_S {
    HI_U32 u32Fbdiv;
    HI_U32 u32Frac;
    HI_U32 u32Refdiv;
    HI_U32 u32Postdiv1;
    HI_U32 u32Postdiv2;
} VO_USER_INTFSYNC_PLL_S;
```


1.2.6.1 配置分频比

“分频比”的值为“前置分频”（u32PreDiv）与“设备时钟分频”（u32DevDiv）的积。“前置分频”与“时钟分频”的概念可以通过查询《HiMPP V4.0 媒体处理软件开发参考》中“视频输出”章节中可以查到。

图1-13 时钟源时钟频率与接口时钟之间的关系



各个芯片的 VO CRG 配置中，都有一个分频比的配置。

LCD 屏中，则可能需要多个时钟节拍来构造一个像素。

- 如果是 RGB 串行输出，则需要看 LCD 屏的说明书
  - 如果 3 个时钟节拍输出一个像素（R+G+B），则配置为“3 分频”；
  - 若是 4 个时钟节拍输出一个像素（R+G+B+无效数据），则配置为“4 分频”。
- 如果是 RGB 并行输出，则一般选择“不分频”。

1.2.6.2 时钟反向

时钟是否反向由对接的屏幕决定，调试时可以结合示波器观察 clk，并与 RGB LCD 屏幕说明书比对。

1.2.6.3 配置分频时钟

关于时钟计算和配置方法，可以参考《HiMPP V4.0 媒体处理软件开发参考》中 VO_USER_INTFSYNC_INFO_S 中的介绍将上述值配到该结构体中。

查看芯片手册找到 pll 选择与计算方法。也可通过 CRG 寄存器表格找到 VDP 时钟控制寄存器，用寄存器查看方法查看 VDP 选择了哪种时钟源。

说明
以下配置分频时钟方法不适用 Hi3516EV200/Hi3516EV300/Hi3518EV300/Hi3516DV200，具体配置方法请参考相关芯片手册。

对 Hi3516CV500 芯片，如：himd.l 0x12010108

VDP 的 LCD 分频时钟或者 PLL 时钟频率计算方法：

以一款 320*240 帧率为 60 的屏为例，若此款屏的说明说中给出的时序信息如下：

HACT	HBP	HFB	HSA	VACT	VBP	VFB	VSA
320	64	7	1	240	14	9	1

根据表 1-3 中的对应关系可得：

u16Hact：320；u16Hbb：65；u16Hfb：7；u16Vact：240；u16Vbb：15；u16Vfb：9。

首先计算时钟频率，单位 MHz。

X = （u16Hact + u16Hbb + u16Hfb）* (u16Vact + u16Vbb + u16Vfb) * framerate * u32PreDiv*u32DevDiv / 10^6。

按上述屏来计算的话，X =（320 + 65 + 7）*（240 + 14 + 9） * 60* 1 * 1 /10^6 = 6.209

LCD 分频系数 u32LcdMClkDiv =（X / 1188）* (2^27), 此时 X 最大为 75MHz。

因此，u32LcdMClkDiv = （6.209 / 1188）* (2^27) = 0xAB448。

PLL 时钟频点的 u32Fbdiv，u32Frac、u32Refdiv、u32Postdiv1 和 u32Postdiv2 参数。

公式：X = 24 *（u32Fbdiv + u32Frac/（2^24））/ u32Refdiv/ u32Postdiv1/ u32Postdiv2。

已上述屏为例，u32Fbdiv，u32Frac、u32Refdiv、u32Postdiv1 和 u32Postdiv2 分别为 1、0x35935F、2、2、1。

1.3 屏幕验证与调试

1.3.1 管脚复用与驱动能力确认

步骤 1 确认控制接口 SPI/I2C 的接口管脚复用是否配置正常。

步骤 2 确认屏幕的数据接口(LCD_DATA、HSYNC、VSYNC、CLK 等)的接口管脚复用是否配置正常。驱动能力在调试初期可以先配置较高档位，待调亮屏幕后，再调优，以免损坏器件。

可以参照《HI35XX_PINOUT_CN》和硬件设计图，通过 himd.l + 寄存器地址，来读出管脚复用状态。

如果状态不符合，请在代码的对应地方，修改不匹配的复用关系。

以 Hi3559V200 芯片的 LCD_CLK 引脚为例，在《Hi3559V200_PINOUT_CN》中找到 LCD_CLK 所在的寄存器地址，如图 1-14 所示：

图1-14 LCDCLK 复用关系示意图

0x112F_0034	0x0400	31:11	保留。
		10	电平转换速率控制，为0时电平转换速率快，为1时电平转换速率慢。
		9	下拉电阻使能，高有效。
		8	上拉电阻使能，高有效。
		7:4	驱动能力，0~7对应IO驱动能力IO4_档位1~档位8。
		3:0	功能选择： 0:GPIO0_6 2:LCD_CLK 3:VOU_CLK

当读寄存器 0x112F0034 的值 3:0 位不是 0x2，则代表 LCD 屏的 LCD_CLK 的引脚复用关系错误，需要在代码的相应地方修改。

1.3.2 复位操作流程确认

步骤 1 首先确认复位管脚能否正常置高置低，用万用表或者示波器观测均可。

查询硬件设计图，找到复位管脚对应的 GPIO 口。确认对应的管脚的复用关系为 GPIO。

参考芯片手册“外围”章节 GPIO 小节，将该 GPIO 的方向配成输出。

在 Hi3559V200 上，配置 GPIO5_2 的方向为输出为例：

找到 GPIO5 的基地址为 0x120D5000

图1-15 GPIO 口基地址示意图

GPIO 控制器	基地址
GPIO8	0x120D_8000
GPIO7	0x120D_7000
GPIO6	0x120D_6000
GPIO5	0x120D_5000
GPIO4	0x120D_4000
GPIO3	0x120D_3000
GPIO2	0x120D_2000
GPIO1	0x120D_1000
GPIO0	0x120D_0000

将基地址+偏移地址 0x400 的寄存器的第 2 位配成 1。

图1-16 GPIO 方向控制寄存器示意图

GPIO_DIR

GPIO_DIR 为 GPIO 方向控制寄存器。用来配置 GPIO 管脚方向。

Offset Address: 400　　Total Reset Value: 0x00

Bits	Access	Name	Description	Reset
[7:0]	RW	gpio_dir	GPIO 方向控制寄存器。bit[7:0]分别对应 GPIO_DATA[7:0]，各比特可独立控制。 0：输入； 1：输出。	0x00

参考芯片手册“外围”章节 GPIO 小节，分别将该 GPIO 口拉高拉低，通过示波器测量引脚状态是否正确。

在 Hi3559V200 上，拉高拉低 GPIO5_2 为例，前提是已经配置了 GPIO5_2 为输出：

找到对应的 GPIO_DATA 寄存器，为 0x120D5000+0x1>>(2+2)，如果是 GPIO5_3，则是 0x120D5000+0x1>>(3+2)。

再将该寄存器的第 2 位分别配成 0 和 1，对应的是下拉和上拉。

步骤 2 再通过查询屏幕的说明书，确认复位操作是否和屏幕要求一致。如先置高，再置低，再置高，并确认各步骤间的延时要求。

----结束

1.3.3 背光控制确认

背光控制分两种：

- 一种可以动态调整背光亮度。

  动态调整背光亮度的 LCD 屏，多数情况是通过 PWM 调整占空比来实现亮度调整，需实现 PWM 驱动。
- 一种只能通过硬件配置亮度（即无法动态配置）。

  调试阶段可先将背光调至最亮。

1.3.4 控制命令通路确认

可以通过以下几种方法确认控制命令发送是否成功：

- 一般屏幕的控制命令发送接口为 SPI 或者 I2C,可以通过示波器观察命令发送时的波形，并结合对应协议，核对发送信息，并与屏幕说明书指定的信息发送时序比对。
- 可以通过读取一些屏幕的寄存器，看读取到的寄存器值是否符合预期，以达到测试调试通路是否正常的目的。可以先将一个寄存器设置成一个特定的值，再将这个寄存器读出，看读出的值是否符合预期。需要注意的是有些寄存器是只读的，或者只写的，并不是所有的寄存器都同时支持读写功能，具体需要看屏幕驱动 IC 的说明书手册。
- 通过发送一些特定的序列，开启屏幕的自测功能，比如屏幕的 colorbar 厂测程序等。

1.3.5 VO 输出时钟和时序确认

1.3.5.1 输出时钟

VO 的输出时钟，可以通过示波器测量 vo_clk 引脚的频率，通过示波器测量得到的频率应该与通过以下两种方法得的频率一致。

- 观察 proc 信息中的帧率，根据（u16Hact + u16Hbb + u16Hfb）* (u16Vact + u16Vbb + u16Vfb) * framerate=输出时钟频率，可以反推是否输出符合预期。在业务程序运行过程中，通过串口终端输入命令 cat /proc/umap/vo 的方法查看 VO 模块的 proc 信息。

  可以通过 VO 的 proc 信息查看到配置的时钟源和时钟源参数信息以及 VDP 逻辑的中断也就是设备帧率信息，即图 1-17 中的 IntRate，即实际的帧率数。

图1-17 部分 VO 的 proc 信息

```
~ # cat /proc/umap/vo

[VO] Version: [Hi3516CV500_MPP_V1.0.0.0 B010 Release], Build Time[Mar 26 2019, 17:26:57]

-----DEVICE CONFIG-----------------------------------------------------------------
  DevId   DevEn       Mux1       Mux2       Mux3      IntfSync       BkClr  DevFrt
     0       Y  LCD_24BIT                                   -        0xff      50

-----DEVICE CLOCK INFO-------------------------------------------------------------
  DevId   DevEn  ClkSource   FbDiv    Frac     RefDiv   PostDiv1  PostDiv2   LCDMCLK  VoDevDiv  VoPreDiv ClkReverse
     0       Y    LCDMCLK                                                   3787922         1         1          N
```

```
-----DEV Int Status---------------------------------------------------------------
   DevId IntRate IntTime MaxIntT TimePrM  IntGapT  MaxGapT
      0      59     203     216   11342    16668    16760
```

- 此外对于 linux 版本还可以在业务运行的过程中输入通过命令：watch –n 10 cat /proc/interrupts 查看每 10 秒 VO 产生的中断个数，通过计算后得到实际帧率。

图1-18 VO 设备中断信息

```
~ # watch -n 10 cat /proc/interrupts
Every 10s: cat /proc/interrupts

 49:       2636          0      GIC-0  90 Level     VO Int
```

```
 49:       3237          0      GIC-0  90 Level     VO Int
```

根据图 1-18 所示，可得（3237-2636）/10 = 60。

1.3.5.2 时序确认

通过示波器将观测 hsync 和 vsync 以及 clk，结合 RGB 时序协议，可以得到各个时序信息实际生效的值，即 hpw，hfp，hact 等，该过程一般只需关注时序有无波形变换，然后检查配置的 VO 时序参数即可。

1.3.6 VO Colorbar 调试

colorbar 是一种重要的调试手段，通过 colorbar 的显示，能够看到特定的色条，有利于分析时序信息异常，颜色异常。

如果 VDP 的配置上有问题时大多数都可以在 colorbar 上表现出来。如果 colorbar 上的颜色出现了偏色，则可能 VDP 的 CSC 设置有问题，如果 colorbar 的图像显示出现错位，则可能存在时钟时序的问题。如果 colorbar 的显示完全不成型，则 RGB LCD 屏的设置可能存在问题。

colorbar 的调试手段可以很好的帮助定位问题，定位时需要根据具体现象具体分析。

调试手段：

通过设置 VDP 寄存器的方式启动 VDP 设备 colorbar。根据芯片手册查找 VDP 寄存器 DHD0_CTRL 或者是 DHD1_CTRL（DHD0 还是 DHD1 取决于对接屏幕的是哪个设备）。在不改变该寄存器其他位值的状态下，配置 colorbar 使能位 cbar_en 使能；配置

colorbar 色彩空间选择位 cbar_sel 为 VGA；配置 colorbar 模式选择位 cbar_mode 水平或者垂直。配置 DHD0 寄存器更新位 regup 为 1。

以 Hi3516CV500 为例：

查看寄存器：himd.l 0x1144d000

打开 colorbar：himm 0x1144d000 0xc0000011

关闭 colorbar：himm 0x1144d000 0x80000011

其中 0x11440000 为 VDP 寄存器基址，0xd000 为 DHD0_CTRL 寄存器偏移地址

VDP colorbar 样式

图1-19 VDP 水平 colorbar 样式

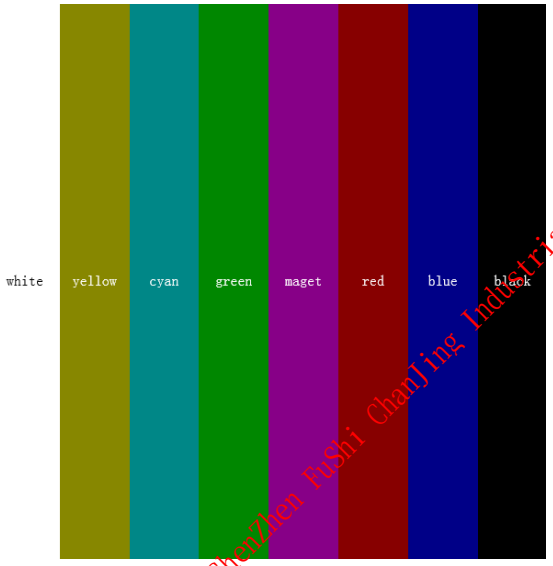
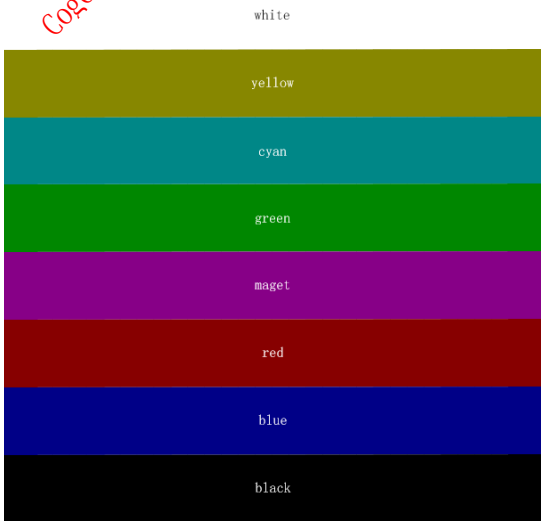


图1-20 VDP 垂直 colorbar 样式



1.4 显示屏调试 FAQ

1.4.1 屏幕全黑，毫无反应

【问题现象】

屏幕完全黑色，没有响应。

【问题分析】

有很多因素会导致屏幕无响应，需要逐个环节排查，例如器件损坏、连接异常、供电异常、管脚复用配置异常、reset 管脚状态异常、vo 时序配置参数错误等。

【解决方案】

步骤 1 硬件确认。

找硬件工程师确认屏幕供电、连接正常，且器件无损坏。

步骤 2 确认软件流程的正确性。

请比对 1.2 章节配置步骤中的 RGB 屏配置时序图，需要特别注意的是 reset 管脚操作有严格的时间要求，一般为 120ms 后才能接受控制命令。

步骤 3 确认管脚复用与驱动能力配置正确，请参考 1.2.1 配置管脚复用和驱动能力章节

步骤 4 确认控制命令传输通路是否正常，请参考 1.3.4 控制命令通路确认章节。

步骤 5 确认背光开启，确保背光打开，请参考 1.2.3 配置背光章节。

步骤 6 确认 vo 的输出时钟与输出序列是否正确

请参考 1.2.5 配置 VO 输出序列 和 1.3.5 VO 输出时钟和时序确认来检查 vo 输出配置是否正确。

----结束

1.4.2 屏幕显示位置错误

【问题现象】

图像只在屏幕上显示出一部分。

【问题分析】

这种情况的表象为实际显示的图像在屏幕上只显示一部分，显示图像起始点不是屏幕的初始点。

【解决方案】

一般是主芯片侧的行或者场的消隐区配置与屏幕的配置未匹配导致的，需要将两者保持一致，可以调整屏幕配置，也可以调整芯片侧配置。

1.4.3 屏幕显示裂屏

【问题现象】

显示图像起点位置移至屏幕中间。

【问题分析】

调整前后消隐区大小没有变化。可能是输出像素时钟与时序不匹配。

【解决方案】

通过 1.3.5 节“VO 输出时钟和时序确认”中的调试手段定位时钟是否正常，如果不对确定时钟有问题。

1.4.4 屏幕显示大小错误

【问题现象】

图像在屏幕上显示时出现实际值大小不匹配的情况。

【问题分析】

这种情况表象为实际显示图像的尺寸超过屏幕尺寸（即显示不全）或者小于屏幕尺寸。

【解决方案】

一般是主芯片侧的行或者场的有效区配置与屏幕的配置未匹配导致的，需要将两者保持一致，可以调整屏幕配置，也可以调整芯片侧配置。

1.4.5 屏幕显示颜色异常

这种情况表象为图像的颜色与预期发生明显的差别，有多种表象，举例如下。

画面完全混乱显示效果无法辨认出目标的轮廓

【问题现象】

图像画面完全混乱，看不出任何的图像效果。

【问题分析】

这种情况表现多为接口类型选择有误，比如主芯片侧选用了 BT656 接口发送，而屏幕端选择了 RGB 类型。

【解决方案】

请核对设置的 VDP 接口类型与所选择的接口是否一致。

出现颜色发生变化

【问题现象】

RGB LCD 显示图像出现红、蓝、绿出现错乱的情况。

【问题分析】

这种情况可能为 RGB 发送顺序未匹配。

【解决方案】

对于串行 RGB 来说，可以尝试以下几种解决方式：

- 请确认屏幕发送 RGB 的顺序和屏幕接受 RGB 顺序一致，有些屏幕支持 RGB 接收顺序的调整，这种情况的话，更改屏幕寄存器配置即可。
- clk 输出的极性如果和屏幕预期的不一致，会导致 rgb 顺序错乱，可以尝试修改 HI_MPI_VO_SetUserIntfSyncInfo 接口中 VO_USER_INTFSYNC_INFO_S 结构体的 bClkReverse 成员。
- 海思的串行 RGB 输出有两种方式可选：三分频或者四分频。
  - 配置时序是以三个或四个像素时钟周期作为配置的步进。比如：同步宽度增大 1，实际同步信号增加了三个或四个像素的时钟周期；
  - 如果屏幕侧对消隐区或者同步信号时序配置并非三个或四个像素时钟周期的倍数，则可能出现 RGB 接收错位的情况，比如：RGB 被解析成为了 GBR，这时可以通过调整屏幕侧对消隐区或者同步信号时序配置。

对于并行 RGB 来说

- 先确认硬件的线序是否正确，是否出现 RGB 排列错误。
- 其次看当前的软件配置，是否能保证主芯片侧 RGB 的排序和屏幕接受 RGB 的排序一致。

假如出现了不一致的情况，可以考虑调整主芯片侧或者屏幕侧的排序（有些屏幕支持调整）。对于主芯片侧 RGB 的排序配置调整方法，请参阅《Hi35xx xx Camera SoC 用户指南》的 VDP 章节，查询相关配置寄存器。

显示颜色异常

【问题现象】

图像显示的颜色不正常，颜色转换方式不对。

【问题分析】

这种情况有可能是视频层的 CSC 没有配置正确，LCD 或者 MIPI 屏幕需要转换 RGB 的 CSC 输出才行。

【解决方案】

调用 HI_MPI_VO_SetVideoLayerCSC 或者 HI_MPI_VO_SetGraphicLayerCSC 函数设置 CSC 属性。设置方法可参考《HiMPP V4.0 媒体处理软件开发参考》中的“视频输出”章节中相关接口的操作说明。

显示线条有色差

【问题现象】

显示图像的轮廓线处有色差。

【问题分析】

这种情况的表象为线条颜色发生异常，特别是图像的轮廓线，受相邻像素的影响，这种情况一般为时序有细微错误，导致数据传输的时候，该像素的的信息和相邻像素的信息发生了传输错误，这种情况有多种可能。

【解决方案】

硬件干扰较大，请先用示波器看看是否是波形符合预期，如果不符合，可能是走线过长导致（一般飞线才可能出现这种情况），或者是管脚驱动能力配置不足。

视频输出的 clk 相位错误，未和屏幕匹配，需要将主芯片侧和屏幕侧调整到时序正确。

1.4.6 屏幕显示效果不够满意

【问题现象】

图像显示时在亮度、色度、对比度等方面达不到想要的效果。

【问题分析】

屏幕显示效果不够满意。

【解决方案】

调节屏幕显示效果，需要从两个角度出发：

- 调整主芯片测的图像输出效果。
- 调整屏幕侧的显示效果参数。

主芯片测的显示效果可以通过调整 vo 的参数，可以调整亮度对比度等，屏幕侧一般可以调整 gamma 曲线，亮度、色度等，一般屏幕厂商会给出推荐配置，能满足大部分场景需求。

1.4.7 屏幕画面转换时出现残影

【问题现象】

屏幕显示的图像在变换的过程中出现上一帧画面的残留。

【问题分析】

残影一般由 vcom 电压配置不合理或者一些供电管脚电压异常导致。

【解决方案】

请查看屏幕手册说明或者屏幕厂商获取解决办法。

1.4.8 显示帧率低/画面卡顿

【问题现象】

显示的画面出现卡顿的情况。

【问题分析】

一般这种情况是数据时钟频率不够。

【解决方案】

请调整时钟频率。

1.4.9 显示画面较暗，但背光正常

【问题现象】

图像显示画面较暗，但是背光确实正常的。

【问题分析】

可能 RGB 屏部分 DATA 线无数据，硬件设计的时候，没有参考 Hi35XX 配置正确的管脚，导致屏幕与主芯片间部分管脚无数据。

【解决方案】

出现画面较暗的原因较多，遇到这种问题时，可以先排除上述问题分析的可能性。

2 MIPI LCD 屏对接指引

说明
Hi3516EV200/Hi3516EV300/Hi3518EV300/Hi3516DV200 不支持 MIPI Tx。

2.1 环境准备

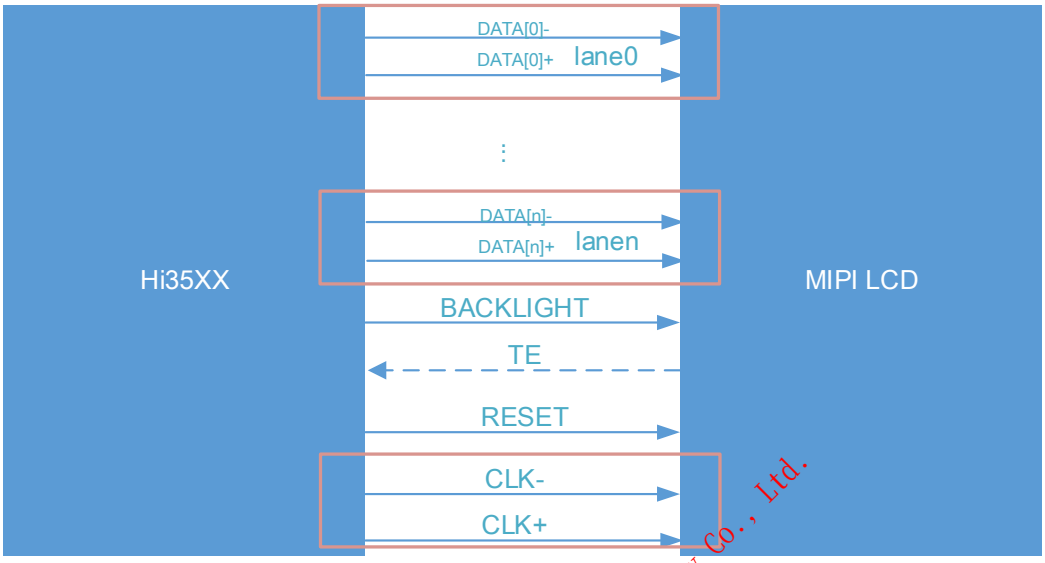
2.1.1 MIPI DSI 屏幕接口介绍

常用于智能手机领域，优点为：更低功耗、更高数据传输率以及更小的 PCB 占用空间。Hi35xx 系列芯片部分支持该类型接口。

MIPI DSI 屏幕一般有以下几种信号，如图 2-1 所示。

- mipi 时钟线（CLK）
- mipi 数据线（DATA），最大为 4Lane
- TE 线是一根信号同步线，一般 command mode 的屏幕才会有这个引脚，用于帧同步用。用于海思平台 MIPI Tx 的 SLAVE 模式。
- 背光控制信号（BACKLIGHT）
- 复位引脚 (RESET)

图2-1 MIPI DSI 接口连线示意图



关于 MIPI Tx 接口更为细致的介绍以及 SDK 配置方法，可以参考《Hi35xx xx Camera SoC 用户指南》的视频接口章节中的 MIPI Tx 小节，以及《MIPI 使用指南》。

2.1.2 硬件连线确认

- 确认硬件设计按照《Hi35xx 硬件设计用户指南》的要求，且连线无异常。
- 确认硬件设计按照《Hi35xx LCD 输出说明》的要求，配置正确的管脚，否则会导致错误管脚无数据输出，显示异常。

2.2 配置步骤

根据上节环境准备的内容，在接口和连线上了解了屏幕对接的配置，在这一章中将对屏幕对接时在软件方面需要进行的配置进行说明。

海思芯片对接 MIPI LCD 屏幕基本框架图。

图2-2 对接 MIPI 屏基本框架图

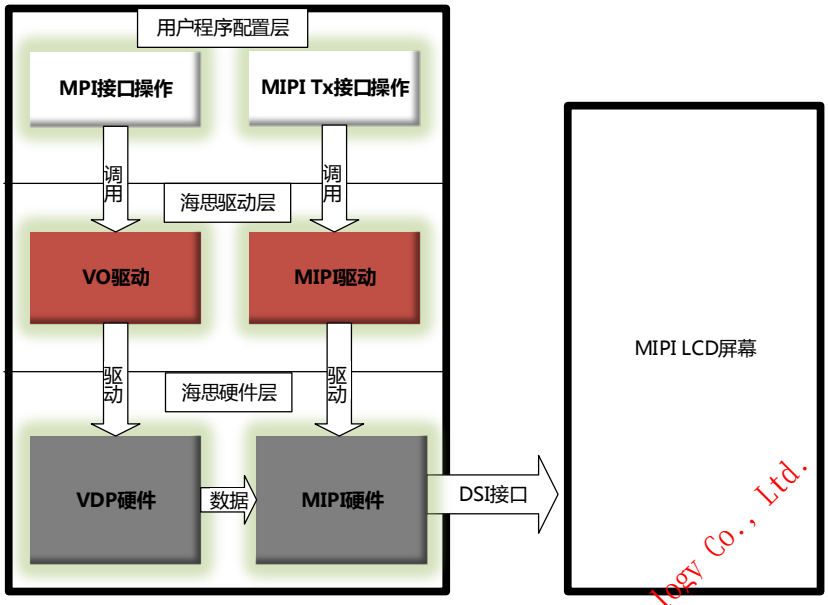
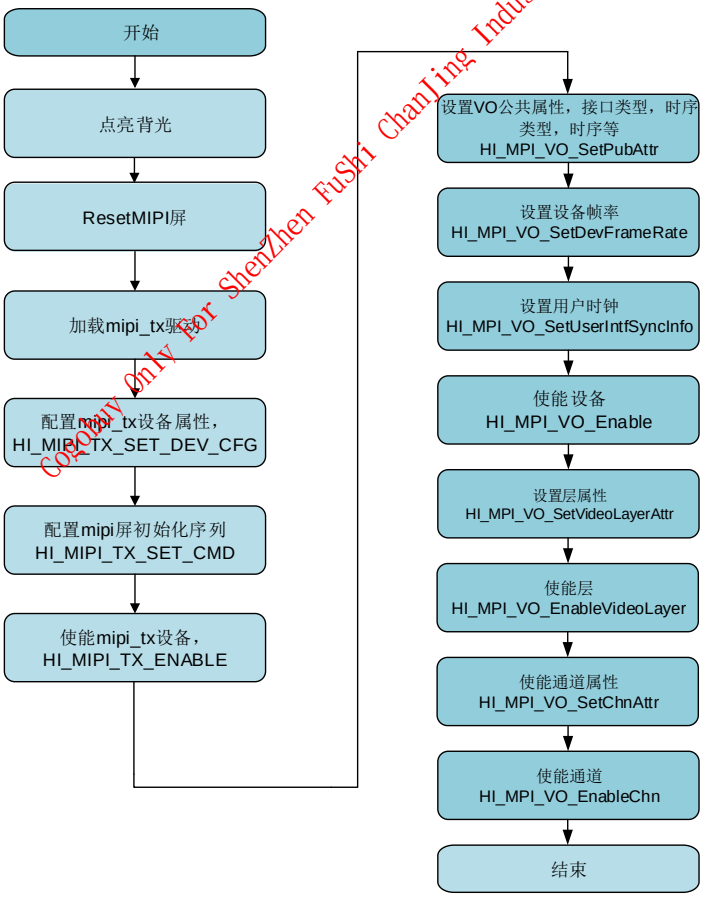


图2-3 MIPI 屏配置流程图



2.2.1 配置管脚复用和驱动能力

阅读板端硬件设计图，并参考 2.1.1 MIPI DSI 屏幕接口介绍中说明，找到显示屏驱动 IC 与主芯片交互的控制管脚、数据交互管脚等，并记录对应管脚号。

在主芯片的管脚信息文档中《Hi35xx_PINOUT_CN》，查询上述管脚的配置寄存器，并记录对应寄存器，用于后续的配置管脚复用。

注意

- 管脚驱动能力配置取决于管脚对接外设，如果配的过大可能会导致时序波形有较大过冲，从而导致管脚受损，配的太小可能导致波形无法达到时序要求，因此，建议将驱动能力从小到大调试到符合其管脚对应时序要求。
- 对于 Hi3519AV100\Hi3556AV100 芯片，如需使用 MIPI DSI 功能，需要确保 MISC_CTRL1 寄存器的 mipitxphy_cmos_mode_enable 比特位配置为 0。关于此寄存器的详细描述请参考《Hi35xx xx Camera SoC 用户指南》。

2.2.2 配置 MIPI 屏复位

由于 MIPI 屏一般复位管脚用的是 GPIO 口，所以需要对 GPIO 口进行配置，同时进行屏的复位操作。

- 查询硬件原理图，获取复位的 GPIO 管脚。
- 配置复位所用的 GPIO 方向为输出。
- 配置复位所用的 GPIO 的复位操作时序。

屏的复位操作需要参考屏幕的说明书，若无复位操作或者复位的时序与屏幕要求的不匹配，或者电平不匹配，屏幕可能会无法点亮或者工作异常。

2.2.3 配置背光

一般是通过 PWM 动态调整输出电流，继而控制屏幕背光亮度；关于 PWM 的设置可以参考《Hi35xx xx Camera SoC 用户指南》中的 PWM 配置章节。

或者使用电阻固定电流、无法动态调整，只能开关背光。

2.2.4 配置 MIPI 屏幕

对于 MIPI 屏的配置主要包括两个部分：

- MIPI 设备属性的配置。
- MIPI 屏幕初始化序列的配置。

初始化 MIPI 屏步骤如下所示。

步骤 1 配置 MIPI Tx 设备属性

SDK 提供了结构体 combo_dev_cfg_t，将该结构体中每一项都赋值，再通过 HI_MIPI_TX_SET_DEV_CFG 配置下去。

combo_dev_cfg_t 结构体定义

```
typedef struct
{
    unsigned int       devno;
    short             lane_id[LANE_MAX_NUM];
    output_mode_t      output_mode;
    video_mode_t       video_mode;
    output_format_t     output_format;
    sync_info_t         sync_info;
    unsigned int       phy_data_rate;
    unsigned int       pixel_clk;
} combo_dev_cfg_t
```

sync_info_t 结构体定义

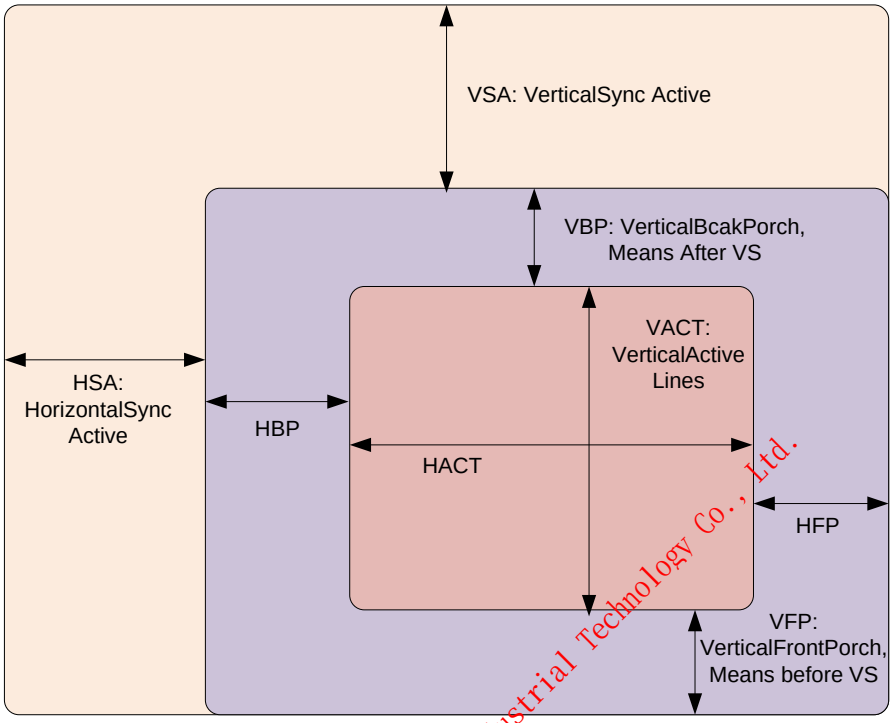
```
typedef struct
{
    unsigned short  vid_pkt_size;
    unsigned short  vid_hsa_pixels;
    unsigned short  vid_hbp_pixels;
    unsigned short  vid_hline_pixels;
    unsigned short  vid_vsa_lines;
    unsigned short  vid_vbp_lines;
    unsigned short  vid_vfp_lines;
    unsigned short  vid_active_lines;
    unsigned short  edpi_cmd_size;
} sync_info_t;
```


combo_dev_cfg_t 中 sync_info_t（MIPI Tx 设备的同步信息）比较难配置，下面详细介绍它的配置方法，如图 2-4 所示。

- 粉红色区域为 MIPI 屏显示的有效区域，两条直线即 MIPI 屏的宽高；
- 淡黄色区域的两条直线为行同步脉冲区像素个数以及帧同步脉冲区行数；
- 紫色区域的两条直线为行同步前后消隐区像素个数以及帧同步前后消隐区行数。

对于这些参数，需要咨询屏幕厂商或者查询屏幕手册。

图2-4 MIPI DSI 协议下 MIPI 像素区域示意图



根据屏幕的说明书来填充 combo_dev_cfg_t 的值，如表 2-1 所示。

表2-1 combo_dev_cfg_t 结构体赋值示意表

成员名称	描述
devno	MIPI Tx 设备号，配 0。
lane_id	4lane：配成{0,1,2,3} 3lane：配成{0,1,2,-1} 2lane：配成{0,1,-1,-1} 1lane：配成{0,-1,-1,-1} 未使用的 lane 设置为-1。
output_mode_t	MIPI Tx 输出模式。 参数选择范围： OUTPUT_MODE_CSI、OUTPUT_MODE_DSI_VIDEO、OUTPUT_MODE_DSI_CMD。 三种参数分别表示了 MIPI Tx 不同的数据输出模式。具体选择哪种输出模式，参考屏幕说明书。

成员名称	描述
video_mode_t	VIDEO 模式下的数据格式。 参数选择范围：BURST_MODE、NON_BURST_MODE_SYNC_PULSES、NON_BURST_MODE_SYNC_EVENTS。 三种模式下，传递的数序和数据包位置不同，具体根据特定屏要求赋值。参考屏幕说明书。 如果 output_mode_t 属性为 OUTPUT_MODE_DSI_CMD，本属性配置不生效。
output_format_t	MIPI Tx 输出数据类型。 参数选择范围：OUT_FORMAT_RGB_16_BIT、OUT_FORMAT_RGB_18_BIT、OUT_FORMAT_RGB_24_BIT、OUT_FORMAT_YUV420_8_BIT_NORMAL、OUT_FORMAT_YUV420_8_BIT_LEGACY、OUT_FORMAT_YUV422_8_BIT。 参数表示的是 MIPI Tx 发送的数据包中数据的格式，根据特定屏要求赋值。参考屏幕说明书。
vid_pkt_size	接收包大小，数据上等于 hact，水平有效区长度，单位：像素。
vid_hsa_pixels	行同步脉冲区像素个数。根据屏幕要求配置，要求和 VO 时序中的水平同步信号宽度（u16Hpw）一致。
vid_hbp_pixels	行同步脉冲后消隐区像素个数。
vid_hline_pixels	每行像素总数，大小上等于 hact+hsa+hbp+hfp。
vid_vsa_lines	帧同步脉冲区行数。根据屏幕要求配置，要求和 VO 时序中的垂直同步信号宽度（u16Vpw）一致。
vid_vbp_lines	帧同步脉冲后消隐区行数。
vid_vfp_lines	帧同步脉冲前消隐区行数。
vid_active_lines	数据上等于 vact，垂直有效区长度，单位：行数。
epdi_cmd_size	写内存命令字节数。VIDEO Mode 时该值无效，CMD Mode 时该值设为 hact。
phy_data_rate	数据输入速率，单位为 Mbps。配置值>= (hact+hsa+hbp+hfp)*(vact+vsa+vbp+vfp)*output format bits * framerate / lane_num/（10^6）。
pixel_clk	像素时钟，单位为 KHz。配置值 = (hact+hsa+hbp+hfp)*(vact+vsa+vbp+vfp) * framerate/ 1000 向上取整

注意

- 对于某些时序的屏幕，使用表格中 phy_data_rate 的计算公式得到的值没有点亮屏幕，且这时读取 MIPI Tx 的寄存器 INT_ST1（中断状态 1 寄存器）信息，如果第 24bit 位值为 1。则说明 phy_data_rate 设置的值偏小，需要适当增大。
- INT_ST1（中断状态 1 寄存器）的地址可以参见《Hi35xx xx Camera SoC 用户指南》视频接口章节 MIPI Tx 小节。

示例：

假设从屏幕厂商中拿到的数据为：

- MIPI Tx 输出模式为 DSI_VIDEO
- 数据格式为 BURST_MODE
- 输出数据类型为 OUT_FORMAT_RGB_24_BIT

且给出了屏幕所需要的时序信息：

- 水平有效区（HACT）：1080（像素）
- 水平后消隐（HBP）：20（像素）
- 水平前消隐（HFP）：130（像素）
- 水平同步时序（HSA）：8（像素）
- 垂直有效区（VACT）：1920（行数）
- 垂直后消隐（VBP）：26（行数）
- 垂直前消隐（VFP）：16（行数）
- 垂直同步时序（VSA）：10（行数）
- 设备帧率为 60fps

则可得到 MIPI Tx 的设备属性的配置为：

```
combo_dev_cfg_t MIPI_TX_1080X1920_60_CONFIG =
{
    .devno  = 0,
    .lane_id  =  {0, 1, 2, 3},
    .output_mode  =  OUTPUT_MODE_DSI_VIDEO,
    .output_format  =  OUT_FORMAT_RGB_24_BIT,
    .video_mode  =  BURST_MODE,
    .sync_info = {
        .vid_pkt_size  = 1080,
        .vid_hsa_pixels  = 8,
        .vid_hbp_pixels  = 20,
        .vid_hline_pixels  = 1238,
        .vid_vsa_lines  = 10,
        .vid_vbp_lines  = 26,
```



```
        .vid_vfp_lines   = 16,
        .vid_active_lines  = 1920,
        .edpi_cmd_size  = 0,
    },
    .phy_data_rate  =  880,
.pixel_clk  =  146480,
 };
```

其中 phy_data_rate 和 pixel_clk 若计算出的值为小数，可以向上取整。

步骤 2 配置 MIPI Tx 屏幕初始化序列

屏幕一般都有初始化的过程，RGB LCD 屏一般通过 I2C 或者 SPI 发送数据或者命令来进行屏幕初始化，MIPI LCD 屏则是通过 MIPI Tx PHY 接口来发送指定类型的数据包。

初始化序列由屏幕厂商提供。

屏的初始化序列一般包括像素格式、数据刷新方向、Gamma 配置等，初始化序列中每一个指令具体含义，请在 MIPI LCD 屏的说明书中查找。

SDK 提供了结构体 cmd_info_t，如需将该结构体中每一项都赋值，可通过 HI_MIPI_TX_SET_CMD 配置下去，建议在 HI_MIPI_TX_ENABLE 之前调用，此时 mipi 工作于 LP 模式，数据以低速发送，兼容性较好，HI_MIPI_TX_GET_CMD 读操作也建议在 HI_MIPI_TX_ENABLE 之前调用。

cmd_info_t 结构体定义如下所示。

```
typedef struct
{
    unsigned int        devno;
    unsigned short      data_type;
    unsigned short      cmd_size;
    unsigned char       *cmd;
} cmd_info_t;
```

MIPI 命令一般由，数据类型+数据地址+数据索引+数据 1 +数据 2+…+数据 N，数据索引为数据的个数。数据地值和数据一般为 16 位。

屏幕厂商提供的初始化序列一般有寄存器地址和对应的数据，因此需要根据屏幕厂商给的序列，填充数据类型和数据索引。

填充 cmd_info_t 可参考表 2-2。

表2-2 cmd_info_t 结构体赋值示意表

成员名称	描述
devno	MIPI Tx 设备号，配 0。

成员名称	描述
data_type	写命令数据类型，即 DCS(DisplayCommandSet)(指令集)中的 Data Type。根据数据个数选择数据类型。 类型 1、当只有寄存器地址没有数据时，数据类型选择 0x5 或者 0x13，一般情况下通用，具体使用哪个请咨询厂商或者查询屏幕说明书； 类型 2、有寄存器地址和一个数据时，数据类型选择 0x15 或者 0x23，一般情况下通用，具体使用请咨询厂商； 类型 3、有寄存器地址且数据个数大于等于两个数据类型一般用 0x39 或者 0x29，一般情况下通用，具体使用请咨询厂商。
cmd_size	数据类型个数。但是只有大于或等于两个数据时才用来表示数据个数。 类型 1、当没有数据时，为寄存器地址； 类型 2、当一个数据时，低八位为地址，高八位为数据。
cmd	命令数据指针。 类型 1、类型 2、当小于两个数据时，可置为 NULL； 类型 3、否则为数据。

注意

对于命令数据类型参数的配置的选择，需要咨询厂商。如果没有得到厂商支持，建议没有数据时选择 0x5，有一个数据时选择 0x15，多个数据时选择 0x39。

下面举例说明三种配置方式：

- MIPI TX 只发送地址 cmd_info_t 结构体配置示例代码。

  当需要发送地址 0x11 时，配置如下所示。

```
cmd_info.devno      = 0;
cmd_info.cmd_size   = 0x11;
cmd_info.data_type  = 0x05;
cmd_info.cmd       = NULL;
s32Ret = ioctl(fd, HI_MIPI_TX_SET_CMD, &cmd_info);
if (HI_SUCCESS != s32Ret)
{
    printf("MIPI_TX SET CMD failed\n");
    return;
}
```

- MIPI TX 发送地址和一个数据 cmd_info_t 结构体配置示例代码。

当需要往 0x36 地址上写数据 0x48 时，配置如下所示。

```
cmd_info.devno     = 0;
cmd_info.cmd_size   = 0x4836;
cmd_info.data_type  = 0x15;
cmd_info.cmd       = NULL;
s32Ret = ioctl(fd, HI_MIPI_TX_SET_CMD, &cmd_info);
if (HI_SUCCESS != s32Ret)
{
    printf("MIPI_TX SET CMD failed\n");
    return;
}
```

- MIPI TX 发送地址和多个数据 cmd_info_t 结构体配置示例代码。

  当需要往 0xe1 地址上依次写数据 0xf0，0x03..0x23 时，配置如下所示。

```
cmd[0] = 0xe1;
cmd[1] = 0xf0;
cmd[2] = 0x03;
cmd[3] = 0x23;
cmd[4] = 0xbc;
cmd[5] = 0xa7;
cmd[6] = 0xcc;
cmd[7] = 0xba;
cmd[8] = 0xbc;
cmd[9] = 0xbb;

cmd_info.devno     = 0;
cmd_info.cmd_size   = 10;
cmd_info.data_type  = 0x39;
cmd_info.cmd       = cmd;
s32Ret = ioctl(fd, HI_MIPI_TX_SET_CMD, &cmd_info);
if (HI_SUCCESS != s32Ret)
{
    printf("MIPI_TX SET CMD failed\n");
    close(fd);
    return;
}
```


步骤 3 读取 MIPI 屏幕初始化序列值（此步骤用于调试获取配置，如屏幕厂商信息，单起业务时可以不做，建议在 HI_MIPI_TX_ENABLE 之前调用）。

sync_info_t 结构体定义如下。

```
typedef struct
```



```
{
unsigned int     devno;
unsigned short   data_type;
unsigned short   data_param;
unsigned short   get_data_size;
unsigned char   *get_data;
} sync_info_t;
```

成员名称	描述
devno	MIPI Tx 设备号，配 0。
data_type	命令数据类型，即 DCS(DisplayCommandSet)(指令集)中的 Data Type。读命令数据类型一般采用 0x06 或者 0x04、0x14、0x24，具体使用哪个请咨询厂商或者查询屏幕说明书。
data_param	数据参数，低八比特为第一个参数，高八比特为第二个参数、不用时填 0。
get_data_size	读取数据的长度。
*get_data	获取到的数据存放地址指针，需要用户分配。

当读取 0xda 的一个字节：

```
cmd_info.devno = 0;
cmd_info.cmd_size = 0x01;
cmd_info.data_type = 0x37;
cmd_info.cmd = NULL;
s32Ret = ioctl(fd,HI_MIPI_TX_SET_CMD,&cmd_info);
if(HI_SUCCESS != s32Ret)
{
    Printf("MIPI_TX SET CMD failed\n");
}

uleep(10000);

char data_back[10] ={0};
memset(data_back,0x0,10);
get_cmd_info.data_param = 0xda;
get_cmd_info.data_type = 0x14;
get_cmd_info.devno = 0;
get_cmd_info.get_data =data_back;
get_cmd_info.get_data_size =1;
```



```
s32Ret = ioctl(g_mipitxfd, HI_MIPI_TX_GET_CMD, &get_cmd_info);
if (HI_SUCCESS != s32Ret)
{
printf("MIPI_TX SET CMD failed\n");
close(g_mipitxfd);
return;
}
```

当读取 0xa1 的多个字节：

```
cmd_info.devno = 0;
cmd_info.cmd_size = 0x04;
cmd_info.data_type = 0x37;
cmd_info.cmd = NULL;
s32Ret = ioctl(fd,HI_MIPI_TX_SET_CMD,&cmd_info);
if(HI_SUCCESS != s32Ret)
{
    Printf("MIPI_TX SET CMD failed\n");
}

uleep(10000);

char data_back[10] ={0};
memset(data_back,0x0,10);
get_cmd_info.data_param = 0xa1;
get_cmd_info.data_type = 0x24;
get_cmd_info.devno = 0;
get_cmd_info.get_data =data_back;
get_cmd_info.get_data_size =4;
s32Ret = ioctl(g_mipitxfd, HI_MIPI_TX_GET_CMD, &get_cmd_info);
if (HI_SUCCESS != s32Ret)
{
printf("MIPI_TX SET CMD failed\n");
close(g_mipitxfd);
return;
}
```


注意

读取多个字节时，数据类型为 0x14 还是 0x24 由屏幕决定。

2.2.5 配置 VO 输出序列

当对接一款屏幕时，通过设置 VO 设备公共属性 HI_MPI_VO_SetPubAttr 接口的方式选择用户时序等配置信息。该接口的使用方法在《HiMPP V4.0 媒体处理软件开发参考》中“视频输出章节”中可以查到。配置用户时序接口参数类型 VO_PUB_ATTR_S，结构体如下所示。

```
typedef struct hiVO_PUB_ATTR_S
{
    HI_U32                u32BgColor;
    VO_INTF_TYPE_E         enIntfType;
    VO_INTF_SYNC_E         enIntfSync;
    VO_SYNC_INFO_S        stSyncInfo;
} VO_PUB_ATTR_S;
```

VO_PUB_ATTR 中，enIntfType 表示的是接口类型。

- 确认用户时序接口类型

  enIntfType 的值由屏本身决定。一般所配项如表 2-3 所示。

表2-3 VO_INTF_TYPE_E 结构体赋值示意表

成员名称	描述
VO_INTF_MIPI	MIPI 输出
VO_INTF_MIPI_SLAVE	MIPI_SLVAE 输出，即屏通过 TE 信号引脚主动发同步信息给主芯片，一般用于 Command Mode 的 MIPI 屏，当前仅 HI3519A 和 HI3556A 支持。

- 确认用户时序类型

  enIntfSync 的值表示时序类型，一般选择 VO_OUTPUT_USER。

表2-4 VO_INTF_SYNC_E

成员名称	描述
VO_OUTPUT_USER	用户时序类型

- 填充用户时序信息

  一般来说，如图 2-5 所示，是某典型 MIPI 屏的像素区域示意图，而对于 VO 来说，User 时序对应的 LCD 像素区域示意图则如图 2-6 所示。用户时序结构体的 VO_SYNC_INFO_S 数据类型如下。

```
typedef struct hiVO_SYNC_INFO_S {
    HI_BOOL bSynm;
    HI_BOOL bIop;
    HI_U8 u8Intfb;
```



```
    HI_U16 u16Vact;
    HI_U16 u16Vbb;
    HI_U16 u16Vfb;

    HI_U16 u16Hact;
    HI_U16 u16Hbb;
    HI_U16 u16Hfb;

    HI_U16 u16Hmid;
    HI_U16 u16Bvact;
    HI_U16 u16Bvbb;
    HI_U16 u16Bvfb;

    HI_U16 u16Hpw;
    HI_U16 u16Vpw;

    HI_BOOL bIdv;
    HI_BOOL bIhs;
    HI_BOOL bIvs;
} VO_SYNC_INFO_S;
```


图2-5 MIPI DSI 协议下 MIPI像素区域示意图

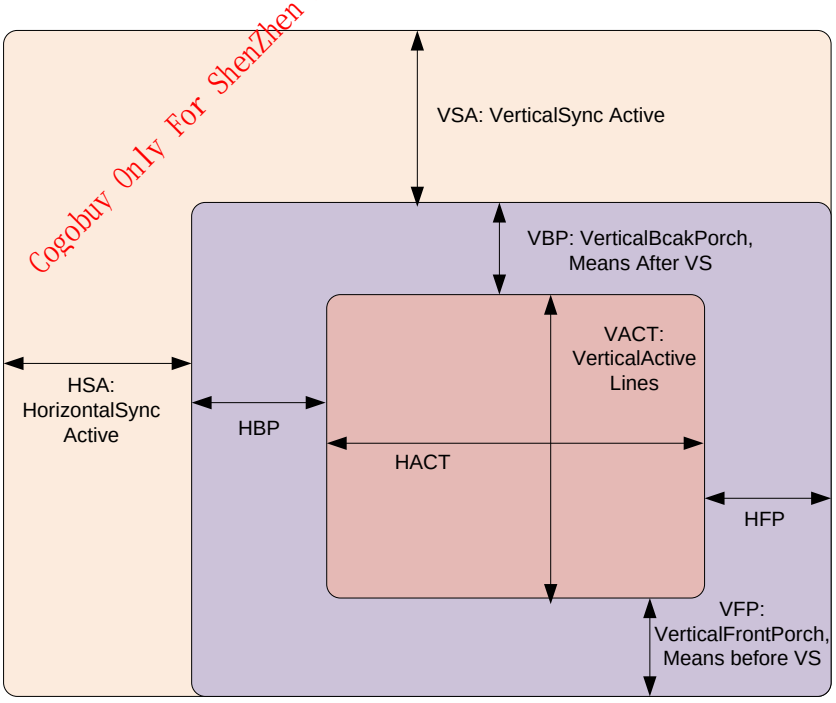
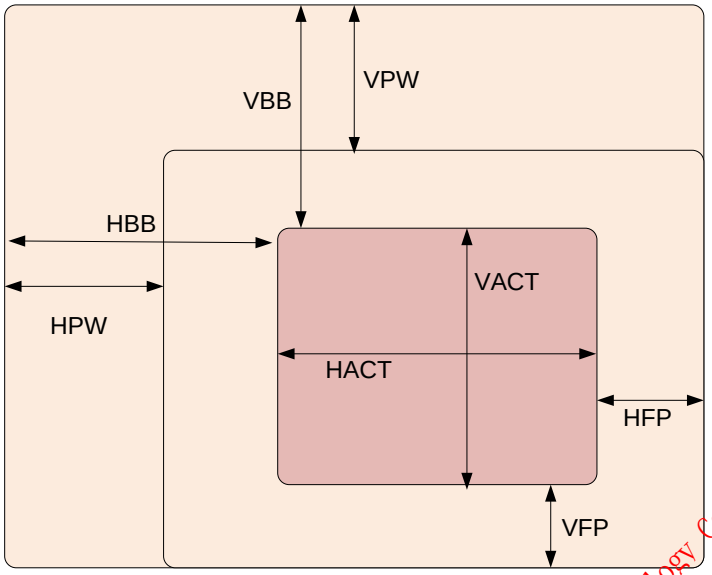


图2-6 VO User 时序下的 LCD 像素区域示意图



- 图 2-6 中 VBP 垂直消隐后肩对应中图 2-5 的 VSA+VBP；
- 图 2-6 中 HBP 水平消隐后肩对应图 2-5 中的 HSA+HBP。
- 图 2-6 中 HPW 对应图 2-5 中的 HSA。
- 图 2-6 中 VPW 对应图 2-5 中的 VSA。
- 图 2-6 中 HACT 对应图 2-5 中的 HACT。
- 图 2-6 中 VACT 对应图 2-5 中的 VACT。

表2-5 VO_SYNC_INFO_S 结构体赋值示意表

成员名称	描述
bSynm	同步模式。参数无意义配 0。
bIop	0 为隔行时序，1 为逐行时序。MIPI 和 LCD 配置为 1。
u8Intfb	输出接口位宽。参数无意义配 0。
u16Vact	垂直有效区，单位为行。根据屏幕手册可知。
u16Vbb	垂直消隐后肩，单位为行。根据屏幕手册可知(VSA+VBP)。
u16Vfb	垂直消隐前肩，单位为行。根据屏幕手册可知。
u16Hact	水平有效区，单位为像素。根据屏幕手册可知。
u16Hbb	水平消隐后肩，单位为像素。根据屏幕手册可知（HSA+HBP）。
u16Hfb	水平消隐前肩，单位为像素。根据屏幕手册可知。

成员名称	描述
u16Hmid	底场垂直同步有效像素值。逐行时序时无意义，置为 1。
u16Bvact	底场垂直有效区，逐行时有效，单位为行。逐行时序时无意义，置为 1。
u16Bvbb	底场垂直消隐后肩，逐行时有效，单位为行。逐行时序时无意义，置为 1。
u16Bvfb	底场垂直消隐前肩，逐行时有效，单位为行。逐行时序时无意义，置为 1。
u16Hpw	水平同步信号的宽度，单位为像素。
u16Vpw	垂直同步信号的宽度，单位为行。
bIdv	数据有效信号的极性。0 为高有效，1 为低有效。MIPI 接口固定要求高有效。
bIhs	水平有效信号的极性，0 为高有效，1 为低有效。MIPI 接口固定要求高有效。
bIvs	垂直有效信号的极性，0 为高有效，1 为低有效。MIPI 接口固定要求高有效。

用于配置 MIPI 屏时得到的相同的屏幕参数时，用户时序的配置为：

```
VO_PUB_ATTR_S stPubAttr;
. u32BgColor = 0xFF,
. enIntfType = VO_INTF_MIPI,
. enIntfSync = VO_OUTPUT_USER,
.stSyncInfo {
    . bSynm = 0,
    . bIop  = 1,
    . u8Intfb = 0,
    . u16Vact = 1920,
    . u16Vbb = 26,
    . u16Vfb = 16,
    . u16Hact = 1080,
    .u16Hbb = 20,
    . u16Hfb = 130,
    .u16Hmid = 0,
    . u16Bvact = 0,
    . u16Bvbb = 0,
    . u16Bvfb = 0,
    . u16Hpw = 8,
    . u16Vpw = 10,
    . bIdv = 0,
```



```
    . bIhs = 0,
    . bIvs = 0,
  }
}
```

2.2.6 配置 VO 输出时钟

当配好 User 时序后，用户还需要配置 VO 的时钟频率。VO 接口输出的时钟频率即为用户要求设置的时钟频率。具体频率值可以时序和帧率信息根据公式计算通过《RGB_MIPI 屏幕时钟时序计算器》的表格自动生成。VO 接口输出时钟频率为时钟源输出时钟除以分频后得到。因此，为了得到接口输出时钟，需要先确定分频比，然后确认时钟源时钟频率。之后再调用 HI_MPI_VO_SetUserIntfSyncInfo 函数配置 VO_USER_INTFSYNC_INFO_S 结构体配置时钟,相关接口的使用可以在《HiMPP V4.0 媒体处理软件开发参考》的“视频输出”章节中可以查到。

用户时序信息

```
typedef struct hiVO_USER_INTFSYNC_INFO_S {
    VO_USER_INTFSYNC_ATTR_S stUserIntfSyncAttr;
    HI_U32                  u32PreDiv;
    HI_U32                  u32DevDiv;
    HI_BOOL                 bClkReverse;
} VO_USER_INTFSYNC_INFO_S;
```

时钟源属性

```
typedef struct hiVO_USER_INTFSYNC_ATTR_S {
    VO_CLK_SOURCE_E enClkSource;
    union {
        VO_USER_INTFSYNC_PLL_S stUserSyncPll;
        HI_U32 u32LcdMClkDiv;
    };
} VO_USER_INTFSYNC_ATTR_S;
```

时钟源选择

```
typedef enum hiVO_CLK_SOURCE_E {
    VO_CLK_SOURCE_PLL,
    VO_CLK_SOURCE_LCDMCLK,
    VO_CLK_SOURCE_BUTT
} VO_CLK_SOURCE_E;
```

PLL 参数配置

```
typedef struct hiVO_USER_INTFSYNC_PLL_S {
    HI_U32 u32Fbdiv;
    HI_U32 u32Frac;
    HI_U32 u32Refdiv;
```

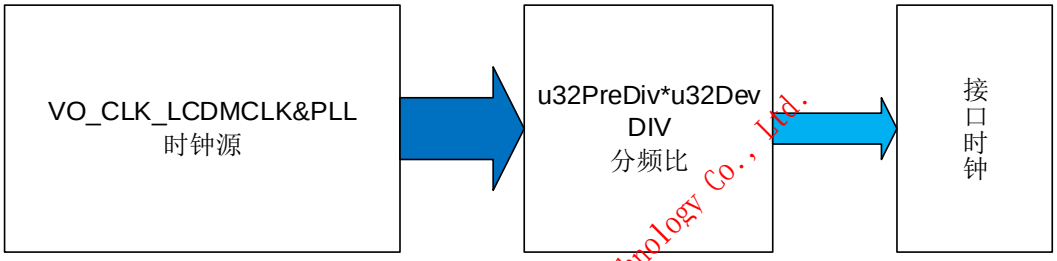


```
    HI_U32 u32Postdiv1;
    HI_U32 u32Postdiv2;
} VO_USER_INTFSYNC_PLL_S;
```

2.2.6.2 配置分频比

“分频比”的值为“前置分频”（u32PreDiv）与“设备时钟分频”（u32DevDiv）的积。“前置分频”与“时钟分频”的概念可以通过查询《HiMPP V4.0 媒体处理软件开发参考》中“视频输出”章节中可以查到。

图2-7 时钟源时钟频率与接口时钟之间的关系



各个芯片的 VO CRG 配置中，都有一个分频比的配置。

- MIPI DSI 输出，则一般选择“不分频”。
- MIPI 屏幕不需要分频，因此分频比配置为 1，即将“前置分频”(u32PreDiv)和“设备时钟分频”（u32DevDiv）均设置为 1。

2.2.6.3 时钟反向

对于 MIPI 屏幕，直接固定反向即可。

2.2.6.4 配置分频时钟

关于时钟计算和配置方法，可以参考《HiMPP V4.0 媒体处理软件开发参考》中 VO_USER_INTFSYNC_INFO_S 中的介绍将上述值配到该结构体中。

查看芯片手册找到 pll 选择与计算方法。也可通过 CRG 寄存器表格找到 VDP 时钟控制寄存器，用寄存器查看方法查看 VDP 选择了哪种时钟源。

对 Hi3516CV500 芯片，如：himd.l 0x12010108

VDP 的 LCD 分频时钟或者 PLL 时钟频率计算方法：

以一款 320*240 帧率为 60 的屏为例，若此款屏的说明说中给出的时序信息如下：

HACT	HBP	HFB	HSA	VACT	VBP	VFB	VSA
320	64	7	1	240	14	9	1

根据表 1-3 中的对应关系可得：

u16Hact：320；u16Hbb：65；u16Hfb：7；u16Vact：240；u16Vbb：15；u16Vfb：9。

首先计算时钟频率，单位 MHz。

X = （u16Hact + u16Hbb + u16Hfb）* (u16Vact + u16Vbb + u16Vfb) * framerate * u32PreDiv*u32DevDiv / 10^6。

按上述屏来计算的话，X = （320 + 65 + 7）*（240 + 14 + 9）* 60* 1 * 1 /10^6 = 6.209

LCD 分频系数 u32LcdMClkDiv = （X / 1188）* (2^27), 此时 X 最大为 75MHz。

因此，u32LcdMClkDiv = （6.209 / 1188）* (2^27) = 0xAB448。

PLL 时钟频点的 u32Fbdiv，u32Frac、u32Refdiv、u32Postdiv1 和 u32Postdiv2 参数。

公式：X = 24 * （u32Fbdiv + u32Frac/（2^24））/ u32Refdiv/ u32Postdiv1/ u32Postdiv2。

已上述屏为例， u32Fbdiv，u32Frac、u32Refdiv、u32Postdiv1 和 u32Postdiv2 分别为 1、0x35935F、2、2、1。

2.3 配置验证与调试

2.3.1 管脚复用与驱动能力确认

步骤 1 确认控制接口 SPI/I2C 的接口管脚复用是否配置正常。

步骤 2 确认屏幕的数据接口(LCD_DATA、HSYNC、VSYNC、CLK 等)的接口管脚复用是否配置正常。驱动能力在调试初期可以先配置较高档位，待调亮屏幕后，再调优，以免损坏器件。

可以参照《Hi35XX_PINOUT_CN》和硬件设计图，通过 himd + 寄存器地址，来读出管脚复用状态。

如果状态不符合，请在代码的对应地方，修改不匹配的复用关系。

以 HI3559V200 芯片的 LCD_CLK 引脚为例，在《Hi3559V200_PINOUT_CN》中找到 LCD_CLK 所在的寄存器地址，如图 2-8 所示。

图2-8 LCDCLK 复用关系示意图

0x112F_0034	0x0400	31:11	保留。
		10	电平转换速率控制，为0时电平转换速率快，为1时电平转换速率慢。
		9	下拉电阻使能，高有效。
		8	上拉电阻使能，高有效。
		7:4	驱动能力，0~7对应IO驱动能力IO4_档位1~档位8。
		3:0	功能选择: 0:GPIO0_6 2:LCD_CLK 3:VOU_CLK

当读寄存器 0x112F0034 的值 3:0 位不是 0x2，则代表 LCD 屏的 LCD_CLK 的引脚复用关系错误，需要在代码的相应地方修改。

2.3.2 复位操作流程确认

步骤 1 首先确认复位管脚能否正常上拉下拉。

查询硬件设计图，找到复位管脚对应的 GPIO 口。确认对应的管脚的复用关系为 GPIO。

参考芯片手册 GPIO 章节，将该 GPIO 的方向配成输出。

在 Hi3559V200 上，配置 GPIO5_2 的方向为输出为例：

找到 GPIO5 的基地址为 0x120D5000

图2-9 GPIO 口基地址示意图

GPIO 控制器	基地址
GPIO8	0x120D_8000
GPIO7	0x120D_7000
GPIO6	0x120D_6000
GPIO5	0x120D_5000
GPIO4	0x120D_4000
GPIO3	0x120D_3000
GPIO2	0x120D_2000
GPIO1	0x120D_1000
GPIO0	0x120D_0000

将基地址+偏移地址 0x400 的寄存器的第 2 位配成 1。

图2-10 GPIO 方向控制寄存器示意图

GPIO_DIR

GPIO_DIR 为 GPIO 方向控制寄存器。用来配置 GPIO 管脚方向。

Offset Address: 400　　Total Reset Value: 0x00

Bits	Access	Name	Description	Reset
[7:0]	RW	gpio_dir	GPIO 方向控制寄存器。bit[7:0]分别对应 GPIO_DATA[7:0]，各比特可独立控制。 0：输入； 1：输出。	0x00

参考芯片手册 GPIO 章节，分别将该 GPIO 口拉高拉低，通过示波器测量引脚状态是否正确。

在 Hi3559V200 上，拉高拉低 GPIO5_2 为例，前提是已经配置了 GPIO5_2 为输出：

找到对应的 GPIO_DATA 寄存器，为 0x120D5000+0x1>>(2+2)，如果是 GPIO5_3，则是 0x120D5000+0x1>>(3+2)。

再将该寄存器的第 2 位分别配成 0 和 1，对应的是下拉和上拉。

步骤 2 再通过查询屏幕的说明书，确认复位操作是否和屏幕要求一致。如先上拉，再下拉，再上拉，并确认各步骤间的延时要求。

----结束

2.3.3 屏幕控制命令通路验证

海思平台提供的 MIPI Tx 驱动，提供了 HI_MIPI_TX_GET_CMD 用于获取屏幕的寄存器值，HI_MIPI_TX_SET_CMD 用于设置屏幕的寄存器值，可以尝试以下四种方法验证控制命令传输通路，建议在 HI_MIPI_TX_ENABLE 之前调用 HI_MIPI_TX_GET_CMD 与 HI_MIPI_TX_SET_CMD，因为 HI_MIPI_TX_ENABLE 接口调用后会将 mipitx 从 LP 模式切换到 HS 模式，控制命令建议使用 LP 模式：

- 读屏幕的 ID 寄存器，一般为 0xDA 或者 0xA1 寄存器，具体寄存器地址请查询屏幕说明书。

  如下为读 0xDA 一个字节的例子：

  //先用 0x37 命令配置最大返回字节数为 1.在用 0x14 读出 1 个字节。

```
cmd_info.devno     = 0;
cmd_info.cmd_size  = 0x01;
cmd_info.data_type =  0x37;/*set maximum return packet size*/
cmd_info.cmd       = NULL;
ioctl(fd, HI_MIPI_TX_SET_CMD, &cmd_info);

get_cmd_info.data_param = 0xda;
get_cmd_info.data_type = 0x14;
get_cmd_info.devno = 0;
get_cmd_info.get_data =data_back;
get_cmd_info.get_data_size =1;
s32Ret = ioctl(fd, HI_MIPI_TX_GET_CMD, &get_cmd_info);
```

  如下为读 0x1A 五个字节的例子：

  //先用 0x37 命令配置最大返回字节数为 5.在用 0x14 读出 5 个字节。

```
cmd_info.devno     = 0;
cmd_info.cmd_size  = 0x05;
cmd_info.data_type =  0x37;/*set maximum return packet size*/
cmd_info.cmd       = NULL;
ioctl(fd, HI_MIPI_TX_SET_CMD, &cmd_info);

get_cmd_info.data_param = 0xda;
get_cmd_info.data_type = 0x14;
get_cmd_info.devno = 0;
get_cmd_info.get_data =data_back;
get_cmd_info.get_data_size =5;
s32Ret = ioctl(fd, HI_MIPI_TX_GET_CMD, &get_cmd_info);
```

注意：读命令使用 0x14、0x24、0x4 或者 0x6 取决于屏幕，一般常见的屏幕使用 0x14 和 0x6 的居多。

另外为了方便调试，除了利用接口调用来实现读写，海思还提供了读写的小工具，mipitx_write，mipitx_read，注意该工具无配置 mipi 属性功能，需要在 HI_MIPI_TX_SET_DEV_CFG 之后使用。以 Hi3559V200_MobileCam 为例，该调试工具放置于 board/Hi3559V200_MobileCam_SDK_V1.0.1.1/amp/a7_liteos/mpp/tools，如需使用，需在 Huawei LiteOS 侧调用 tools_cmd_register()，之后在命令行即可使用 mipitx_write，mipitx_read。

- 找一个屏幕可写可读的寄存器，尝试修改后读出，看操作是否符合预期。
- 利用屏幕的调试模式或者厂测模式，一般通过简单的发送几个控制命令，即可以进入这种模式，看到一些画面，可以以此检测命令发送通路,具体如何操作请查询屏幕说明书。
- 另外可以利用示波器观测发送控制命令时的波形，看是否匹配屏幕需求，或者有协议分析仪的话也可以直接分析传输数据，不过这种方法依赖对 MIPI 协议的理解，容易失误。

控制命令通路如果不通，请检查硬件连接与供电、reset 引脚工作状态（注意时序、电平匹配等），MIPI Tx 的 combo_dev_cfg_t 结构体属性配置，重点检查 phy_data_rate 与 lane_id 成员。

2.3.4 背光控制

背光控制分三种：

- 可以动态调整背光亮度。

  动态调整背光亮度的 LCD 屏，多数情况是通过 PWM 调整占空比来实现亮度调整，需实现 PWM 驱动。
- 只能通过硬件配置亮度（即无法动态配置）。
- 为用 mipi 发送打开背光的控制命令，可以通过屏幕驱动 IC 来控制背光。

具体哪种方法，请查阅屏幕说明书，并确保屏幕背光亮起。

2.3.5 VO 输出时钟和时序确认

2.3.5.1 VO 输出时钟

VO 的 proc 信息中的 IntRate 为统计的设备帧率，（Hact + Hbb + Hfb）* (Vact + Vbb + Vfb) * framerate = 输出时钟。在 Hact、hbb 等信息正确时，如果该帧率符合预期，则说明输出时钟正确。

在业务程序运行过程中，通过终端输入命令 cat /proc/umap/vo 的方法查看 VO 模块的 proc 信息。

图2-11 部分 VO 的 proc 信息

```
[VO] Version: [Hi3516CV500_MPP_V1.0.0.0 B010 Release], Build Time[Mar 26 2019, 17:26:57]

-----DEVICE CONFIG-----------------------------------------------------------
  DevId   DevEn       Mux1       Mux2       Mux3      IntfSync        BkClr  DevFrt
     0       Y       MIPI                                  -         0xff      60

-----DEVICE CLOCK INFO-------------------------------------------------------
  DevId   DevEn  ClkSource   FbDiv    Frac     RefDiv   PostDiv1  PostDiv2   LCDMCLK  VoDevDiv  VoPreDiv ClkReverse
     0       Y        PLL      70  214748          2          4         3                 1         1          N
```

```
-----DEV Int Status---------------------------------------------------------------
   DevId IntRate IntTime MaxIntT TimePrM  IntGapT  MaxGapT
      0      59     203     216   11342    16668    16760
```

可以通过 VO 的 proc 信息查看到配置的时钟源和时钟源参数信息以及 VDP 逻辑的中断也就是设备帧率信息。

此外还可以当媒体业务运行在 LINUX 端时，还可以通过命令：watch –n 10 cat /proc/interrupts 查看每 10 秒 VO 产生的中断个数，通过计算后得到实际帧率。

图2-12 VO 设备中断信息

```
~ # watch -n 10 cat /proc/interrupts
Every 10s: cat /proc/interrupts
```

```
 49:       2636          0     GIC-0  90 Level     VO Int
```

```
 49:       3237          0     GIC-0  90 Level     VO Int
```

根据图 2-12 所示，可得（3237-2636）/10 = 60。

2.3.5.2 时序确认

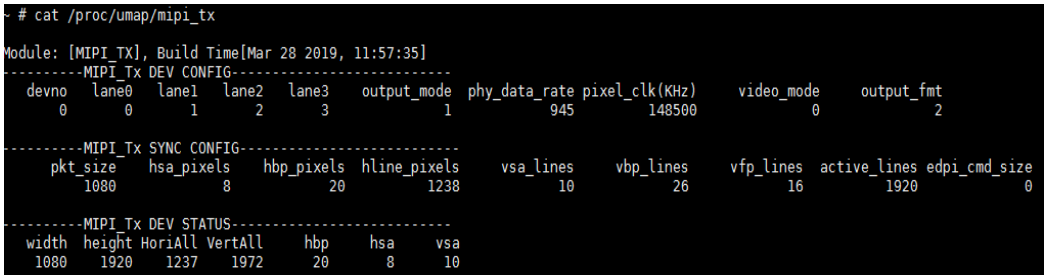
通过 MIPI Tx 的 proc 信息可以获取 MIPI Tx 检测到的 vo 输出的时序信息，该 proc 的说明在例如下图中的 MIPI Tx DEV STATUS 字段中图 2-5 的信息。具体如下：

- Width 对应 HACT
- height 对应 VACT
- HoriAll 对应：HSA+HBP+HACT+HFP
- VertAll 对应：VSA+VBP+VACT+VFP
- hbp 对应 HBP
- hsa 对应 HSA
- vsa 对应 VSA

注意

这 MIPI Tx 的时序信息，需要折算成 VO 时序信息后再和 VO 的配置参数比较。

图2-13 MIPI Tx 的 proc 信息



2.3.6 VO COLORBAR

colorbar 是一种重要的调试手段，通过 colorbar 的显示，我们能够看到 VDP 逻辑产生的图像，颜色和时序的信息。

如果 VDP 的配置上有问题时大多数都可以在 colorbar 上显示出来。

如果 colorbar 的图像显示出现错位或者残缺，则可能存在时钟时序的问题。

colorbar 的调试手段可以很好的帮助我们定位问题，定位时需要根据具体现象具体分析。

调试手段：

通过设置 VDP 寄存器的方式启动 VDP 设备 colorbar。根据芯片手册查找 VDP 寄存器 DHD0_CTRL 或者是 DHD1_CTRL（DHD0 还是 DHD1 取决于对接屏幕的是哪个设备）。在不改变该寄存器其他位值的状态下，配置 colorbar 使能位 cbar_en 使能；配置 colorbar 色彩空间选择位 cbar_sel 为 VGA；配置 colorbar 模式选择位 cbar_mode 水平或者垂直。配置 DHD0 寄存器更新位 regup 为 1。

以 Hi3516CV500 为例：

查看寄存器：himd.l 0x1144d000

打开 colorbar：himm 0x1144d000 0xc0000011

关闭 colorbar：himm 0x1144d000 0x80000011

其中 0x11440000 为 VDP 寄存器基址，0xd000 为 DHD0_CTRL 寄存器偏移地址

colorbar 样式：

图2-14 VDP 水平 colorbar 样式：

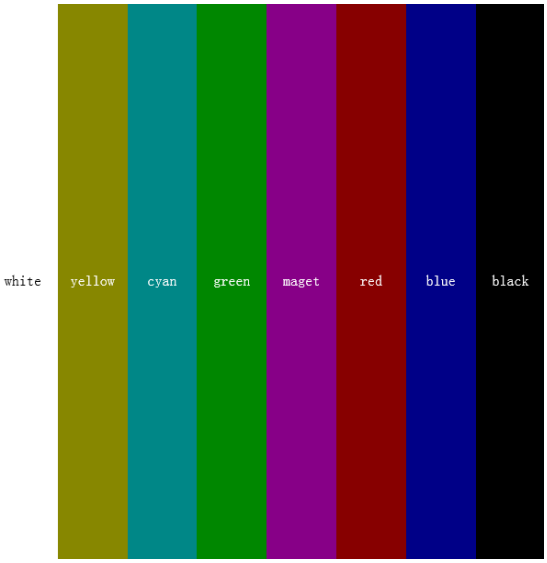


图2-15 VDP 垂直 colorbar 样式：

2.3.7 MIPI 配置属性查看方法

查看配置的 MIPI 设备属性由两种方法：

- 方法一：通过查看 MIPI Tx 的 proc 信息的方法查看配置属性。proc 中显示的是用户配置的参数。

图2-16 proc 显示参数

```
~ # cat /proc/umap/mipi_tx

Module: [MIPI_TX], Build Time[Mar 28 2019, 11:57:35]
----------MIPI_Tx DEV CONFIG--------------------------
  devno   lane0   lane1   lane2   lane3    output_mode  phy_data_rate pixel_clk(KHz)     video_mode     output_fmt
     0       0       1       2       3              1            945         148500              0              2

----------MIPI_Tx SYNC CONFIG--------------------------
     pkt_size    hsa_pixels    hbp_pixels  hline_pixels     vsa_lines     vbp_lines     vfp_lines  active_lines edpi_cmd_size
         1080             8            20          1238            10            26            16          1920             0
```

- 方法二：通过查看寄存器的方式查看配置的设备属性值。

  如：packet 大小配置寄存器 VID_PKT_SIZE；HSA 时间配置寄存器 VID_HSA_TIME；HBP 时间配置寄存器 VID_HBP_TIME；HLINE 时间配置寄存器 VID_HLINE_TIME；VSA 行数配置寄存器 VID_VSA_LINES；

  VBP 行数配置寄存器 VID_VBP_LINES；VFP 行数配置寄存器 VID_VFP_LINES；VACTIVE 行数配置寄存器 VID_VACTIVE_LINES；

  其中 VID_HSA_TIME，VID_HBP_TIME，VID_HLINE_TIME 为对应配置参数进过计算后得到的值，可能跟所配置的值大小不一致，获取到的为实际配置到寄存器中的值。换算关系为 VID_HSA_TIME = ((phy_data_rate +26)/27*27) *has_pixels *125 /pixel_clk;

  - 与设置一致寄存器，以寄存器 VID_PKT_SIZE 为例，查看方法如图 2-17 所示。

图2-17 寄存器配置信息

```
~ # himd.l 0x1127003c
*** Board tools : ver0.0.1_20121120 ***
[debug]: {source/utils/cmdshell.c:168}cmdstr:himd.l
====dump memory 0x1127003C====
0000:  00000438 00000000 00000000 00000006
```

  - 与配置有换算关系，以 VID_HSA_TIME 为例，查看方法如图 2-18 所示。

图2-18 寄存器配置信息

```
~ # himd.l 0x11270048
*** Board tools : ver0.0.1_20121120 ***
[debug]: {source/utils/cmdshell.c:168}cmdstr:himd.l
====dump memory 0x11270048====
0000:  00000006 0000000f 000003d8 0000000a
```

计算方法:

VID_HSA_TIME = ((phy_data_rate +26)/27*27) *has_pixels *125 /pixel_clk

= ((945+26)/27*27)*8*125/148500

= 6

查看配置的 MIPI 初始化序列的方法：

通过 MIPI Tx 的接口 HI_MIPI_TX_GET_CMD，可以从屏幕端获取配置的初始化序列。方法见上文 2.2.2 节中“配置 MIPI 屏复位”。

2.3.8 MIPI Tx colorbar 调试方法

由于海思平台的 MIPI Tx 模块与 VO 模块相对独立，可以通过 MIPI Tx 的 colorbar 功能，先排查 MIPI Tx 的属性是否配置正确。

如果打开 MIPI Tx colorbar 可以显示但是不开 MIPI Tx colorbar 的时候无法显示，则说明 VO 在送图像给 MIPI Tx 的过程中出现了问题。具体问题还需结合上文进一步分析在此列出几种可能的场景。

- colorbar 显示错位，本该从屏幕最边缘开始的颜色，偏到了中间或者其他的位置，这种问题大概率是 MIPI TX 的时序配置问题。
- colorbar 颜色异常，屏幕可能会有配置 RGB 的排序相关的寄存器，海思平台只能发送 RGB 顺序，不可调整。另外 MIPITX 的 output_format 配置异常也可能导致颜色异常。

  如果配置 MIPI Tx colorbar 无法显示出图形，则说明 MIPI Tx 送图像给屏幕的过程出了问题。

调试手段：

调试通过配置寄存器的方法打开和关闭 MIPI Tx 的 colorbar。

- 打开 colorbar 步骤：

步骤 1 通过查找芯片手册中 MIPI Tx 章节找到寄存器 OPERATION_MODE 的地址，配置值 0x0。

步骤 2 找到寄存器 VID_MODE_CFG 的地址，配置 bit16（vpg_en）为 1，其余 bit 保持不变。

步骤 3 找到寄存器 PWR_UP 的地址，先配置 0x0，在配置值为 0x1。

---结束

- 关闭 colorbar 步骤：

步骤 1 通过查找芯片手册中《视频接口》章节中“MIPI Tx”小节，找到寄存器 VID_MODE_CFG 的地址，配置 bit16（vpg_en）为 0，其余 bit 位保持不变。

步骤 2 找到寄存器 OPERATION_MODE 的地址，配置值为 0x80050000。

步骤 3 找到寄存器 PWR_UP 的地址，先配置 0x0，在配置值为 0x1。

----结束

以 Hi3516CV500，output_mode 为 OUTPUT_MODE_DSI_VIDEO，video_mode 为 BURST_MODE 为例：

- 打开 colorbar：

  himm 0x11270208 0x0

  himm 0x11270038 0x13f02

  himm 0x11270004 0x0

himm 0x11270004 0x1

- 关闭 colorbar：

  himm 0x11270038 0x3f02

  himm 0x11270208 0x80050000

  himm 0x11270004 0x0

  himm 0x11270004 0x1

其中 0x1127 为 MIPI 寄存器基址；0x00000208 为 OPERATION_MODE 偏移地址；0x00000038 为 VID_MODE_CFG 偏移地址；0x00000004 为 PWR_UP 偏移地址。

注意

MIPI Tx 的 CMD 模式不支持 colorbar。建议在 MIPI Tx 使能后使用 colorbar 调试手段。

Colorbar 样式，如图 2-19 和图 2-20 所示。

图2-19 MIPI 垂直样式

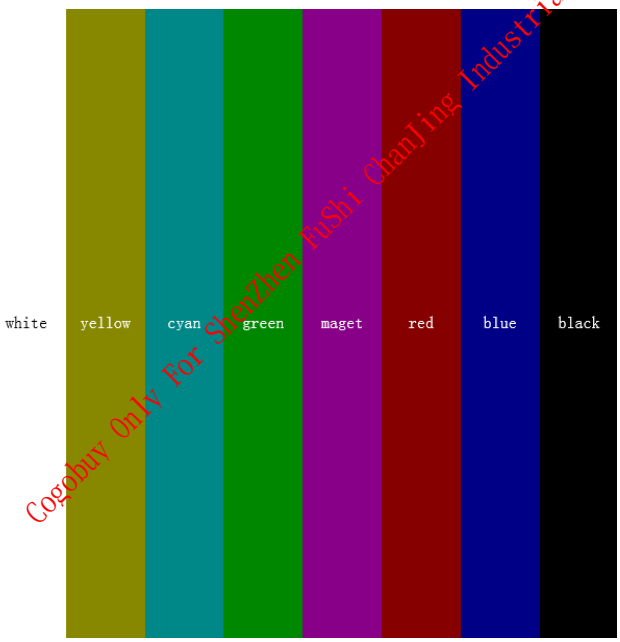
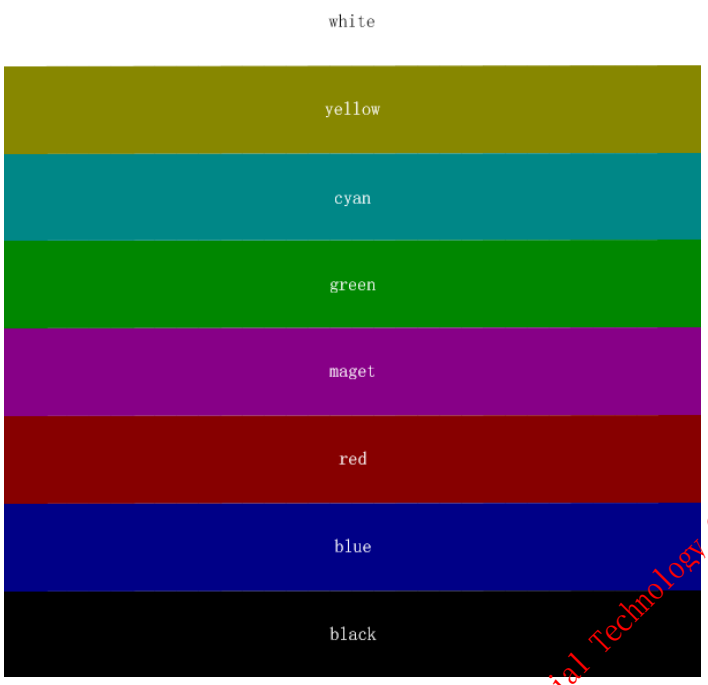


图2-20 MIPI 水平样式



2.4 显示屏调试 FAQ

2.4.1 屏幕全黑，毫无反应

【问题现象】

屏幕完全黑色，没有响应。

【问题分析】

有很多因素会导致屏幕无响应，需要逐个环节排查，例如器件损坏、连接异常、供电异常、管脚复用配置异常、reset 管脚状态异常、VDP 参数配置异常、MIPI 参数配置错误等

【解决方案】

步骤 1 硬件确认。

找硬件工程师确认屏幕供电、连接正常，且器件无损坏，注意各个 IO 口电平匹配。

步骤 2 确认软件流程的正确性。

请比对 2.2 章节配置步骤中的 mipi 设备初始化流程，需要特别注意的是 reset 管脚操作有严格的时间要求，一般为 120ms 后才能接受控制命令。

步骤 3 确认管脚复用与驱动能力配置正确，请参考 2.3.1 管脚复用与驱动能力 章节。

步骤 4 确认控制命令传输通路是否正常，请参考 2.3.3 屏幕控制命令通路验证章节。

步骤 5 确认背光开启。

对于 mipi 屏幕来说，打开背光通常有两种方式，一种为 IO 口电平操作，一般为 GPIO 或者 PWM，一种为用 mipi 发送打开背光的控制命令。具体哪种方法，请查阅屏幕说明书，并确保屏幕背光亮起。

步骤 6 确认 mipitx 的视频时序信息属性配置是否正常。

如果发现控制命令传输通路已经正常，通常可能出在 vdp 或者 mipitx 的参数异常。由于海思平台的 mipitx 模块与 vdp 模块相对独立，可以通过 mipitx 的 colorbar 功能，先排查 mipitx 的属性是否配置正确，mipitx 的 corlorbar 请参考 2.3.8 MIPI Tx colorbar 调试方法章节，colorbar 的开启应在 HI_MIPI_TX_ENABLE 之后。

如果 mipitx 的 colorbar 无显示或者显示位置异常。请先核对 combo_dev_cfg_t 的配置是否正确，详细请参阅 2.2.4 配置 MIPI 屏幕章节。核对之后仍然存在问题，建议重新从步骤 1 开始，若多次前面几个步骤确认无误，有可能存在 mipi d-phy 的时序参数与屏幕侧存在兼容问题，请查看 2.4.14 小节。

步骤 7 确认 vdp 的输出是否正确，是否与 mipitx 的时钟 pixel_clk 匹配。

由于 mipitx 的视频信号来源于 VDP 模块，在步骤 4 的基础上，mipitx 的 colorbar 已经可以正确显示，但 vdp 的图像却显示不出时，考虑问题应该出在 vdp 的属性配置（包括 vdp 的输出时钟）或者 mip 配置结构体 combo_dev_cfg_t 中 pixel_clk 配置。

有关 VDP 的配置方法，请参阅 2.2.5 配置 VO 输出序列，有关输出时钟配置，请参阅配置输出时钟配置。

----结束

2.4.2 屏幕显示位置错误

【问题现象】

图像只在屏幕上显示出一部分。

【问题分析】

这种情况的表象为实际显示的图像在屏幕上只显示一部分，显示图像起始点不是屏幕的初始点。

【解决方案】

一般是主芯片侧的行或者场的消隐区配置与屏幕的配置未匹配导致的，需要将两者保持一致，可以调整屏幕配置，也可以调整芯片侧配置。

2.4.3 屏幕显示裂屏

【问题现象】

显示图像起点位置移至屏幕中间。

【问题分析】

调整前后消隐区大小没有变化。可能是输出像素时钟与时序不匹配。

【解决方案】

通过 1.3 屏幕验证与调试节中的调试手段定位时钟是否正常，如果时钟确定时钟有问题。

2.4.4 屏幕显示图像出现错乱

【问题现象】

点亮屏幕后，发现屏幕图像错乱，打开 MIPI Tx clolorbar 后，发现随机性出现某些行颜色显示刚好反过来了。

【问题分析】

从 colorbar 的现象看，图像的扫描方式出现了问题。扫描出现随机的情况。

【问题原因】

MIPI 屏幕的扫描方式由硬件决定，当配置扫描方式的接口电压不准时，导致扫描方式出现随机变换的情况，因此出现图像错乱。

2.4.5 屏幕显示大小错误

【问题现象】

图像在屏幕上显示时出现实际值大小不匹配的情况。

【问题分析】

这种情况表象为实际显示图像的尺寸超过屏幕尺寸（即显示不全）或者小于屏幕尺寸。

【解决方案】

一般是主芯片侧的帧或者场的有效区配置与屏幕的配置未匹配导致的，需要将两者保持一致，可以调整屏幕配置，也可以调整芯片侧配置。

2.4.6 屏幕显示颜色异常

这种情况表象为图像的颜色与预期发生明显的差别，有多种表象，举例如下。

出现颜色发生变化

【问题现象】

显示图像出现红、蓝、绿出现错乱的情况。

【问题分析】

这种情况可能为 RGB 发送顺序未匹配。

【解决方案】

尝试调整屏幕侧接收 RGB 顺序，有些屏幕可以通过寄存器配置。

显示颜色偏色

【问题现象】

显示颜色不正常，出现偏色现象。

【问题分析】

这种情况可能是配置的 MIPI 设备输出模式不正确，造成 MIPI 设备参数配置不正确。

【解决方案】

更换 MIPI 设备输出模式的配置。MIPI Tx 输出模式，一般为 DSI_VIDEO 或者 DSI_CMD，根据特定屏要求赋值。具体配置需要咨询屏幕厂商。

显示颜色异常

【问题现象】

图像显示的颜色不正常，颜色转换方式不对。

【问题分析】

这种情况有可能是视频层的 CSC 没有配置正确，LCD 或者 MIPI 屏幕需要转换 RGB 的 CSC 输出才行。

【解决方案】

调用 HI_MPI_VO_SetVideoLayerCSC 或者 HI_MPI_VO_SetGraphicLayerCSC 函数设置 CSC 属性。设置方法可参考《HiMPP V4.0 媒体处理软件开发参考》中的“视频输出”章节中相关接口的操作说明。

显示线条有色差

【问题现象】

显示图像的轮廓线处有色差。

【问题分析】

这种情况的表象为线条颜色发生异常，特别是图像的轮廓线，受相邻像素的影响，这种情况一般为时序有细微错误，导致数据传输的时候，该像素的的信息和相邻像素的信息发生了传输错误，这种情况有多种可能。

【解决方案】

硬件干扰较大，请先用示波器看看是否是波形符合预期，如果不符合，可能是走线过长导致（一般飞线才可能出现这种情况），或者是管脚驱动能力配置不足。

视频输出的 clk 相位错误，未和屏幕匹配，需要将主芯片侧和屏幕侧调整到时序正确。

2.4.7 屏幕显示效果不够满意

【问题现象】

图像显示时在亮度、色度、对比度等方面达不到想要的效果。

【问题分析】

屏幕显示效果不够满意。

【解决方案】

调节屏幕显示效果，需要从两个角度出发：

一是调整主芯片测的图像输出效果。

二是调整屏幕侧的显示效果参数。

主芯片测的显示效果可以通过调整 vo 的参数，可以调整亮度对比度等，屏幕侧一般可以调整 gamma 曲线，亮度、色度等，一般屏幕厂商会给出推荐配置，能满足大部分场景需求。

2.4.8 屏幕画面转换时出现残影

【问题现象】

屏幕显示的图像在变换的过程中出现上一帧画面的残留。

【问题分析】

残影一般由 vcom 电压配置不合理或者一些供电管脚电压异常导致。

【解决方案】

请查看屏幕手册说明或者屏幕厂商获取解决办法。

2.4.9 显示帧率低/画面卡顿

【问题现象】

显示的画面出现卡顿的情况。

【问题分析】

一般这种情况是数据时钟频率不够。

【解决方案】

请调整时钟频率。

2.4.10 显示画面较暗，但背光正常

【问题现象】

图像显示画面较暗，但是背光确实正常的。

【问题分析】

可能 RGB 屏部分 DATA 线无数据，硬件设计的时候，没有参考 Hi35XX 配置正确的管脚，导致屏幕与主芯片间部分管脚无数据。

【解决方案】

出现画面较暗的原因较多，遇到这种问题时，可以先排除上述问题分析的可能性。

2.4.11 退出显示业务后重新启动，RGB 屏无法显示

【问题现象】

退出显示业务后重新启动，RGB 屏无法显示，需要重新配置屏幕驱动。

【问题分析】

与 RGB 屏幕的特性有关系，部分 RGB 屏幕在停止显示业务前，需要先关闭屏幕或者让屏幕进入休眠状态，在显示业务重新启动后再打开或唤醒屏幕。

【解决方案】

在退出显示业务前，关闭屏幕或进入休眠状态，在显示业务重新启动之后再打开或者唤醒屏幕。

2.4.12 MIPI 检测不到前端时序行信息

【问题现象】

在调试时发现，VDP 设置的时钟和时序都没有问题，查看中断数也没有问题，但是 MIPI 检测到的前端送每帧行有效和总行数为 0，而每行的有效区像素个数和行的总像素个数正确；

【问题分析】

配置的 MIPI 设备参数中的 phy_data_rate 速率属性值较小，导致 MIPI 不断重启，造成检测到的有效区行数为 0。

【解决方案】

可以尝试如下解决方式：

- 方式 1：增加 phy_data_rate 的值，使之达到比较合理的范围（建议增加至理论计算带宽的 1.2 倍或者更多，注意不要超过屏幕侧的接收范围）。
- 方式 2：对于 Hi3519AV100\Hi3556AV100 芯片，需要确保 MISC_CTRL1 寄存器的 mipitxphy_cmos_mode_enable 比特位配置为 0。关于此寄存器的详细描述请参考《Hi35xx xx Camera SoC 用户指南》。

2.4.13 设置 MIPI 设备属性 phy_data_rate 与实际检测出来的差别较大

【问题现象】

检测到的 MIPI 输出时钟比预期值小很多。

【问题分析】

设置 phy_data_rate 的属性后，从 MIPI 时钟检测到的频率值应该为 phy_data_rate 的一半。实际值与预期相差较大。

【解决方案】

查看 MIPI 供电电压是否为需要的电压。如果不是，修改成预期使用电压。

2.4.14 MIPI D-PHY 时序与屏幕端匹配确认

【问题现象】

MIPITX 的配置与 VDP 的配置已检查完成，外围硬件配置与硬件操作顺序已检查完成，屏幕仍然全黑。

【问题分析】

经由海思 MIPITX 驱动配置出的 MIPI D-PHY，工作于一个兼容性较好的时序参数，但是有可能屏幕对于时序有着比较特别的要求，需要根据屏幕要求的 MIPI D-PHY 时序参数，来修改海思 MIPITX 驱动，以达到发送与接收端时序匹配。

【解决方案】

步骤 1 查阅屏幕的驱动 ic 的说明书，找到关于 MIPI D-PHY 时序相关参数（如 TLPX、THS-TRAIL 等）的范围说明。

步骤 2 利用高速示波器测量板端 MIPI 时序，并与步骤 1 查阅到的时序相比对，如果发现超过了范围，则可能是时序未匹配导致的屏幕未显示。

步骤 3 经由步骤 2，若发现 MIPI D-PHY 时序相关参数和屏幕的时序要求不能匹配，则需要修改 MIPITX 驱动。

例如：需要增加 THS-TRAIL 这个时序参数，则请找到 mipi_tx_hal.c 中 set_phy_reg(DATA0_THS_PREPARE, data_ths_prepare)、set_phy_reg(DATA1_THS_PREPARE, data_ths_prepare))、set_phy_reg(DATA2_THS_PREPARE, data_ths_prepare)、set_phy_reg(DATA3_THS_PREPARE, data_ths_prepare)、这几个位置，并增加配置的参数 data_ths_prepare。需要注意的是，增大该配置值 1，实际上增加的 THS-TRAIL 的时间为 8000/actual_phy_data_rate(单位：纳秒)，actual_phy_data_rate 的计算方法请查看 mipi_tx_hal.c。

关于其他更多 MIPITX D-PHY 的配置细节，请参考《Hi35xx xx Camera SoC 用户指南》中 MIPITX 章节。

