Panel Interconnection

User Guide

Issue 04

Date 2019-09-12

HiSilicon (Shanghai) Technologies Co., Ltd.

Address: New R&D Center, 49 Wuhe Road, Bantian,
Longgang District,
Shenzhen 518129 P. R. China

Website: http://www.hisilicon.com/en/

Email: support@hisilicon.com

About This Document

Purpose

This document describes how to debug the RGB LCD and MIPI LCD panels in the HiSilicon SoC solution to facilitate the development of RGB LCD and MIPI LCD services.

Note: In this document, an RGB LCD panel refers to an LCD panel with the RGB interface, that is, an RGB panel. An MIPI LCD panel refers to an LCD panel with the MIPI DSI as the interface, that is, an MIPI panel.

Related Version

The following table lists the product version related to this document.

Product Name	Version
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

Intended Audience

This document (guide) is intended for:

- Technical support engineers
- Software development engineers

Change History

Changes between document issues are cumulative. The latest document issue contains all changes made in previous issues.

Issue 04 (2019-09-12)

This issue is the fourth official release, which incorporates the following changes:

Sections 1.4.5, 2.2.1, and 2.4.12 are modified.

Issue 03 (2019-07-30)

This issue is the third official release, which incorporates the following changes:

In section 2.4.6 is modified.

Sections 2.4.11 and 2.4.14 are added.

Issue 02 (2019-06-20)

This issue is the second official release, which incorporates the following changes:

Sections 2.2.2, 2.2.4, 2.3.3, and 2.3.8 are modified.

Issue 01(2019-05-25)

This issue is the first official release, which incorporates the following changes:

Sections 2.2.4 and 2.3.3 are modified.

Issue 00B01(2019-04-28)

This issue is the first draft release.

Contents

Figures

Tables

1 Interconnection with an RGB Panel

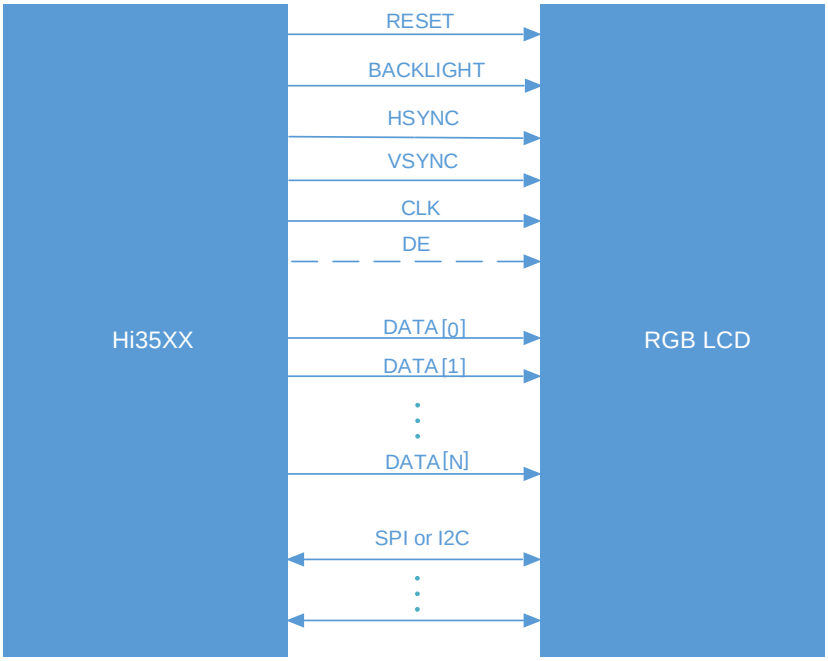
1.1 Environment Preparation

1.1.1 RGB Panel Interfaces

The board and the panel are connected using four types of lanes, which have different functions, as shown in Figure 1-1.

- Control command interface: SPI or I^2C
- Image data transmission interface: HSYNC, VSYNC, CLK, DATA 0–N, and DE
- Backlight control interface: BACKLIGHT
- Hardware reset interface: RESET

Figure 1-1 RGB panel interfaces



1.1.1.1 Control Command Interface

The control command interface can be used to modify or obtain the configuration of the panel registers.

Read the manuals of the RGB and MIPI LCD panels to know the type of the control command interface. Generally, it is an SPI or I^2C interface. The configuration is not required for some panels. Some panels even have no control command interface.

1.1.1.2 Data Transmission Interface

It is also called the RGB interface, through which the board sends image data to the LCD panel. The interface consists of signals such as HSYNC, VSYNC, CLK, DATA 0–*N*, and DE. Some panels do not have the DE signal. The RGB signals are transmitted in serial RGB or parallel RGB mode based on whether the RGB signals are sent by the DATA lane in time division multiplexing (TDM) mode. Specifically, the R, G, and B components are transmitted successively in serial mode while simultaneously in parallel mode.

The clock lanes HSYNC, VSYNC, and CLK ensure that the RGB data is transmitted from the SoC to the LCD panel at the correct timing.

NOTICE

The data enable (DE) mode is described as follows:

- In DE mode, Hi35xx notifies the RGB panel of the start and end of the vertical active lines (VACT) by using the DE lane, as shown in Figure 1-2.
- Therefore, when the DE mode of the RGB panel is enabled and the DE pin of the hardware is correctly connected to the panel, the horizontal blanking interval is not important to the panel.
- By default, the SDK configured with the DE mode. Therefore, when the DE mode of the LCD panel is enabled, no further configuration is required on Hi35xx. If the DE mode is not used on the RGB panel, no further configuration is required on Hi35xx either, because the RGB panel will ignore the DE signal automatically.

The RESET lane is used to reset a panel.

The DATA lane transfers RGB data. 6-bit, 8-bit, 16-bit, 18-bit, and 24-bit data transfer is supported.

1.1.1.3 Backlight Interface

It is used for adjusting the backlight brightness.

Generally, the output current is dynamically adjusted through the pulse width modulator (PWM) to control the backlight brightness of the panel. For details about how to set the PWM, see the "PWM" section in the *Hi35xx xx Camera SoC Data Sheet*. Alternatively, you may disable the dynamic PWM adjustment by using resistors, so that the backlight can only be turned on or off.

1.1.2 Connecting the Hardware

- Ensure that the hardware design complies with the *Hi35xx Hardware Design Data Sheet* and the connection is proper.

- Ensure that the pins are correctly configured in the hardware design by following the *Hi35xx LCD Output Description*. Otherwise, no data is output from the pins and the display is abnormal.

1.1.3 Enabling the Command Control Interface

When connecting to the RGB panel, check whether the nodes of the control bus I^2C or SPI is deployed and whether the clock is enabled.

Step 1 Check the deployment of the I^2C/SPI controller.

- Run the **ls /dev** command over the serial port to query the Linux deployment, as shown in Figure 1-2.

Figure 1-2 I^2C/SPI controller deployment example on Linux

```
/proc # ls /dev/
console             mtdblock5           tty32
cpu_dma_latency     mtdblock6           tty33
fb0                 mtdblock7           tty34
full                network_latency     tty35
gpiochip0           network_throughput  tty36
gpiochip1           null                tty37
gpiochip10          ptmx                tty38
gpiochip11          ram0                tty39
gpiochip2           ram1                tty4
gpiochip3           ram10               tty40
gpiochip4           ram11               tty41
gpiochip5           ram12               tty42
gpiochip6           ram13               tty43
gpiochip7           ram14               tty44
gpiochip8           ram15               tty45
gpiochip9           ram2                tty46
gsensor             ram3                tty47
hi_gpio             ram4                tty48
hi_lsadc            ram5                tty49
hi_tde              ram6                tty5
hi_userproc         ram7                tty50
i2c-2               ram8                tty51
i2c-3               ram9                tty52
i2c-7               random              tty53
ipcm                root                tty54
kmem                rtc0                tty55
```

As shown in Figure 1-2, I2C-2, I2C-3, and I2C-7 are deployed on Linux without the SPI device. Therefore, an exception occurs if an SPI device is used on Linux or the I^2C device is removed.

Run the **ls /dev** command over the serial port to query the Huawei LiteOS deployment, as shown in Figure 1-3.

Figure 1-3 SPI/I²C controller deployment on Huawei LiteOS

```
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
```

As shown in Figure 1-3, the I2C-0, I2C-1, and I2C-4, as well as SPI0 CS0, SPI1 CS0, SPI2 CS0, and SPI2 CS1 are deployed on Huawei LiteOS. Therefore, an exception occurs if the SPI or I²C device is not deployed on Huawei LiteOS.

NOTICE

If both ends share the same I²C/SPI controller, unexpected exceptions occur. Therefore, you are advised to deploy separate I²C/SPI controllers on both ends.

Step 2 Change the deployment of the I²C/SPI controller.

If the I²C/SPI controller deployment relationship does not meet service requirements, you need to change it as required. For example, if the panel driver is deployed on Linux or Huawei LiteOS but the required SPI2 is not deployed on Linux or Huawei LiteOS, you need to change the controller deployment relationship.

- For Linux, the following uses the SPI0 deployment as an example:
  - Open the **osdrv/opensource/kernel/linux-x.x.x/arch/arm/boot/drs/Hi35xx.dtsi** file and add **spi0=&spi_bus0** to the aliases structure.

Figure 1-4 Adding the SPI0 statement

```
#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;
    };
```

- In the definition of **spi_bus0**, change the value of **status** from **disabled** to **okey**.

Figure 1-5 Adding the SPI0 definition

```
        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";
        };
```

- Recompile and burn the kernel.

- For Huawei LiteOS, the following uses the deployment of SPI0 and I2C0 as an example:

- Add SPI0, open the **spi.h** file in the **osdrv/platform/liteos/platform/bsp/borad/Hi35xx/include/hisoc/** directory, and set **#define SPI0_ENABLE** to **1**, as shown in Figure 1-6.

Figure 1-6 Adding SPI0 on Huawei LiteOS

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

- Add I2C0, open the **borad.c** file in **osdrv/platform/liteos/platform/bsp/borad/Hi35xx/**, and add **#define ENABLE_I2C0** to the corresponding position, as shown in Figure 1-7.

Figure 1-7 Adding I2C0 on Huawei LiteOS

```
#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
```

Step 3 Configure I^2C/SPI pin multiplexing and clock gating.

- Multiplex the required pin as an I^2C/SPI pin. For details, see the *Hi35xx_PINOUT_EN*.
- Enable the I^2C/SPI clock by locating the corresponding register bit for I^2C/SPI clock gating. For details, see the "Clock" section in the *Hi35xx xx Camera SoC Data Sheet*.

----End

1.2 Interconnection Configuration

This section describes the software configuration required for panel interconnection.

Figure 1-8 shows the basic framework for the interconnection between a HiSilicon SoC and an RGB panel. You need to configure the data of the user app configuration layer and the panel driver.

Figure 1-8 Framework for the interconnection between a HiSilicon SoC and an RGB panel

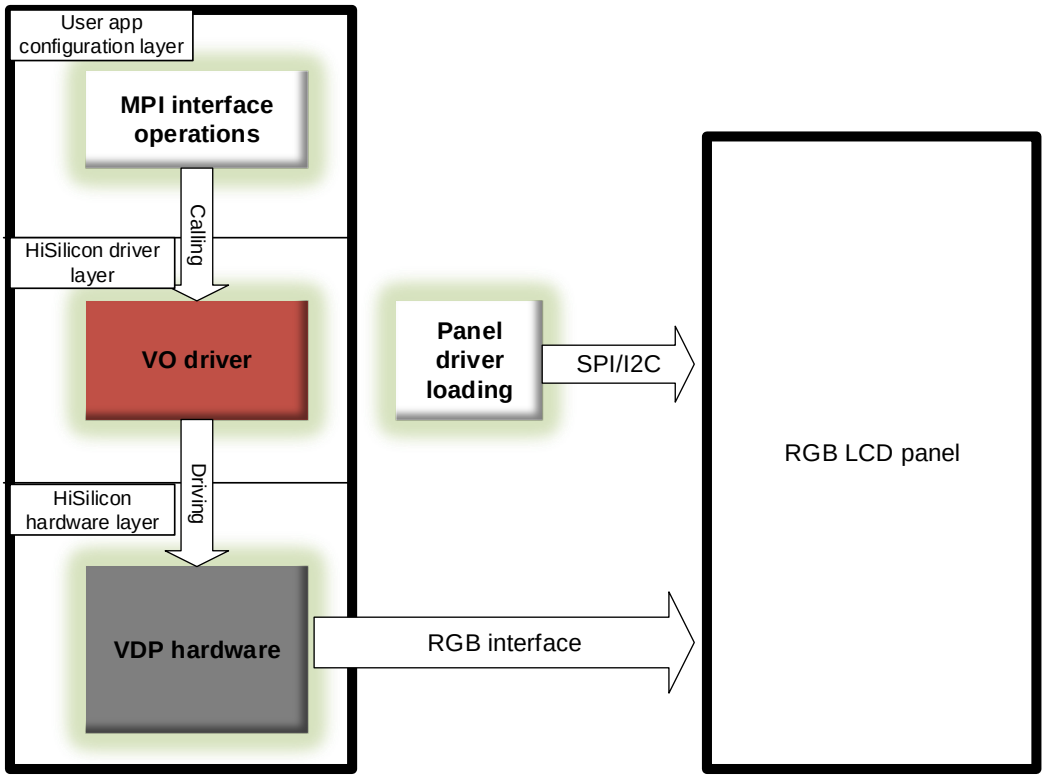
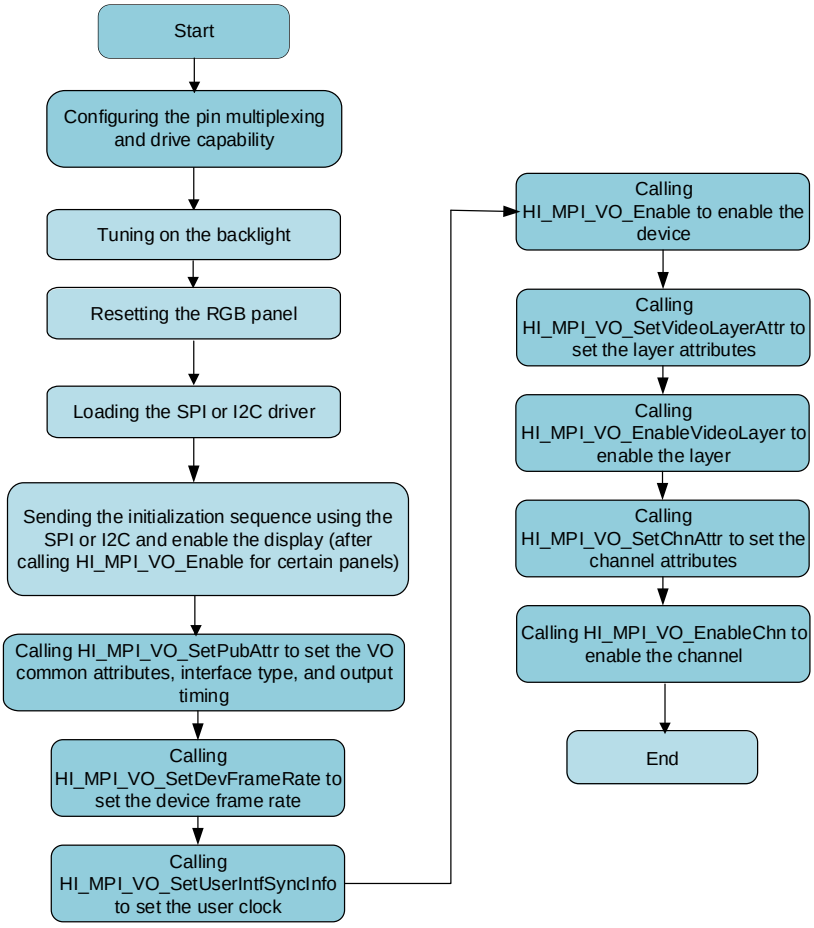


Figure 1-9 shows the configuration flowchart of an RGB panel.

Figure 1-9 RGB panel configuration flowchart



1.2.1 Configuring the Pin Multiplexing and Drive Capability

For an RGB panel, you can find the I^2C/SPI pins and the data interaction pins of the panel and the SoC by referring to the hardware schematic diagram. Then, configure the pin multiplexing and drive capability by setting the corresponding registers by referring to the *Hi35xx_PINOUT_EN*.

NOTICE

The pin drive capability depends on the peripherals connected to the pins. If the capability is set too high, a large overshoot may occur in the timing waveform and the pin may be damaged. If the capability is set too low, the waveform cannot meet the timing requirements. Therefore, an optimal setting is a balanced one.

1.2.2 Resetting the Panel

Generally, the GPIO port is used for resetting an RGB panel. Therefore, you need to configure the GPIO port and reset the panel.

Query the hardware schematic diagram to obtain the GPIO pin for reset. Set the pin to output (relative to the SoC).

For details about how to reset the panel, see the panel manual. If no reset operation is performed, the panel interconnection may fail.

1.2.3 Configuring the Backlight

Generally, the output current is dynamically adjusted through the PWM to control the backlight brightness. For details about how to set the PWM, see the "PWM" section in the *Hi35xx xx Camera SoC Data Sheet*.

Alternatively, you may disable the dynamic PWM adjustment by using resistors, so that the backlight can only be turned on or off.

1.2.4 Configuring the Initialization Sequence for the Panel

Panel initialization is generally required. You can initialize an RGB panel by sending data or commands through the I^2C or SPI.

The initialization sequence is provided by the panel vendor. Figure 1-10 shows an extract from the initialization sequence provided by an RGB panel vendor. The following information can be obtained:

This is a panel that communicates through the SPI. 8-bit, 9-bit, or 16-bit data is transferred over the SPI as configured.

The initialization sequences of the panels are similar. The differences lie in the data transfer mode and transfer value.

Figure 1-10 Initialization sequence of a panel

```
SPI_WriteComm(0x11);

Delay(120);          //Delay 120ms

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

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

The initialization sequence of a panel includes the pixel format, data refresh direction, and gamma configuration. For details about each instruction in the initialization sequence, see the manual of the panel driver IC.

1.2.5 Configuring the VO Output Timing

1.2.5.1 Configuring the User Timing

Use the user-timing interface to connect to a panel. For details, see chapter 4 "VO" in the *HiMPP V4.0 Media Processing Software Development Reference*. Configure the user-timing interface parameters in VO_PUB_ATTR_S as follows:

```
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;
```

- Confirm the user timing interface type.

  The value of **enIntfType** is determined by the panel itself. Table 1-1 lists the variables to be configured.

Table 1-1 VO_INTF_TYPE_E structure

Member	Description
VO_INTF_LCD_6BIT	RGB666 or RGB565 serial
VO_INTF_LCD_8BIT	RGB888 serial
VO_INTF_LCD_16BIT	RGB565 parallel
VO_INTF_LCD_18BIT	RGB666 parallel
VO_INTF_LCD_24BIT	RGB888 parallel

- Confirm the user timing type.

  enIntfSync indicates the timing type. Generally, set it to **VO_OUTPUT_USER**.

Table 1-2 VO_INTF_SYNC_E

Member	Description
VO_OUTPUT_USER	User timing mode

- Fill in the user timing information.

  Figure 1-11 shows the typical specifications of an RGB panel. Figure 1-12 shows the LCD pixel area at the VO user timing. The data structure is 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;
```

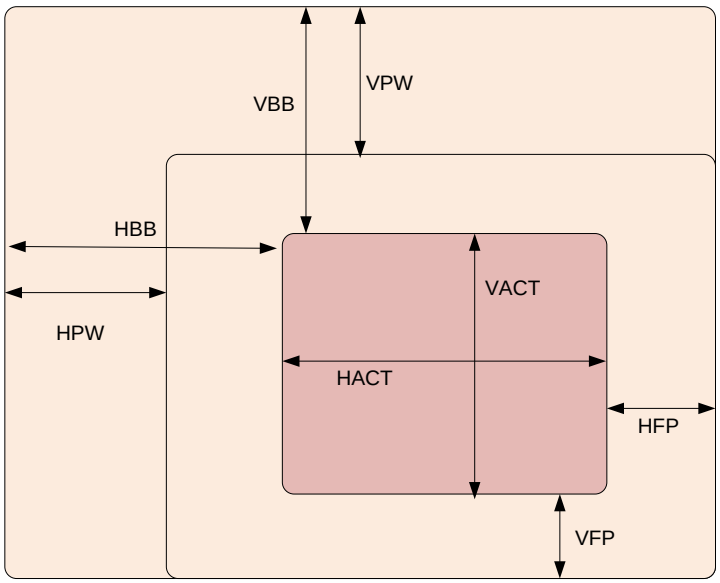


```
    HI_U16 u16Hfb;

    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;
```

Figure 1-11 Typical timing specifications of an RGB panel

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

Figure 1-12 Diagram of the LCD pixel area at the VO user timing



- **hpw + hbp** in Figure 1-11 corresponds to **HBB** in Figure 1-12.
- **hfp** in Figure 1-11 corresponds to **HFP** in Figure 1-12
- **vsa + vbp** in Figure 1-11 corresponds to **VBB** in Figure 1-12.
- **vfp** in Figure 1-11 corresponds to **VFP** in Figure 1-12.

Table 1-3 VO_SYNC_INFO_S structure

Member	Description
bSynm	Sync mode. The parameter is meaningless. Set it to **0**.
bIop	The value **0** indicates the interlaced mode, and **1** indicates the progressive mode. Set this parameter to **1** for MIPI and LCD panels.
u8Intfb	Bit width of the output interface. The parameter is meaningless. Set it to **0**.
u16Vact	Vertical active area (unit: line) For details, see the panel manual.
u16Vbb	Vertical back blank porch (unit: line) u16Vbb = VSA + VBP. For details, see the panel manual.
u16Vfb	Vertical front blank porch (unit: line) For details, see the panel manual.
u16Hact	Horizontal active area (unit: pixel) For details, see the panel manual.
u16Hbb	Horizontal back blank porch (unit: pixel) u16Hbb = HSA + HBP. For details, see the panel manual.
u16Hfb	Horizontal front blank porch (unit: pixel) For details, see the panel manual.
u16Hmid	Vertical sync active pixel value of the bottom field This parameter is meaningless in progressive mode. In progressive mode, set it to **1**.
u16Bvact	Vertical active area of the bottom field (unit: line) This parameter is valid in progressive mode, but is meaningless. In progressive mode, set it to **1**.
u16Bvbb	Vertical back blank porch of the bottom field (unit: line) This parameter is valid in progressive mode, but is meaningless. In progressive mode, set it to **1**.
u16Bvfb	Vertical front blank porch of the bottom field (unit: line) This parameter is valid in progressive mode, but is meaningless. In progressive mode, set it to **1**.
u16Hpw	Horizontal pulse width (unit: pixel)
u16Vpw	Vertical pulse width (unit: line)

Member	Description
bIdv	Polarity of the valid data signal. The value **0** indicates the signal is active high, and **1** indicates the signal is active low.
bIhs	Polarity of the horizontal valid data signal. The value **0** indicates the signal is active high, and **1** indicates the signal is active low.
bIvs	Polarity of the vertical valid data signal. The value **0** indicates the signal is active high, and **1** indicates the signal is active low.

1.2.6 Configuring the VO Output Clock

After configuring the user timing, you need to configure the VO clock frequency. The clock frequency for VO is the required clock frequency.

The frequency value can be automatically generated based on the timing and frame rate by using the *Clock Timing Calculator for RGB and MIPI Panels*. The output clock frequency of the VO interface is obtained by dividing the output clock by the frequency division coefficient. Therefore, to obtain the output clock of the interface, you need to determine the frequency division ratio and then confirm the clock frequency of the clock source. Then call HI_MPI_VO_SetUserIntfSyncInfo to configure the VO_USER_INTFSYNC_INFO_S structure to configure the clock. For details about the MPI, see chapter 4 "VO" in the *HiMPP V4.0 Media Processing Software Development Reference*.

NOTE

The *Clock Timing Calculator for RGB and MIPI Panels* does not support Hi3516E V200/Hi3516E V300/Hi3518E V300/Hi3516D V200.

User Timing Information

```
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;
```

Clock Source Attributes

```
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;
```

Clock Source Selection

```
typedef enum hiVO_CLK_SOURCE_E {
```

```
    VO_CLK_SOURCE_PLL,
    VO_CLK_SOURCE_LCDMCLK,
    VO_CLK_SOURCE_BUTT
} VO_CLK_SOURCE_E;
```

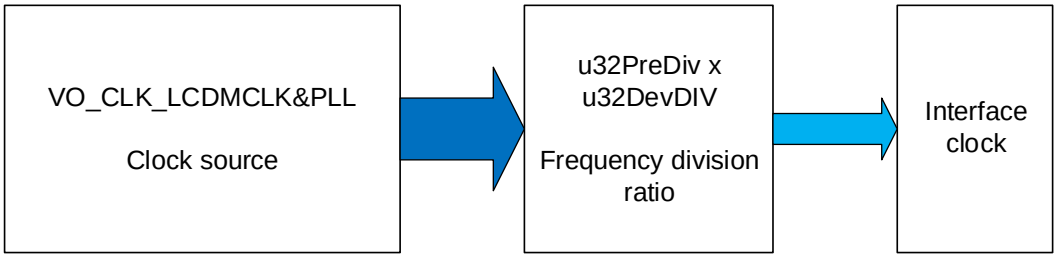
PLL Parameter Configuration

```
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 Frequency Division Ratio

The frequency division ratio is the product of **u32PreDiv** (pre-frequency division coefficient) and **u32DevDiv** (clock frequency division coefficient). For details about pre-frequency division and clock frequency division, see chapter 4 "VO" in the *HiMPP V4.0 Media Processing Software Development Reference*.

Figure 1-13 Relationship between the clock frequency of the clock source and the interface clock



The VO VDP CRG configuration of each SoC provides the configuration of the frequency division ratio.

For an LCD panel, constructing a pixel can take multiple clock cycles.

- For the RGB serial output, set the frequency division ratio according to the LCD panel manual.
  - If one pixel (R+G+B) is output every three clock cycles, set the frequency division ratio to **3**.
  - If one pixel (R+G+B+invalid data) is output every four clock cycles, set the frequency division ratio to **4**.
- For the RGB parallel output, set the frequency division ratio to **1**.

1.2.6.2 Clock Reverse

Whether the clock is reversely connected depends on the interconnected panel. During debugging, you can check the clock by using the oscilloscope and compare the check result with the RGB panel manual.

1.2.6.3 Frequency-Divided Clock

For details about how to calculate and configure the clock, see VO_USER_INTFSYNC_INFO_S in the *HiMPP V4.0 Media Processing Software Development Reference.*

Find the description about PLL selection and calculation in the SoC data sheet. You can also find the video display processor (VDP) clock control register among the CRG registers and check the clock source selected by the VDP.

NOTE
The following configuration about the frequency-divided clock does not apply to Hi3516E V200/Hi3516E V300/Hi3518E V300/Hi3516D V200. For details, see the SoC data sheet.

Take Hi3516C V500 as an example. Run the command **himd.l 0x12010108.**

The frequency of the LCD frequency-divided clock or PLL clock frequency of the VDP is calculated as follows:

Take a 320 x 240 panel at 60 fps as an example. Assume that the timing information of the panel is as follows:

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

According to the mapping relationship described in Table 1-3:

u16Hact = 320, u16Hbb = 65, u16Hfb = 7, u16Vact = 240, u16Vbb = 15, u16Vfb = 9

Calculate the clock frequency (unit: MHz) as follows:

X = (u16Hact + u16Hbb + u16Hfb) x (u16Vact + u16Vbb + u16Vfb) x Frame rate x u32PreDiv x u32DevDiv/10^6

= (320 + 65 + 7) x (240 + 14 + 9) x 60 x 1 x 1/10^6

= 6.209

LCD frequency division coefficient (u32LcdMClkDiv) = (X/1188) x (2^27). The maximum value of **X** is 75 MHz. Therefore, u32LcdMClkDiv = (6.209/1188) x (2^27) = 0xAB448.

For parameters **u32Fbdiv**, **u32Frac**, **u32Refdiv**, **u32Postdiv1**, and **u32Postdiv2** for the PLL clock frequency: X = 24 x [u32Fbdiv + u32Frac/(2^24)]/u32Refdiv/u32Postdiv1/u32Postdiv2

Therefore, **u32Fbdiv**, **u32Frac**, **u32Refdiv**, **u32Postdiv1**, and **u32Postdiv2** are **1**, **0x35935F**, **2**, **2**, and **1,** respectively.

1.3 Configuration Verification and Debugging

1.3.1 Checking the Pin Multiplexing and Drive Capability

Step 1 Check the pin multiplexing of the control command interface SPI or I^2C.

Step 2 Check the pin multiplexing of the data interface (LCD_DATA, HSYNC, VSYNC, and CLK). During preliminary debugging, you can configure a higher level for the drive capability. After the panel is turned on, optimize the drive capability to avoid component damage.

Refer to the *HI35XX_PINOUT_EN* and hardware design diagram, and obtain the pin multiplexing configuration by using the command **himd.l+*register address***.

Modify the multiplexing configuration as required in the code.

Take the LCD_CLK pin of Hi3559 V200 as an example. Find the address of the register controlling LCD_CLK in *Hi3559V200_PINOUT_EN*, as shown in Figure 1-14.

Figure 1-14 LCD_CLK multiplexing relationship

0x112F_0034	0x0400	31:11	Reserved
		10	Level conversion rate control. The value 0 indicates that the level conversion rate is high. The value 1 indicates that the level conversion rate is low.
		9	Pull-down resistor enable, active high
		8	Pull-up resistor enable, active high
		7:4	Drive capability. Values 0–7 correspond to I/O drive capability IO4_level 1–8, respectively.
		3:0	Function selection: 0: GPIO0_6 2: LCD_CLK 3: VOU_CLK

If the value of bit[3:0] in the 0x112F0034 register is not **0x2**, it indicates that the LCD_CLK pin multiplexing configuration of the LCD panel is incorrect and needs to be modified in the code.

----End

1.3.2 Checking the Reset Process

Step 1 Check whether the reset pin can be set high/low normally by using a multimeter or oscilloscope.

1. Refer to the hardware design diagram and find the GPIO port corresponding to the reset pin. Ensure that the pin is multiplexed as GPIO.
2. Set the GPIO direction to output by referring to the "GPIO" section in chapter 01-12 "Peripherals" in the SoC data sheet.

   The following takes GPIO5_2 of Hi3559 V200 as an example:

   The base address of GPIO5 is 0x120D5000.

Figure 1-15 Base addresses of the GPIO port

GPIO Controller	Base Address
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

Find the register with the offset address 0x400. Set its second bit to **1**.

Figure 1-16 GPIO direction control register

GPIO_DIR

GPIO_DIR is a GPIO direction control register. It is used to configure the direction of each GPIO pin.

Offset Address: 400 Total Reset Value: 0x00

Bits	Access	Name	Description	Reset
[7: 0]	RW	gpio_dir	GPIO direction control register. Bit[7: 0] correspond to GPIO_DATA [7: 0] respectively. Each bit can be controlled separately. 0: input 1: output	0x00

3. Refer to the "GPIO" section in chapter 01-12 "Peripherals" in the SoC data sheet, check whether the GPIO port can be set high/low normally by using an oscilloscope.

   Assume that GPIO5_2 of Hi3559 V200 is set high.

   Find the corresponding GPIO_DATA register 0x120D5000+0x1>>(2+2). For GPIO5_3, the address of the GPIO_DATA register is 0x120D5000+0x1>>(3+2).

   Set the second bit of this register to **0**, and then to **1**, which correspond to the low level and high level, respectively.

Step 2 Check whether the reset operation is consistent with the panel requirement defined in the panel manual. For example, you can set the reset pin high, low, and then high again, to check the delay.

----End

1.3.3 Checking the Backlight Control

There are two backlight control modes:

- Dynamically adjusting the backlight brightness

  In most cases, it is realized by adjusting the duty cycle using the PWM. The PWM driver needs to be implemented.

- Setting the backlight brightness by using the hardware (dynamic adjustment not supported)

  During the debugging, set the backlight brightness to the maximum.

1.3.4 Checking the Control Command Path

Check whether a control command can be successfully issued, for example:

- Generally, the SPI or I^2C interface is used for control command issuing for a panel. Observe the waveform when the command is sent by using an oscilloscope, check the sent data according to the protocol, and check whether the actual timing complies with that specified in the panel manual.
- Read the panel registers to check whether the values read meet the expectation. Particularly, you can set a register to a specific value, read the register, and check whether the value read meets the expectation. Note that some registers are read-only or write-only. For details, see the manual of the panel driver IC.
- Send some specific sequences to enable the self-test function of the panel, for example, the color bar factory test program.

1.3.5 Checking the VO Output Clock and Timing

1.3.5.1 VO Output Clock

You can use an oscilloscope to measure the frequency of the vo_clk pin. The frequency obtained by using the oscilloscope must be the same as the frequency obtained by using either of the following two methods.

- Check whether the frame rate in the proc information is as expected:

  (u16Hact + u16Hbb + u16Hfb) x (u16Vact + u16Vbb + u16Vfb) x Frame rate = VO output clock frequency

  During the running of a service program, run the **cat /proc/umap/vo** command on the serial port terminal to view the proc information about the VO module.

  In the proc information of the VO module, obtain the configured clock source and clock source parameters, as well as the interrupt of the VDP logic, that is, the actual frame rate of the device (**IntRate** in Figure 1-17).

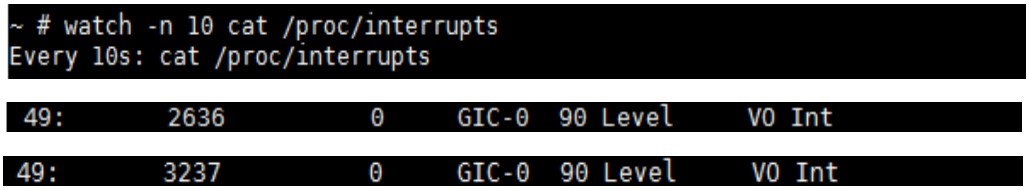
Figure 1-17 VO proc information

```
~ # 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
```


- Run the **watch –n 10 cat /proc/interrupts** command during service running to obtain the number of interrupts generated by the VO module every 10 seconds. Then, calculate the actual frame rate.

Figure 1-18 VO interrupt information



According to Figure 1-18: (3237 – 2636)/10 = 60

1.3.5.2 Checking the Timing

Use an oscilloscope to observe HSYNC, VSYNC, and CLK. Based on the RGB timing protocol, you can obtain the actual timing values, that is, HPW, HFP, and HACT. In this process, you only need to pay attention to whether waveform conversion is performed in the time sequence, check the VO timing parameters.

1.3.6 VO Debugging Using Color Bars

Color bar patterns are widely used for calibrating the timing and color reproduction errors.

VDP configuration errors can always be reflected on the color bars. For instance, color deviation of the color bars could be caused by incorrect CSC settings of the VDP. If the color bars are displayed in a wrong area, the clock timing may be incorrect. If the color bars are not fully displayed, check the setting of RGB panel.

Using color bar patterns can help to locate faults. You need to analyze the faults according to the actual situation.

General debugging procedure: Enable the VDP device color bars by setting the VDP register DHD0_CTRL or DHD1_CTRL (Refer to the SoC data sheet for the accurate register name.) Enable bit **cbar_en**. Set the color space select bit **cbar_sel** to VGA. Set the color bar mode bit **cbar_mode** to horizontal or vertical. Set the update bit **regup** of the DHD0 register to **1**.

Take Hi3516C V500 as an example:

- Check the register: **himd.l 0x1144d000**
- Enable the color bars: **himm 0x1144d000 0xc0000011**
- Disable the color bars: **himm 0x1144d000 0x80000011**

Where, **0x11440000** is the base address of the VDP register, and **0xd000** is the offset address of the DHD0_CTRL register.

Figure 1-19 Horizontal VDP color bars

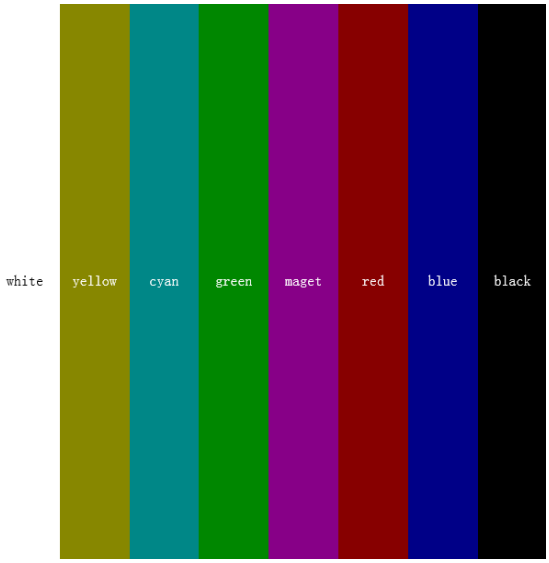
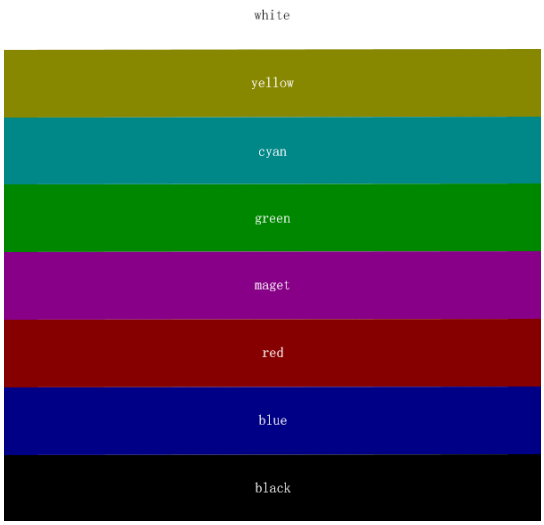


Figure 1-20 Vertical VDP color bars



1.4 FAQs for LCD Debugging

1.4.1 Black Screen Without Response

[Symptom]

The screen goes complete black and does not respond.

[Cause Analysis]

There are many possible causes, for example, a damaged component, a connection failure, a power supply failure, incorrect pin multiplexing configuration, an incorrect status of the reset pin, and incorrect VO timing configuration. You need to check them one by one.

[Solution]

Step 1 Check the hardware.

Contact hardware engineers to check whether the power supply and connection of the panel are normal and whether the components are intact.

Step 2 Check the software process.

Compare it with the timing diagram in the RGB panel configuration in section 1.2 "Interconnection Configuration." Note that the operation on the reset pin should strictly follow the time requirements. Generally, a control command can be received in 120 ms after the reset.

Step 3 Check the pin multiplexing and drive capability. For details, see section 1.2.1 "Configuring the Pin Multiplexing and Drive Capability."

Step 4 Check the control command path. For details, see section 1.3.4 "Checking the Control Command Path."

Step 5 Check the backlight configuration and ensure that the backlight is enabled. For details, see section 1.2.3 "Configuring the Backlight."

Step 6 Check the VO output clock and output timing.

For details, see sections 1.2.5 "Configuring the VO Output Timing" and 1.3.5 "Checking the VO Output Clock and Timing."

----End

1.4.2 Incorrect Position of Display

[Symptom]

Only a part of the image is displayed.

[Cause Analysis]

Only a part of an image is displayed on the screen, and the start point of the displayed image is not the initial point of the screen.

[Solution]

This problem is usually caused by the mismatch between the configuration of the blanking interval of the line or field of the SoC and that of the panel. In this case, you need to modify the configurations on both sides to be consistent. You can modify the configuration on either the SoC or panel side.

1.4.3 Incorrect Display Start

[Symptom]

The start of the image is on the center of the screen.

[Cause Analysis]

Changing the back and front porch settings makes no difference. The possible cause is that the output pixel clock and the timing do not match.

[Solution]

Check the clock by referring to section 1.3.5 "Checking the VO Output Clock and Timing."

1.4.4 Incorrect Display Size

[Symptom]

The actual size of the displayed image does not match the screen.

[Cause Analysis]

In this case, the actual size of the displayed image exceeds the screen size (that is, the image is displayed incompletely) or is smaller than the screen size.

[Solution]

This problem is usually caused by the mismatch between the configuration of the active region of the line or field on the SoC side and that of the panel. In this case, you need to modify the configurations on both sides to be consistent. You can modify the configuration on either SoC or panel side..

1.4.5 Abnormal Colors

The colors of the image are obviously different from what are expected. This problem has multiple symptoms.

Image Display out of Order and Illegible Target Contour

[Symptom]

The image is completely illegible.

[Cause Analysis]

The possible cause is that the interface types used on the panel and SoC are inconsistent. For example, the BT.656 interface is used on the SoC while the RGB interface is selected on the panel.

[Solution]

Check whether the configured VDP interface type is the same as the selected interface type.

Color Changes

[Symptom]

The red, blue, and green colors on the RGB LCD are out of order.

[Cause Analysis]

The RGB TX sequences on the SoC and panel do not match.

[Solution]

For serial RGB:

- Ensure that the sequence of sending RGB signals is the same as the RGB receiving sequence. Some panels support the adjustment of the RGB RX sequence, which means that you can directly modify the register configuration.
- The clock output polarity may be incorrect. In this case, you can modify the **bClkReverse** member of the VO_USER_INTFSYNC_INFO_S structure by calling the HI_MPI_VO_SetUserIntfSyncInfo MPI.
- The serial RGB output of HiSilicon can be implemented in the following two modes: frequency division by 3 or 4.
  - Three or four pixel clock cycles are used as the timing configuration step. For example, increasing the sync width by 1 actually increases the sync signal by three or four pixel clock cycles.
  - If the configured timing of the blanking interval or sync signal on the panel side is not a multiple of three or four pixel clock cycles, the RGB reception may be misplaced. For example, the RGB signal may be parsed as GBR. In this case, correct the timing configuration of the blanking interval or sync signal on the panel.

For parallel RGB:

- Check whether the wire sequence of the hardware is correct and whether errors occur in the RGB sequence.
- Check whether the current software configuration ensures that the RGB sequence on the SoC side is consistent with the RGB RX sequence on the screen.

If inconsistency occurs, you can adjust the sequence on the SoC side or on the screen (some screens support adjustment). For details about how to adjust the RGB sequence configuration on the SoC side, see the VDP section in *Hi35xx xx Camera SoC User Guide* to query relevant register details.

Abnormal Color Display

[Symptom]

The displayed colors are abnormal, and the color conversion mode is incorrect.

[Cause Analysis]

The color space conversion (CSC) of the video layer may not be correctly configured. The output has to be converted to the RGB format by using the CSC function for the LCD or MIPI screen.

[Solution]

Call the HI_MPI_VO_SetVideoLayerCSC or HI_MPI_VO_SetGraphicLayerCSC function to set CSC attributes. For details about how to set the parameters, see the related interface operations in 4 "VO" in the *HiMPP V4.0 Media Processing Software Development Reference*.

Color Deviation in the Displayed Lines

[Symptom]

There is a color deviation around the contour in the image.

[Cause Analysis]

The line color is abnormal, especially the contour line of the image, which is affected by adjacent pixels. This problem is usually caused by a minor error occurred in the timing. As a result, the information about the pixel and the information about the adjacent pixels are

transmitted incorrectly during data transmission. There are many possible causes of this situation.

[Solution]

- If the hardware interference is high, use an oscilloscope to check whether the waveform meets the expectation. If not, the possible cause is that the line is too long (which generally only occurs in the jump wire), or the pin drive capability is insufficient.
- The clk phase of the video output is incorrect and does not match that of the panel. You need to correct timing configured for the SoC and panel.

1.4.6 Unsatisfactory Display Effect

[Symptom]

Performance in aspects such as the brightness, chroma, and contrast are unsatisfactory.

[Cause Analysis]

The display effect of the screen is not satisfactory.

[Solution]

To adjust the display effect of the screen, you need to:

- Adjust the output picture effect on the SoC.
- Adjust the display effect parameters on the screen.

The display effect of the SoC can be adjusted by modifying parameters such as VO parameters, luminance, and contrast. The display effect of the panel can be adjusted by modifying the gamma curve, luminance, and chroma. Generally, the panel vendor provides recommended configurations, which meet the requirements in most scenarios.

1.4.7 Frame Residue During Image Conversion

[Symptom]

Frame residue occurs during image conversion.

[Cause Analysis]

It is usually caused by improper Vcom voltage configuration or abnormal voltage of some power pins.

[Solution]

For details, see the description in the panel manual or seek solutions from the panel vendor.

1.4.8 Low Frame Rate or Freezing Frame

[Symptom]

Frame freezing occurs during the display.

[Cause Analysis]

Generally, the frequency of the data clock is insufficient.

[Solution]

Adjust the clock frequency.

1.4.9 Dark Image with Normal Backlight

[Symptom]

The image is dark but the backlight configuration is correct.

[Cause Analysis]

Some data lines of the RGB screen have no data. The pins are not correctly configured during the hardware design by following the *Hi35xx* As a result, there is not data in some of the pins between the screen and the SoC.

[Solution]

There are many causes for a dark image, but you can take the priority to rectify the fault as described.

2 Interconnection with an MIPI Panel

NOTE
Hi3516E V200/Hi3516E V300/Hi3518E V300/Hi3516D V200 does not support the MIPI TX.

2.1 Environment Preparation

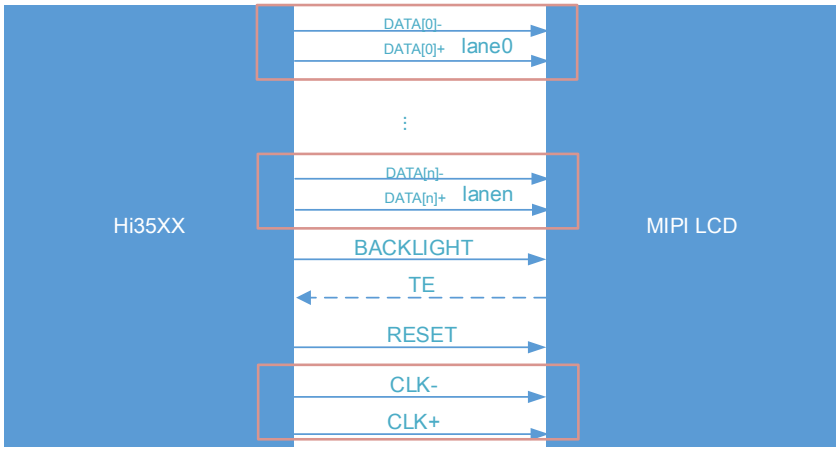
2.1.1 MIPI DSI

Commonly used in the smartphone market, this interface features lower power consumption, higher data transmission rate, and smaller PCB space usage. Some of the Hi35xx SoCs support this type of interface.

Typically, an MIPI DSI panel provides the following signal lanes, as shown in Figure 2-1.

- MIPI clock lane: CLK
- MIPI data lanes (≤ 4): DATA
- Signal synchronization lane: TE. Typically, only a panel of the Command mode has this pin for frame synchronization. It is used when the HiSilicon MIPI TX is in slave mode.
- Backlight control signal: BACKLIGHT
- Reset pin: RESET

Figure 2-1 MIPI DSI connection diagram



For details about the MIPI TX interface and how to configure the SDK, see section 9.4 "MIPI TX" in chapter 9 "Video Interfaces" in the *Hi35xx Mobile Camera SoC Data Sheet*, and the *MIPI User Guide*.

2.1.2 Connecting the Hardware

- Ensure that the hardware design complies with the *Hi35xx Hardware Design Data Sheet* and the connection is proper.
- Ensure that the pins are correctly configured in the hardware design by following the *Hi35xx LCD Output Description*. Otherwise, no data is output from the pins and the display is abnormal.

2.2 Interconnection Configuration

This section describes how to configure the software for panel interconnection.

Figure 2-2 shows the framework for the interconnection between a HiSilicon SoC and an MIPI panel.

Figure 2-2 Framework for the interconnection between a HiSilicon SoC and an MIPI panel

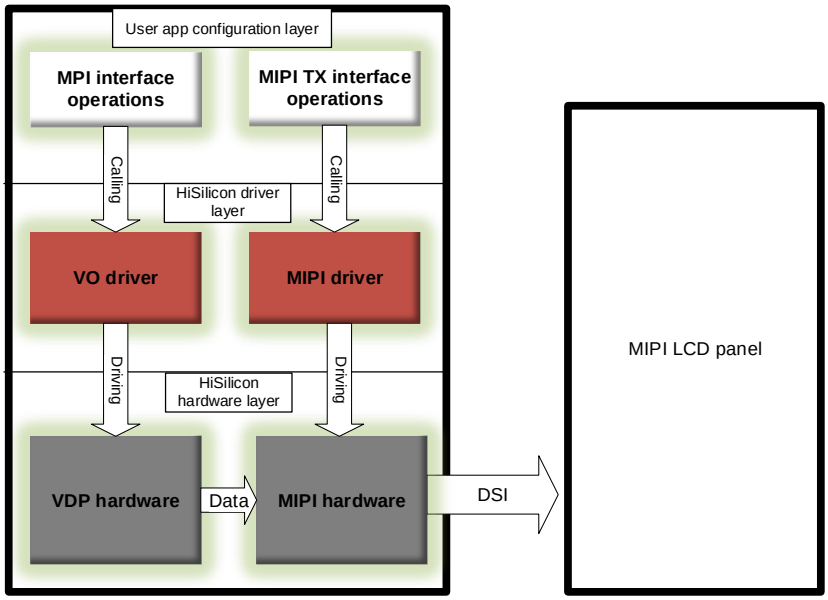
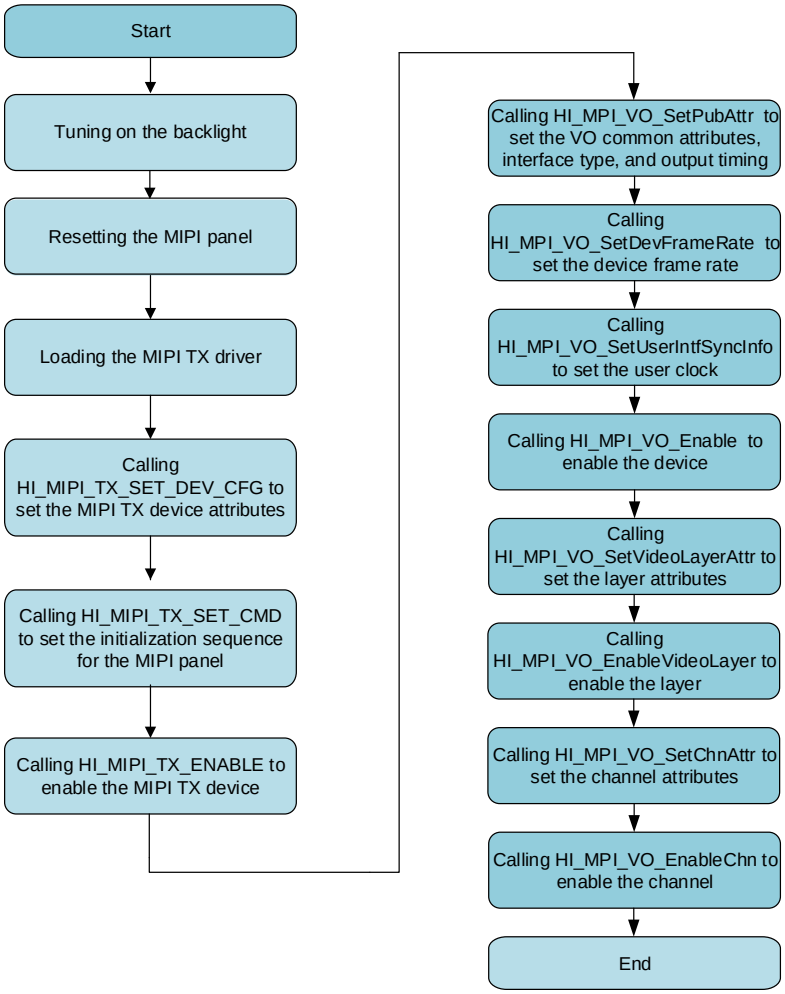


Figure 2-3 MIPI panel configuration flowchart



2.2.1 Configuring the Pin Multiplexing and Drive Capability

Read the board hardware design diagram and refer to the description in 2.1.1 "MIPI DSI" to find the display driver IC and the control pins and data pins that interact with the SoC, and record their pin numbers.

In the *Hi35xx_PINOUT_EN*, find the configuration registers of the preceding pins and record the registers for configuring the pin multiplexing.

NOTICE

- The pin drive capability depends on the peripherals connected to the pins. If the capability is set too high, a large overshoot may occur in the timing waveform and the pin may be damaged. If the capability is set too low, the waveform cannot meet the timing requirements. Therefore, an optimal setting is a balanced one.
- For Hi3519A V100 and Hi3556A V100, if the MIPI DSI function is required, ensure that the **mipitxphy_cmos_mode_enable** bit of the MISC_CTRL1 register is set to **0**. For details about this register, see the *Hi35xx Vxx Camera SoC Data Sheet*.

2.2.2 Resetting the Panel

Generally, the GPIO port is used for resetting an MIPI panel. Therefore, you need to configure the GPIO port and reset the panel.

- Query the hardware schematic diagram to obtain the GPIO pin for reset.
- Set the direction of the pin to output.
- Set the reset operation timing of the GPIO pin used for reset.

For details about how to reset the panel, see the panel manual. If no reset operation is performed, the reset timing does not match the panel requirement, or the level does not match, the panel may fail to be lit or work abnormally.

2.2.3 Configuring the Backlight

Generally, the output current is dynamically adjusted through the PWM to control the backlight brightness. For details about how to set the PWM, see the "PWM" section in the *Hi35xx xx Camera SoC Data Sheet*.

Alternatively, you may disable the dynamic PWM adjustment by using resistors, so that the backlight can only be turned on or off.

2.2.4 Configuring the Panel

The configuration of an MIPI panel consists of two parts:

- MIPI device attributes
- MIPI panel initialization sequence

The procedure for initializing an MIPI panel is as follows:

Step 1 Configure the attributes of the MIPI TX device.

The SDK provides the combo_dev_cfg_t structure. Call HI_MIPI_TX_SET_DEV_CFG and assign a value to each member in the structure.

The combo_dev_cfg_t structure is defined as follows:

```
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
```

The sync_info_t structure is defined as follows:

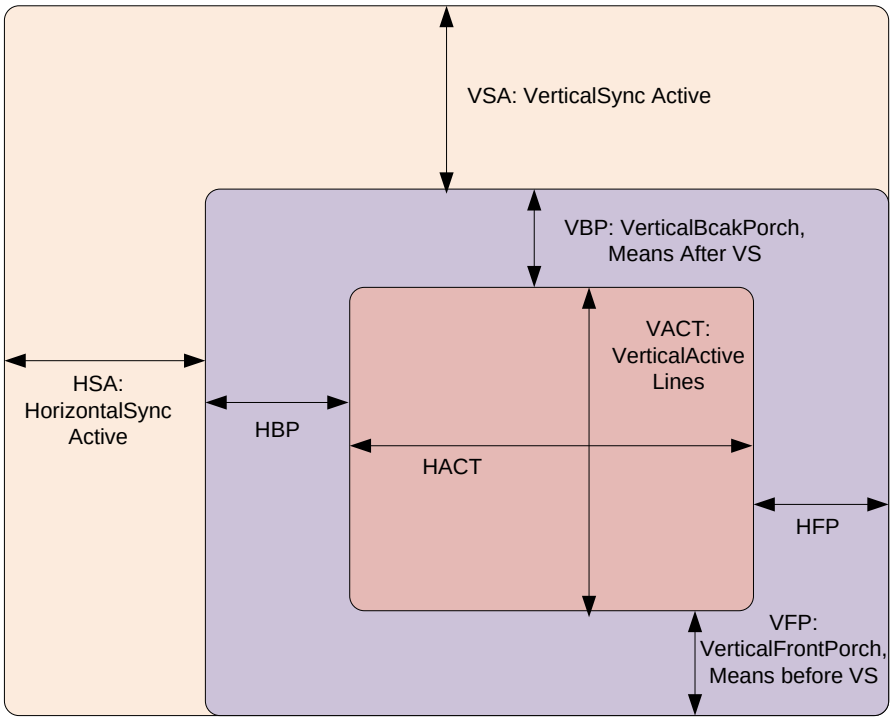
```
typedef struct
{
    unsigned short  vid_pkt_size;
    unsigned short  vid_hsa_pixels;
    unsigned short  vid_hbp_pixels;
    unsigned short  vid_hline_pixels;
    unsigned short  vid_vsa_lines;
    unsigned short  vid_vbp_lines;
    unsigned short  vid_vfp_lines;
    unsigned short  vid_active_lines;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

The following describes how to configure sync_info_t (synchronization information of the MIPI TX device) in combo_dev_cfg_t, as shown in Figure 2-4.

- The pink area indicates the active display area of the MIPI panel. The two double-headed arrows indicate the height and width of the MIPI panel.
- The two double-headed arrows in the light yellow area indicate the number of pixels in the horizontal sync area and the number of lines in the vertical sync area, respectively.
- The two double-headed arrows in the purple area indicate the number of pixels in the horizontal sync front and back blank porches and the number of lines in the vertical sync front and back blank porches, respectively.

For details about these parameters, consult the panel vendor or query the panel manual.

Figure 2-4 Diagram of the MIPI pixel areas under the MIPI DSI protocol



Fill in the member values of combo_dev_cfg_t according to the panel manual, as described in Table 2-1.

Table 2-1 combo_dev_cfg_t structure

Member	Description
devno	MIPI TX device ID. Set this parameter to 0.
lane_id	For 4 lanes, set this parameter to {0,1,2,3}. For 3 lanes, set this parameter to {0,1,2,–1}. For 2 lanes, set this parameter to {0,1,–1,–1}. For 1 lane, set this parameter to {0,–1,–1,–1}. **–1** indicates an unused lane.
output_mode_t	MIPI TX output mode Value: **OUTPUT_MODE_CSI**, **OUTPUT_MODE_DSI_VIDEO**, or **OUTPUT_MODE_DSI_CMD** Assign the value based on the specific panel requirements. For details, see the panel manual.

Member	Description
video_mode_t	Data type in **VIDEO** mode Value: **BURST_MODE**, **NON_BURST_MODE_SYNC_PULSES**, or **NON_BURST_MODE_SYNC_EVENTS** The sequence of the transmitted data and the location of the data packet differ in the three modes. Assign the value based on the specific panel requirements. For details, see the panel manual. If **output_mode_t** is set to **OUTPUT_MODE_DSI_CMD**, this attribute is invalid.
output_format_t	MIPI TX output data format Value: **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**, or **OUT_FORMAT_YUV422_8_BIT** This parameter indicates the format of the data in the data packet sent by the MIPI TX. Assign the value based on the specific panel requirements. For details, see the panel manual.
vid_pkt_size	Size of the received packet (unit: pixel): vid_pkt_size = HACT
vid_hsa_pixels	Number of pixels in the horizontal sync area Assign the value based on the specific panel requirements. The value must be the same as the horizontal pulse width (**u16Hpw**) of the VO timing.
vid_hbp_pixels	Number of pixels in the horizontal sync back porch
vid_hline_pixels	Number of pixels in each line vid_hline_pixels = HACT + HSA + HBP + HFP
vid_vsa_lines	Number of lines in the vertical sync area Assign the value based on the specific panel requirements. The value must be the same as the vertical pulse width (**u16Vpw**) of the VO timing.
vid_vbp_lines	Number of lines in the vertical sync back porch
vid_vfp_lines	Number of lines in the vertical sync front porch
vid_active_lines	vid_active_lines = VACT (unit: line)
epdi_cmd_size	Count of bytes in the write memory command This value is invalid in **VIDEO** mode. Set this value to **HACT** in **CMD** mode.
phy_data_rate	Data input rate (unit: Mbit/s) phy_data_rate ≥ (HACT + HAS + HBP + HFP) x (VACT + VSA + VBP + VFP) x Output format bits x Frame rate/lane_num/(10^6)

Member	Description
pixel_clk	Pixel clock (unit: kHz) pixel_clk = (HACT + HSA + HBP + HFP) x (VACT + VSA + VBP + VFP) x Frame rate/1000 The value is rounded up.

NOTICE

- Some panels may fail to be turned on at the **phy_data_rate** rate calculated by using the formula given in Table 2-1. In this case, read bit[24] of the MIPI TX register INT_ST1 (interrupt status 1 register). If the value is **1**, **phy_data_rate** is too small and needs to be increased to a proper value.
- Find the address of the INT_ST1 register (interrupt status 1 register) in the "MIPI TX" section in chapter 9 "Video Interfaces" in the *Hi35xx xx Camera Soc Data Sheet.*

Example:

Assume that the panel vendor has provided the following information:

- MIPI TX output mode: **DSI_VIDEO**
- Data type: **BURST_MODE**
- Output data format: **OUT_FORMAT_RGB_24_BIT**

Timing information:

- Horizontal active area (HACT): **1080** (unit: pixel)
- Horizontal back porch (HBP): **20** (unit: pixel)
- Horizontal front porch (HFP): **130** (unit: pixel)
- Horizontal sync timing (HSA): **8** (unit: pixel)
- Vertical active lines (VACT): **1920** (unit: line)
- Vertical back porch (VBP): **26** (unit: line)
- Vertical front porch (VFP): **16** (unit: line)
- Vertical sync active (VSA): **10** (unit: line)
- Device frame rate: **60** (fps)

Therefore, the attributes of the MIPI TX device should be configured as follows:

```
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,
 };
```

If the calculated values of **phy_data_rate** and **pixel_clk** are decimals, the values can be rounded up.

Step 2 Configure the initialization sequence of the MIPI TX panel.

Panel initialization is required. An RGB panel uses the I^2C or SPI to send data or commands to initialize the panel. An MIPI panel uses the MIPI TX PHY interface to send data packets of a specified type.

The initialization sequence is provided by the panel vendor.

The initialization sequence of a panel includes the pixel format, data refresh direction, and gamma configuration. For details about each instruction in the initialization sequence, see the manual of the MIPI panel.

The SDK provides the cmd_info_t structure. You can call HI_MIPI_TX_SET_CMD and assign a value to each member in the structure. It is recommended that this MPI be called before HI_MIPI_TX_ENABLE. In this case, the MIPI works in LP mode, data is transmitted at a low speed, and the compatibility is good. It is also recommended that HI_MIPI_TX_GET_CMD be called before HI_MIPI_TX_ENABLE.

The md_info_t structure is defined as follows:

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

Generally, a MIPI command is in the following format: Data type + Data address + Data index + Data 1 + Data 2+…+ Data *N*. The data index indicates the number of data records. The data address or data generally consists of 16 bits.

The initialization sequence provided by the panel vendor generally has a register address and corresponding data. Therefore, a data type and a data index need to be filled according to a sequence provided by a panel vendor.

For details about cmd_info_t, see Table 2-2.

Table 2-2 Value assignment of the cmd_info_t structure

Member	Description
devno	MIPI TX device ID. Set this parameter to **0**.
data_type	Write command data type, that is, Data Type in the Display Command Set (DCS). Select a data type based on the number of data records. Type 1: If only the register address is sent with no data, set this parameter to **0x5** or **0x13**. Generally, both values take the same effect. For details, contact the panel vendor or query the panel data sheet. Type 2: If both the register address and data (one data record) are sent, set this parameter to **0x15** or **0x23**. Generally, both values take the same effect. For details, consult the device vendor. Type 3: If both the register address and data (≥ 2 data records) are sent, set this parameter to **0x39** or **0x29**. They are equivalent in most cases. For details, contact the panel vendor.
cmd_size	Number of data types If there are two or more data types, this parameter is used to indicate the number of data records. Type 1: If there is no data, this parameter indicates the register address. Type 2: If there is one data record, the lower eight bits indicate the address, and the upper eight bits indicate the data.
cmd	Pointer to the command data Types 1 and 2: If there are less than two data records, this parameter can be set to null. Type 3: If there are two or more data records, this parameter indicates the data.

For details about the configuration of the command data type parameter **data_type**, consult the device vendor. Unless otherwise specified by the vendor, you are advised to set **data_type** to **0x5** when no data is available or to **0x15** when only one data record is available. Set it to **0x39** when there are multiple data records.

The following describes the example codes of the cmd_info_t structure with different command data types:

- Type 1: The MIPI TX sends only an address.

  Assume that the address **0x11** is to be sent, the configuration is as follows:

```
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;
}
```

- Type 2: The MIPI TX sends an address and one data record.

  Assume that data **0x48** is to be written to the address **0x36**, the configuration is as follows:

```
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;
}
```

- Type 3: The MIPI TX sends an address and multiple data records.

  Assume that data records **0xf0**, **0x03**, ... , **0x23** are to be written to the address **0xe1** address in sequence, the configuration is as follows:

```
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;
}
```


Step 3 Read the MIPI panel initialization sequence. This step is used to debug the obtained configuration, such as the panel vendor information. This step can be skipped when only panel turning-on is required. It is recommended that the operation be performed before HI_MIPI_TX_ENABLE is called.

The sync_info_t structure is defined as follows:

```
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;
```

Member	Description
devno	MIPI TX device ID. Set this parameter to **0**.
data_type	Data Type in the Display Command Set (DCS). Generally, the data type of a read command is **0x06**, **0x04**, **0x14**, or **0x24**. For details, contact the panel vendor or query the panel data sheet.
data_param	Data parameter. The lower eight bits define the first parameter and the upper eight bits define the second parameter. Set a bit to **0** if the bit is not used.
get_data_size	Length of the read data
*get_data	Pointer to the obtained data storage address. The address needs to be allocated by the user.

To read a byte of **0xda**, the configuration is as follows:

```
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;
}
```

To read multiple bytes of **0xa1**, the configuration is as follows:

```
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;
}
```


NOTICE

When multiple bytes are read, the data type 0x14 or 0x24 is determined by the panel.

----End

2.2.5 Configuring the VO Output Timing

When connecting to a panel, you can set the information such as the user timing by calling MPI HI_MPI_VO_SetPubAttr. For details about the MPI, see chapter 4 "VO" in the *HiMPP V4.0 Media Processing Software Development Reference*. Configure the user-timing interface parameters in VO_PUB_ATTR_S as follows:

```
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;
```

In VO_PUB_ATTR, **enIntfType** indicates the interface type.

- Confirm the user timing interface type.

  The value of **enIntfType** is determined by the panel itself. Table 2-3 lists the variables to be configured.

Table 2-3 VO_INTF_TYPE_E structure

Member	Description
VO_INTF_MIPI	MIPI output
VO_INTF_MIPI_SLAVE	MIPI_SLVAE output, that is, the panel actively sends sync information to the SoC through the TE signal pin. It applies to an MIPI panel in Command mode. Currently, only Hi3519A and Hi3556A support this function.

- Confirm the user timing type.

  enIntfSync indicates the timing type. Generally, set it to **VO_OUTPUT_USER**.

Table 2-4 VO_INTF_SYNC_E

Member	Description
VO_OUTPUT_USER	User timing mode

- Fill in the user timing information.

  Figure 2-5 shows the pixel area of a typical MIPI panel. For the VO, the LCD pixel area corresponding to the user timing is shown in Figure 2-6. The user timing structure VO_SYNC_INFO_S is defined as follows:

```
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;
```


Figure 2-5 Diagram of the MIPI pixel areas under the MIPI DSI protocol

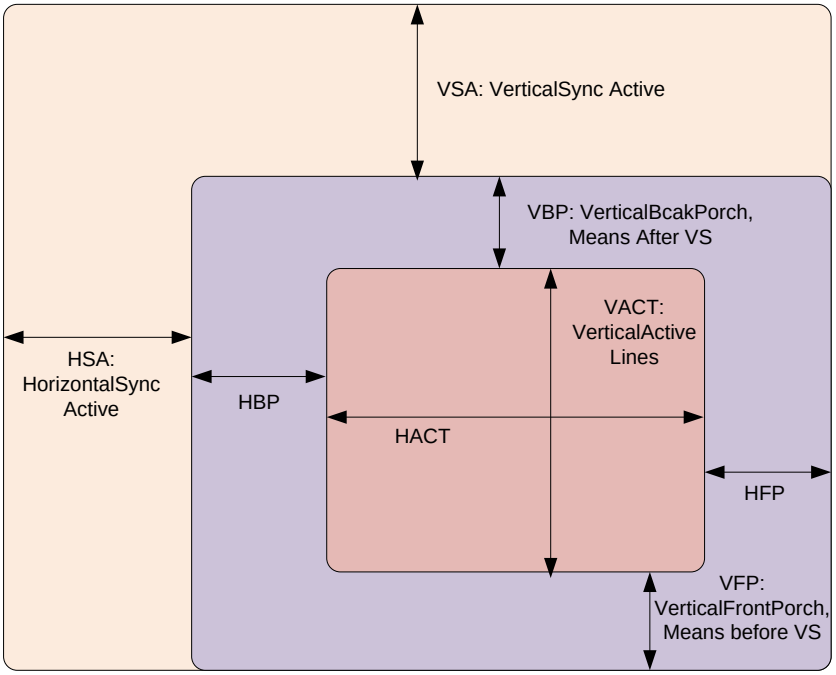
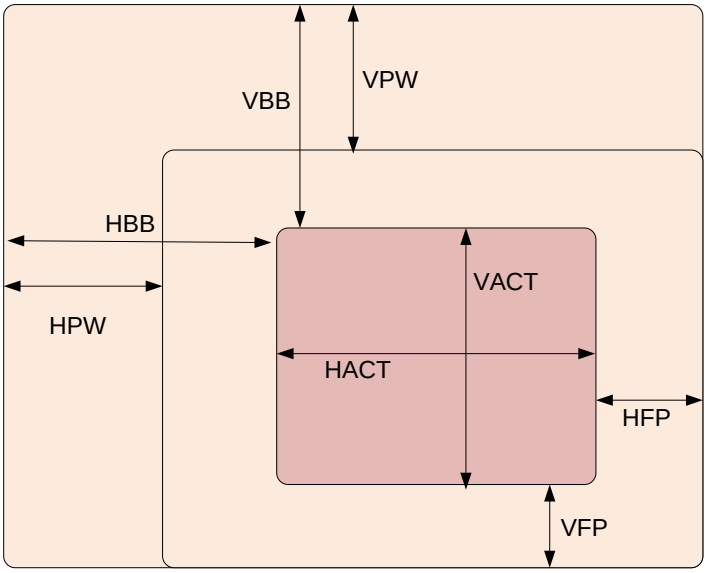


Figure 2-6 Diagram of the LCD pixel area at the VO user timing



- **VBP** in Figure 2-6 corresponds to **VSA** + **VBP** in Figure 2-5.
- **HBP** in Figure 2-6 corresponds to **HSA** + **HBP** in Figure 2-5.
- **HPW** in Figure 2-6 corresponds to **HSA** in Figure 2-5.
- **VPW** in Figure 2-6 corresponds to **VSA** in Figure 2-5.

- **HACT** in Figure 2-6 corresponds to **HACT** in Figure 2-5.
- **VACT** in Figure 2-6 corresponds to **VACT** in Figure 2-5.

Table 2-5 VO_SYNC_INFO_S structure

Member	Description
bSynm	Sync mode. The parameter is meaningless. Set it to **0**.
bIop	The value **0** indicates the interlaced mode, and **1** indicates the progressive mode. Set this parameter to **1** for MIPI and LCD panels.
u8Intfb	Bit width of the output interface. The parameter is meaningless. Set it to **0**.
u16Vact	Vertical active area (unit: line) For details, see the panel manual.
u16Vbb	Vertical back blank porch (unit: line) u16Vbb = VSA + VBP For details, see the panel manual.
u16Vfb	Vertical front blank porch (unit: line) For details, see the panel manual.
u16Hact	Horizontal active area (unit: pixel) For details, see the panel manual.
u16Hbb	Horizontal back blank porch (unit: pixel) u16Hbb = HSA + HBP For details, see the panel manual.
u16Hfb	Horizontal front blank porch (unit: pixel) For details, see the panel manual.
u16Hmid	Vertical sync active pixel value of the bottom field This parameter is meaningless in progressive mode. In progressive mode, set it to **1**.
u16Bvact	Vertical active area of the bottom field (unit: line) This parameter is valid in progressive mode, but is meaningless. In progressive mode, set it to **1**.
u16Bvbb	Vertical back blank porch of the bottom field (unit: line) This parameter is valid in progressive mode, but is meaningless. In progressive mode, set it to **1**.
u16Bvfb	Vertical front blank porch of the bottom field (unit: line) This parameter is valid in progressive mode, but is meaningless. In progressive mode, set it to **1**.
u16Hpw	Horizontal pulse width (unit: pixel)
u16Vpw	Vertical pulse width (unit: line)

Member	Description
bIdv	Polarity of the valid data signal. The value **0** indicates the signal is active high, and **1** indicates the signal is active low. Set it to **0** according to the MIPI requirement.
bIhs	Polarity of the horizontal valid data signal. The value **0** indicates the signal is active high, and **1** indicates the signal is active low. Set it to **0** according to the MIPI requirement.
bIvs	Polarity of the vertical valid data signal. The value **0** indicates the signal is active high, and **1** indicates the signal is active low. Set it to **0** according to the MIPI requirement.

To obtain the same panel settings, configure the user timing as follows:

```
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 Configuring the VO Output Clock

After configuring the user timing, you need to configure the VO clock frequency. The clock frequency for VO is the required clock frequency. The frequency value can be automatically generated based on the timing and frame rate by using the *Clock Timing Calculator for RGB*

and MIPI Panels. The output clock frequency of the VO interface is obtained by dividing the output clock by the frequency division coefficient. Therefore, to obtain the output clock of the interface, you need to determine the frequency division ratio and then confirm the clock frequency of the clock source. Then call HI_MPI_VO_SetUserIntfSyncInfo to configure the VO_USER_INTFSYNC_INFO_S structure. For details about the MPI, see chapter 4 "VO" in the *HiMPP V4.0 Media Processing Software Development Reference*.

User Timing Information

```
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;
```

Clock Source Attributes

```
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;
```

Clock Source Selection

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

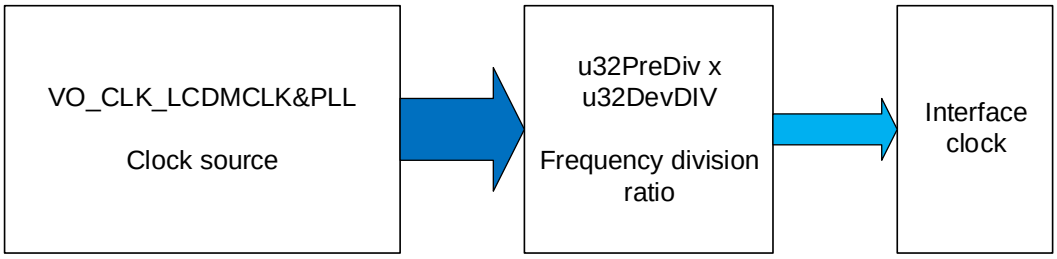
PLL Parameter Configuration

```
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;
```

2.2.6.1 Frequency Division Ratio

The frequency division ratio is the product of **u32PreDiv** and **u32DevDiv**. For details about pre-frequency division coefficient and clock frequency division coefficient, see chapter 4 "VO" in the *HiMPP V4.0 Media Processing Software Development Reference*.

Figure 2-7 Relationship between the clock frequency of the clock source and the interface clock



The VO VDP CRG configuration of each SoC provides the configuration of the frequency division ratio.

- If MIPI DSI output is used, set the frequency division ratio to **1**.
- MIPI panels do not need frequency division. Therefore, set both **u32PreDiv** and **u32DevDiv** to **1**.

2.2.6.2 Clock Reverse

For the MIPI panel, directly fix the reverse.

2.2.6.3 Frequency Division Clock

For details about how to calculate and configure the clock, see VO_USER_INTFSYNC_INFO_S in the *HiMPP V4.0 Media Processing Software Development Reference*.

Find the description about PLL selection and calculation in the SoC data sheet. You can also find the VDP clock control register among the CRG registers and check the clock source selected by the VDP.

Take Hi3516C V500 as an example. Run the command **himd.l 0x12010108**.

The frequency of the LCD frequency-divided clock or PLL clock frequency of the VDP is calculated as follows:

Take a 320 x 240 panel at 60 fps as an example. Assume that the timing information of the panel is as follows:

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

According to the mapping relationship described in Table 1-3:

u16Hact = 320, u16Hbb = 65, u16Hfb = 7, u16Vact = 240, u16Vbb = 15, u16Vfb = 9

Calculate the clock frequency (unit: MHz):

X = (u16Hact + u16Hbb + u16Hfb) x (u16Vact + u16Vbb + u16Vfb) x Frame rate x u32PreDiv x u32DevDiv/10^6

= (320 + 65 + 7) x (240 + 14 + 9) x 60 x 1 x 1/10^6

= 6.209

LCD frequency division coefficient (u32LcdMClkDiv) = (X/1188) x (2^27). The maximum value of **X** is 75 MHz. Therefore, u32LcdMClkDiv = (6.209/1188) x (2^27) = 0xAB448.

For parameters **u32Fbdiv**, **u32Frac**, **u32Refdiv**, **u32Postdiv1**, and **u32Postdiv2** for the PLL clock frequency: X = 24 x [u32Fbdiv + u32Frac/(2^24)]/u32Refdiv/u32Postdiv1/u32Postdiv2

Therefore, **u32Fbdiv**, **u32Frac**, **u32Refdiv**, **u32Postdiv1**, and **u32Postdiv2** are **1**, **0x35935F**, **2**, **2**, and **1**, respectively.

2.3 Configuration Verification and Debugging

2.3.1 Checking the Pin Multiplexing and Drive Capability

Step 1 Check the pin multiplexing of the control command interface SPI or I^2C.

Step 2 Check the pin multiplexing of the data interface (LCD_DATA, HSYNC, VSYNC, and CLK). During preliminary debugging, you can configure a higher level for the drive capability. After the panel is turned on, optimize the drive capability to avoid component damage.

Refer to the *HI35XX_PINOUT_EN* and hardware design diagram, and obtain the pin multiplexing configuration by using the command **himd+*register address***.

Modify the multiplexing configuration as required in the code.

Take the LCD_CLK pin of Hi3559 V200 as an example. Find the address of the register controlling LCD_CLK in *Hi3559V200_PINOUT_EN*, as shown in Figure 2-8.

Figure 2-8 LCD_CLK multiplexing relationship

0x112F_0034	0x0400	31:11	Reserved
		10	Level conversion rate control. The value 0 indicates that the level conversion rate is high. The value 1 indicates that the level conversion rate is low.
		9	Pull-down resistor enable, active high
		8	Pull-up resistor enable, active high
		7:4	Drive capability. Values 0–7 correspond to I/O drive capability IO4_level 1–8, respectively.
		3:0	Function selection: 0: GPIO0_6 2: LCD_CLK 3: VOU_CLK

If the value of bit[3:0] in the 0x112F0034 register is not **0x2**, it indicates that the LCD_CLK pin multiplexing configuration of the LCD panel is incorrect and needs to be modified in the code.

----End

2.3.2 Checking the Reset Process

Step 1 Check whether the reset pin can be set high/low normally.

1. Refer to the hardware design diagram and find the GPIO port corresponding to the reset pin. Ensure that the pin is multiplexed as GPIO.
2. Set the GPIO direction to output by referring to the "GPIO" section in chapter 01-12 "Peripherals" in the SoC data sheet.

The following takes GPIO5_2 of Hi3559 V200 as an example:

The base address of GPIO5 is 0x120D5000.

Figure 2-9 Base addresses of the GPIO port

GPIO Controller	Base Address
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

Find the register with the offset address 0x400. Set its second bit to **1**.

Figure 2-10 GPIO direction control register

GPIO_DIR

GPIO_DIR is a GPIO direction control register. It is used to configure the direction of each GPIO pin.

Offset Address: 400 Total Reset Value: 0x00

Bits	Access	Name	Description	Reset
[7: 0]	RW	gpio_dir	GPIO direction control register. Bit[7: 0] correspond to GPIO_DATA [7: 0] respectively. Each bit can be controlled separately. 0: input 1: output	0x00

3. Refer to the "GPIO" section in chapter 01-12 "Peripherals" in the SoC data sheet, check whether the GPIO port can be set high/low normally by using an oscilloscope.

   Assume that GPIO5_2 of Hi3559 V200 is set high.

   Find the corresponding GPIO_DATA register 0x120D5000+0x1>>(2+2). For GPIO5_3, the address of the GPIO_DATA register is 0x120D5000+0x1>>(3+2).

   Set the second bit of this register to **0**, and then to **1**, which correspond to the low level and high level.

Step 2 Check whether the reset operation is consistent with the panel requirement defined in the panel manual. For example, you can set the reset pin high, low, and then high again, to check the delay.

----End

2.3.3 Checking the Control Command Path

The HiSilicon MIPI TX driver provides HI_MIPI_TX_GET_CMD for obtaining the panel register values and HI_MIPI_TX_SET_CMD for setting the panel register values. You can use any of the following methods to verify the control command path. It is recommended that HI_MIPI_TX_GET_CMD and HI_MIPI_TX_SET_CMD be called before HI_MIPI_TX_ENABLE. If they are called after HI_MIPI_TX_ENABLE, the MIPI TX will switch from the LP mode to the HS mode. The LP mode is recommended for the control command.

- Read the panel ID register (generally 0xDA or 0xA1). For details about the exact register address, see the panel manual.

  To read a byte of the 0xDA register:

  Run the **0x37** command to set the maximum number of returned bytes to **1**. Then, run the **0x14** command to read one byte of data.

```
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);
```

  To read five bytes of the 0x1A register:

  Run the **0x37** command to set the maximum number of returned bytes to **5**. Then, run the **0x14** command to read five bytes of data.

```
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);
```

  Note: Using which read command (**0x14**, **0x24**, **0x4**, or **0x6**) depends on the panel, though **0x1A** and **0x6** are most commonly used.

  To facilitate debugging, HiSilicon provides the read and write tools mipitx_write and mipitx_read. These tools do not support the configuration of MIPI attributes. You can use

these tools only after HI_MIPI_TX_SET_DEV_CFG is called. For the Hi3559 V200 MobileCam as an example, mipitx_write and mipitx_read are stored in **board/Hi3559V200_MobileCam_SDK_V1.0.1.1/amp/a7_liteos/mpp/tools**. Call tools_cmd_register() on the Huawei LiteOS. Then, they are ready to be used in the command line.

- Find a register that can be written and read by the panel, try to write then read the register, and check whether the result is as expected.
- Enter the panel debugging mode or factory test mode by using simple control commands and check the control command path. For details about the procedure, see the panel manual.
- Use an oscilloscope to observe the waveform when a control command is sent and check whether the waveform meets the panel requirements. You can also directly analyze the transferred data by using a protocol analyzer if you know well about the MIPI protocol.

If the control command path fails, check the hardware connection and power supply, reset pin status (pay attention to the timing and level matching), and configuration of the combo_dev_cfg_t structure of the MIPI TX, especially members **phy_data_rate** and **lane_id**.

2.3.4 Checking the Backlight Control

There are three backlight control modes:

- Dynamically adjusting the backlight brightness

  In most cases, it is realized by adjusting the duty cycle using the PWM. The PWM driver needs to be implemented.
- Setting the backlight brightness by using the hardware (dynamic adjustment not supported)
- Setting the backlight brightness by using the panel driver IC, enabling backlight control by the MIPI TX

For details about the mode in use, see the panel manual. Ensure that the panel backlight can be enabled.

2.3.5 Checking the VO Output Clock and Timing

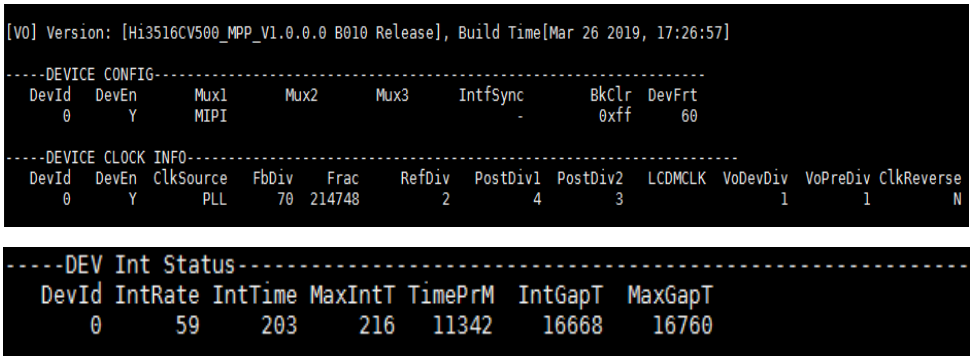
2.3.5.1 Checking the VO Output Clock

In the proc information of the VO module, **IntRate** indicates the frame rate of the device according to statistics. (HACT + HBB + HFB) x (VACT + VBB + VFB) x Frame rate = VO output clock frequency

If the information such as **HACT** and **HBB** is correct and the frame rate meets the expectation, it indicates that the output clock is correct.

During the running of a service program, run the **cat /proc/umap/vo** command on the terminal to view the proc information about the VO module.

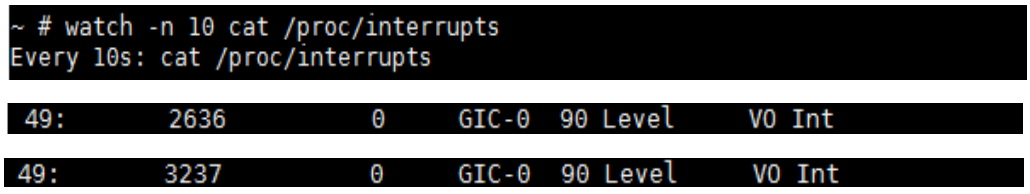
Figure 2-11 VO proc information



In the proc information of the VO module, obtain the configured clock source and clock source parameters, as well as the interrupt of the VDP logic, that is, the actual frame rate of the device.

In addition, when a media service runs on Linux, you can also run the **watch –n 10 cat /proc/interrupts** command to check the number of interrupts generated by the VO module every 10 seconds, and obtain the actual frame rate by calculation.

Figure 2-12 VO interrupt information



According to Figure 2-12: (3237 – 2636)/10 = 60

2.3.5.2 Checking the Timing

In the MIPI TX proc information, you can obtain the VO output timing detected by the MIPI TX in the MIPI TX DEV STATUS field, as shown in Figure 2-5. This field is described as follows:

- **width** corresponds to **HACT**.
- **height** corresponds to **VACT**.
- **HoriAll** corresponds to **HSA** + **HBP** + **HACT** + **HFP**.
- **VertAll** corresponds to **VSA** + **VBP** + **VACT** + **VFP**.
- **hbp** corresponds to **HBP**.
- **hsa** corresponds to **HSA**.
- **vsa** corresponds to **VSA**.

NOTICE

Convert the timing information detected by the MIPI TX into the VO timing and then compare it with the VO configuration parameters.

Figure 2-13 MIPI TX proc information

```
~ # 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

----------MIPI_Tx DEV STATUS--------------------------
   width  height HoriAll VertAll     hbp     hsa     vsa
    1080    1920    1237    1972      20       8      10
```

2.3.6 VO Debugging Using Color Bars

Color bar patterns are widely used for calibrating the timing and color reproduction errors.

VDP configuration errors can always be reflected on the color bars.

For instance, if the color bars are displayed incompletely or in a wrong area, the clock timing may be incorrect.

Using color bar patterns can help to locate faults. You need to analyze the faults according to the actual situation.

General debugging procedure: Enable the VDP device color bars by setting the VDP register DHD0_CTRL or DHD1_CTRL (Refer to the SoC data sheet for the accurate register name.) Enable bit **cbar_en**. Set the color space select bit **cbar_sel** to VGA. Set the color bar mode bit **cbar_mode** to horizontal or vertical. Set the update bit **regup** of the DHD0 register to **1**.

Take Hi3516C V500 as an example:

- Check the register: **himd.l 0x1144d000**
- Enable the color bars: **himm 0x1144d000 0xc0000011**
- Disable the color bars: **himm 0x1144d000 0x80000011**

Where, **0x11440000** is the base address of the VDP register, and **0xd000** is the offset address of the DHD0_CTRL register.

Figure 2-14 Horizontal VDP color bars

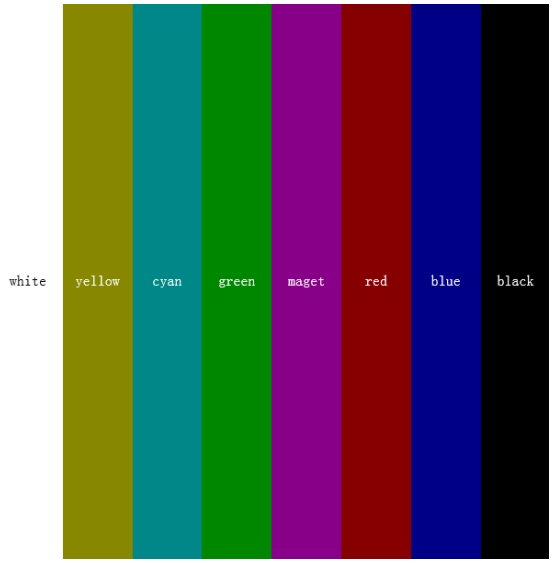
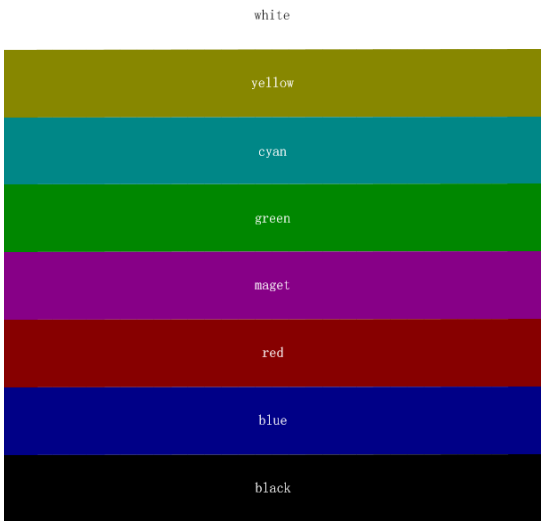


Figure 2-15 Vertical VDP color bars

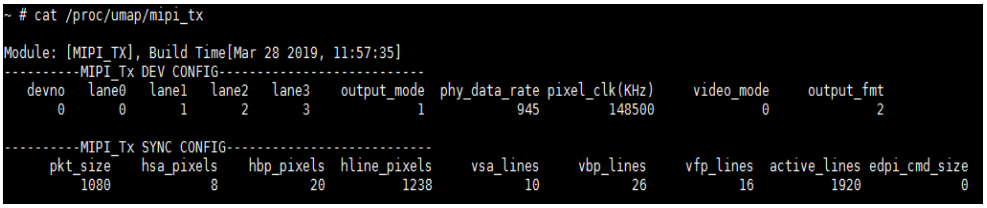


2.3.7 Checking the MIPI Attribute Configuration

You can use either of the following methods to check the configured MIPI device attributes:

- Method 1: Check the configured attributes by using the MIPI TX proc information. The proc information shows the user-defined parameter configuration.

Figure 2-16 Parameter configuration in the proc information



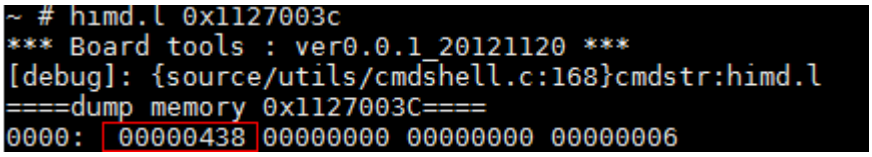
- Method 2: Check the configured attributes by reading the register values.

  For example: packet size configuration register VID_PKT_SIZE, HSA time configuration register VID_HSA_TIME, HBP time configuration register VID_HBP_TIME, HLINE time configuration register VID_HLINE_TIME, VSA line configuration register VID_VSA_LINES, VBP line configuration register VID_VBP_LINES, VFP line configuration register VID_VFP_LINES, and VACTIVE line configuration register VID_VACTIVE_LINES

  The values of registers VID_HSA_TIME, VID_HBP_TIME, and VID_HLINE_TIME are calculation-processed, which may be different from the configured ones. The conversion formula is as follows: VID_HSA_TIME = [(phy_data_rate +26)/27 x 27] x has_pixels x 125/pixel_clk

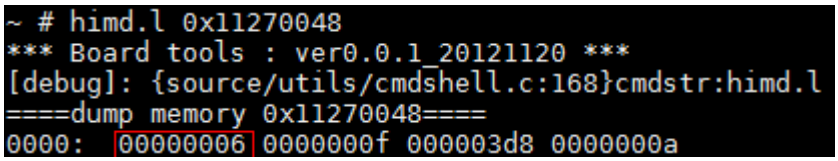
  – Figure 2-17 shows how to check the values of a register involves no calculation conversion, with VID_PKT_SIZE used as an example.

Figure 2-17 Register configuration information



  – Figure 2-18 shows how to check the values of a register involves calculation conversion, with VID_HSA_TIME used as an example.

Figure 2-18 Register configuration information



Calculate the value using the conversion formula as follows:

VID_HSA_TIME = [(phy_data_rate + 26)/27 x 27] x has_pixels x 125/pixel_clk

= [(945+26)/27 x 27] x 8 x 125/148500

= 6

To view the configured MIPI initialization sequence, call HI_MIPI_TX_GET_CMD of the MIPI TX at the panel side.

For details, see 2.2.2 "Resetting the Panel."

2.3.8 MIPI TX Debugging Using Color Bars

The MIPI TX module is independent of the VO module. Therefore, you can check whether the MIPI TX attributes are correctly configured by using the color bars provided by the MIPI TX module.

If the image display is available with the MIPI TX color bar pattern enabled but no image display is available with the MIPI TX color bar pattern disabled, it indicates that a problem occurs with the data path between the VO module and the MIPI TX. The following lists some possible causes to facilitate further analysis.

- If the color bars are displayed in a wrong area, for example, starting from the center of the screen, check the clock timing.
- If the color reproduction of the color bars are abnormal, check the RGB sequences of the panel and the HiSilicon platform, and the **output_format_t** configuration of the MIPI TX.

  If the picture cannot be displayed after MIPI TX color bars is configured, it indicates that a problem occurs when the MIPI TX sends pictures to the panel.

Enable or disable the color bar pattern of MIPI TX by configuring the registers.

- To enable the color bar pattern, perform the following steps:

Step 1 Find the address of register OPERATION_MODE in the SoC data sheet and set it to **0x0**.

Step 2 Find the address of register VID_MODE_CFG and set bit 16 (**vpg_en**) to **1**. Keep other bits unchanged.

Step 3 Find the address of register PWR_UP, and set it to **0x0** and then to **0x1**.

----End

- To disable the color bar pattern, perform the following steps:

Step 1 Find the address of register VID_MODE_CFG in the "MIPI TX" section in the SoC data sheet and set bit 16 (**vpg_en**) to **0**. Keep other bits unchanged.

Step 2 Find the address of register OPERATION_MODE and set it to **0x80050000**.

Step 3 Find the address of register PWR_UP, and set it to **0x0** and then to **0x1**.

----End

The following takes Hi3516 CV500 as an example with **output_mode** set to **OUTPUT_MODE_DSI_VIDEO** and **video_mode** set to **BURST_MODE**.

- Enabling the color bar pattern:

  himm 0x11270208 0x0

  himm 0x11270038 0x13f02

  himm 0x11270004 0x0

  himm 0x11270004 0x1
- Disabling the color bar pattern:

himm 0x11270038 0x3f02

himm 0x11270208 0x80050000

himm 0x11270004 0x0

himm 0x11270004 0x1

Where, **0x1127** is the base address of MIPI registers, **0x00000208** is the offset address of register OPERATION_MODE, **0x00000038** is the offset address of register VID_MODE_CFG, and **0x00000004** is the offset address of register PWR_UP.

NOTICE

MIPI TX in CMD mode does not support color bars. You are advised to enable MIPI TX before using the color bars for debugging.

Figure 2-19 and Figure 2-20 show the color bar styles.

Figure 2-19 Vertical MIPI color bars

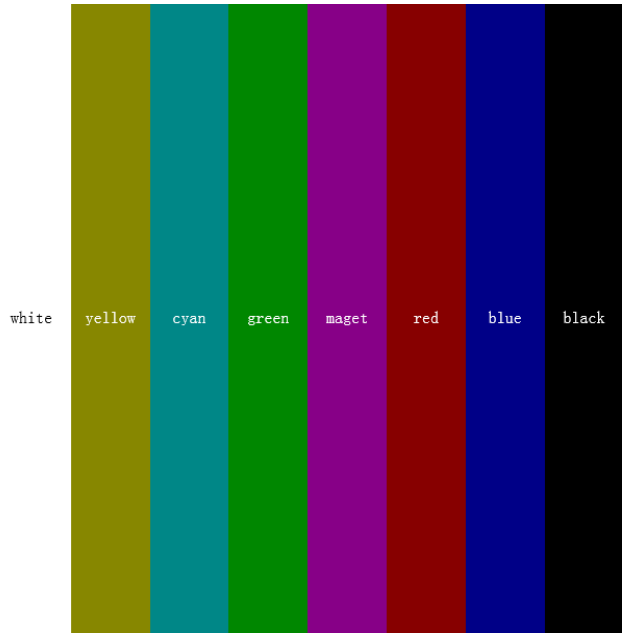
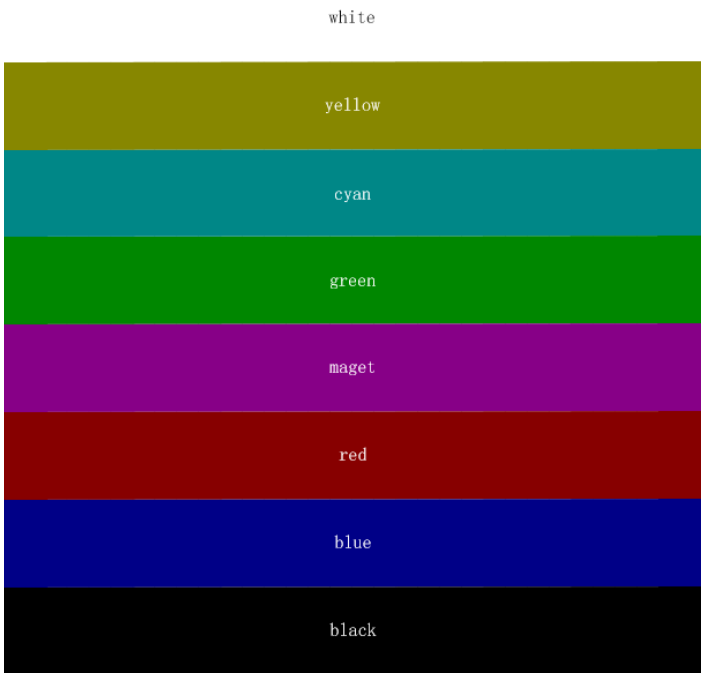


Figure 2-20 Horizontal MIPI color bars



2.4 FAQs for LCD Debugging

2.4.1 Black Screen Without Response

[Symptom]

The screen goes complete black and does not respond.

[Cause Analysis]

There are many possible causes, for example, a damaged component, a connection failure, a power supply failure, incorrect pin multiplexing configuration, an incorrect status of the reset pin, incorrect VDP parameter configuration, and incorrect MIPI parameter configuration. You need to check them one by one.

[Solution]

Step 1 Check the hardware.

Contact hardware engineers to check whether the power supply and connection of the panel are normal and whether the components are intact. Pay attention to the I/O level matching.

Step 2 Check the software process.

Compare it with the initialization process of the MIPI device in 1.2.2 "Resetting the Panel." Note that the operation on the reset pin should strictly follow the time requirements. Generally, a control command can be received in 120 ms after the reset.

Step 3 Check the pin multiplexing and drive capability. For details, see 2.3.1 "Checking the Pin Multiplexing and Drive Capability."

Step 4 Check the control command path. For details, see section 2.3.3 "Checking the Control Command Path."

Step 5 Check whether the backlight is enabled.

For an MIPI panel, the backlight can be enabled in either of the following modes: Set the level of the I/O interface (GPIO or PWM) or send a backlight control command on the MIPI device. For details about the mode in use, see the panel manual. Ensure that the panel backlight can be enabled.

Step 6 Check the configured MIPI TX video timing.

If the control command path is normal, check the VDP or MIPI TX parameter configuration. The MIPI TX module is independent of the VDP module. Therefore, you can check whether the MIPI TX attributes are correctly configured by using the color bars provided by the MIPI TX module. For details about the color bars, see 2.3.8 "MIPI TX Debugging Using Color Bars." The color bar pattern can be enabled only after HI_MIPI_TX_ENABLE is called.

If the MIPI TX color bar pattern is not displayed or the display area is abnormal, check the **combo_dev_cfg_t** configuration. For details, see 2.2.4 "Configuring the Panel." If the problem persists, you are advised to go to Step 1 and try again. If no error is found in the preceding steps, the possible cause may be the incompatibility between the timing parameters of the MIPI D-PHY and those on the screen. In this case, check section 2.4.14 "Check Whether Timings Between MIPI D-PHY and Screen Are Matched."

Step 7 Check the VDP output and whether it matches MIPI TX **pixel_clk**.

The video signals of the MIPI TX come from the VDP module. On the basis of Step 4, if the MIPI TX color bars can be correctly displayed, but the VDP image cannot be displayed, check the VDP attribute configuration (including the VDP output clock) and the **pixel_clk** configuration in the structure combo_dev_cfg_t.

For details about how to configure the VDP, see 2.2.5 Configuring the VO Output Timing. For details about the output clock configuration, see Configuring the Output Clock.

----End

2.4.2 Incorrect Position of Display

[Symptom]

Only a part of the image is displayed.

[Cause Analysis]

Only a part of an image is displayed on the screen, and the start point of the displayed image is not the initial point of the screen.

[Solution]

This problem is usually caused by the mismatch between the configuration of the blanking interval of the line or field of the SoC and that of the panel. In this case, you need to modify the configurations on both sides to be consistent. You can modify the configuration on either SoC or panel side.

2.4.3 Incorrect Display Start

[Symptom]

The start of the image is on the center of the screen.

[Cause Analysis]

Changing the back and front porch settings makes no difference. The possible cause is that the output pixel clock and the timing do not match.

[Solution]

Check the clock by referring to section 1.3.5 "Checking the VO Output Clock and Timing."

2.4.4 Image in Disorder

[Symptom]

After the panel is turned on, it is found that the displayed image is in disorder. After the MIPI TX color bar pattern is enabled, the colors of some of bars are disordered.

[Cause Analysis]

The scan mode changes randomly.

[Root Causes]

The scan mode of the MIPI panel is determined by the hardware. When the voltage of the interface for the scanning mode is inaccurate, the scan mode changes randomly. As a result, the image is disordered.

2.4.5 Incorrect Display Size

[Symptom]

The actual size of the displayed image does not match the screen.

[Cause Analysis]

In this case, the actual size of the displayed image exceeds the screen size (that is, the image is displayed incompletely) or is smaller than the screen size.

[Solution]

This problem is usually caused by the mismatch between the configuration of the active region of the line or field on the SoC side and that of the panel. In this case, you need to modify the configurations on both sides to be consistent. You can modify the configuration on either SoC or panel side.

2.4.6 Abnormal Colors

The colors of the image are obviously different from what are expected. This problem has multiple symptoms.

Color Changes

[Symptom]

The displayed red, blue, and green colors are out of order.

[Cause Analysis]

The RGB TX sequences on the SoC and panel do not match.

[Solution]

Adjust the RGB RX sequence on the panel side. It can be configured by using the register for some panels.

Color Deviation

[Symptom]

Color deviation occurs.

[Cause Analysis]

The configured output mode of the MIPI device may be incorrect. As a result, MIPI device parameters are incorrectly configured.

[Solution]

Change the output mode of the MIPI device. The MIPI TX output mode is generally set to **DSI_VIDEO** or **DSI_CMD**, which is determined based on the specific screen requirements. For details about the configuration, consult the screen manufacturer.

Abnormal Color Display

[Symptom]

The displayed colors are abnormal, and the color conversion mode is incorrect.

[Cause Analysis]

The color space conversion (CSC) of the video layer may not be correctly configured. The output has to be converted to the RGB format by using the CSC function for the LCD or MIPI screen.

[Solution]

Call the HI_MPI_VO_SetVideoLayerCSC or HI_MPI_VO_SetGraphicLayerCSC function to set CSC attributes. For details about how to set the parameters, see the related interface operations in 4 "VO" in the *HiMPP V4.0 Media Processing Software Development Reference*.

Color Deviation in the Displayed Lines

[Symptom]

There is a color deviation around the contour in the image.

[Cause Analysis]

The line color is abnormal, especially the contour line of the image, which is affected by adjacent pixels. This problem is usually caused by a minor error occurred in the timing. As a result, the information about the pixel and the information about the adjacent pixels are transmitted incorrectly during data transmission. There are many possible causes of this situation.

[Solution]

- If the hardware interference is high, use an oscilloscope to check whether the waveform meets the expectation. If not, the possible cause is that the line is too long (which generally only occurs in the jump wire), or the pin drive capability is insufficient.

- The clk phase of the video output is incorrect and does not match that of the panel. You need to correct timing configured for the SoC and panel.

2.4.7 Unsatisfactory Display Effect

[Symptom]

Performance in aspects such as the brightness, chroma, and contrast are unsatisfactory.

[Cause Analysis]

The display effect of the screen is not satisfactory.

[Solution]

To adjust the display effect of the screen, you need to:

- Adjust the output picture effect on the SoC.
- Adjust the display effect parameters on the screen.

The display effect of the SoC can be adjusted by modifying parameters such as VO parameters, luminance, and contrast. The display effect of the panel can be adjusted by modifying the gamma curve, luminance, and chroma. Generally, the panel vendor provides recommended configurations, which meet the requirements in most scenarios.

2.4.8 Frame Residue During Image Conversion

[Symptom]

Frame residue occurs during image conversion.

[Cause Analysis]

It is usually caused by improper Vcom voltage configuration or abnormal voltage of some power pins.

[Solution]

For details, see the description in the panel manual or seek solutions from the panel vendor.

2.4.9 Low Frame Rate or Freezing Frame

[Symptom]

Frame freezing occurs during the display.

[Cause Analysis]

Generally, the frequency of the data clock is insufficient.

[Solution]

Adjust the clock frequency.

2.4.10 Dark Image with Normal Backlight

[Symptom]

The image is dark but the backlight configuration is correct.

[Cause Analysis]

Some data lines of the RGB screen have no data. The pins are not correctly configured during the hardware design by following the *Hi35xx* As a result, there is not data in some of the pins between the screen and the SoC.

[Solution]

There are many causes for a dark image, but you can take the priority to rectify the fault as described.

2.4.11 RGB Panel Display Fails After the Display Service Is Restarted

[Symptom]

After the display service is exited and restarted, the RGB panel has no display. In this case, the panel driver needs to be reconfigured.

[Cause Analysis]

This is related to the features of the RGB panel. For some RGB panels, before exiting the display service, you need to turn off the panel or bring the panel into sleep mode. Turn on or wake up the panel after display service is restarted.

[Solution]

Before exiting the display service, turn off the panel or bring the panel into sleep mode. Turn on or wake up the panel after display service is restarted.

2.4.12 MIPI Fails to Detect the Line Information Sent by the Front-end

[Symptom]

During the debugging, it is found that the VDP clock and timing as well as the number of interrupts are correct. But according to the detection of the MIPI, **VACT** and **VALL** from the front end are 0, but **HACT** and **HALL** are correct.

[Cause Analysis]

The **phy_data_rate** attribute in the MIPI device parameter configuration is set to a small value. As a result, the MIPI restarts repeatedly, and the detected value of **VACT** is **0**.

[Solution]

You can try either of the following solutions:

- Solution 1: Set **phy_data_rate** to a proper value (You are advised to increase the value to 1.2 times or more of the theoretical bandwidth. Note that the value cannot exceed the RX range on the panel.)
- Solution 2: For Hi3519A V100 and Hi3556A V100, ensure that the **mipitxphy_cmos_mode_enable** bit of the MISC_CTRL1 register is set to **0**. For details about this register, see the *Hi35xx Vxx Camera SoC Data Sheet*.

2.4.13 MIPI Device Attribute phy_data_rate Differs Greatly From the Actual Result

[Symptom]

The detected MIPI output clock frequency is much smaller than the expected value.

[Cause Analysis]

When the **phy_data_rate** attribute is set, the detected frequency of the MIPI clock should be half of **phy_data_rate**. The actual value differs greatly from the expected value.

[Solution]

Check the voltage of the MIPI power supply. Change the voltage as required.

2.4.14 Check Whether Timings Between MIPI D-PHY and Screen Are Matched

[Symptom]

The MIPI TX configuration, VDP configuration, peripheral hardware configuration, and hardware operation sequence have all been checked. However, the screen is still all black.

[Cause Analysis]

The MIPI D-PHY configured by using the HiSilicon MIPI TX driver works under timing parameters with good compatibility. However, the screen may have special requirements on the timing. Therefore, the HiSilicon MIPI TX driver needs to be modified based on the MIPI D-PHY timing parameters required by the screen to match the timings at the TX end and RX end.

[Solution]

Step 1 Refer to the guidance on the driver IC of the screen and find the descriptions about the MIPI D-PHY timing parameters (such as **TLPX** and **THS-TRAIL**).

Step 2 Use a high-speed oscilloscope to measure the MIPI timing on the board and compare the timing with that queried in Step 1. If the timing is out of the range, the screen may not display any image due to timing mismatch.

Step 3 If the MIPI D-PHY timing parameters do not meet the timing requirements of the screen in Step 2, you need to modify the MIPI TX driver.

For example, to add the timing parameter **THS-TRAIL**, find **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)**, and **set_phy_reg(DATA3_THS_PREPARE, data_ths_prepare)** in **mipi_tx_hal.c**, and add the configured parameter **data_ths_prepare**. Note that if the configured value of the **data_ths_prepare** parameter is increased by 1, the actual **THS-TRAIL** time is increased by 8000/actual_phy_data_rate (unit: ns). For details about how to calculate **actual_phy_data_rate**, see **mipi_tx_hal.c**.

For more details about configuration of the MIPI TX D-PHY, see section 9.4 "MIPI TX" in the *Hi35xx xx Camera SoC Data Sheet*.

