PANEL 开发参考

Version 0.01

REVISION HISTORY

Revision No.	Description	Date
0.01	• Initial release	11/21/2019

目录

REVISION HISTORY i

1. 概述 2

编写目的 2

适用范围 2

相关人员 2

2. 模块介绍 3

2.1 模块功能介绍 3

2.2 模块源码文件介绍 3

3. 模块功能 4

3.1 多路拼图 4

3.2 PIP 4

3.3 图像缩放 5

3.4 图像裁剪 6

3.5 图像旋转 7

3.6 SSC 8

3.7 PQ 8

3.7.1 3x3 Matrix 8

3.7.2 Color Tempeture 9

3.7.3 Gamma 9

3.8 输出设备 13

3.8.1 TTL 13

3.8.2 MIPI 15

4. 点屏配置 18

5. 参考 DEMO 26

6. 调试方法和常见问题 27

7. TTL/MIPI 硬件连接 32

1. 概述

编写目的

本文档主要的内容是 DISP 模块在 LCD 输出场景下的使用说明，让显示相关应用开发人员了解显示驱动的接口及使用流程，了解显示模块功能及使用限制，并介绍了相关调试方法和问题排查手段。

适用范围

SSD201
SSD202
SSR623

相关人员

与显示相关的应用开发人员，及与显示相关的其他模块的开发人员。

2. 模块介绍

2.1 模块功能介绍

a) 支持两个 video layer，layer0 支持 16 windows 拼图，layer1 单个 window 作为 PIP 使用
b) 单个 window scaling up
c) 单个 window crop
d) 支持以 video layer 为单位做 ratote 90/270
e) video pixel format 支持 NV12
f) 支持 Contrast、Hue、Luma、Saturation、Sharpness、Gamma 等图像 PQ 调整
g) 支持 SSC，有效减低 EMI
h) 支持 LCD 显示（包括 TTL PANEL、MIPI PANEL），TTL 输出 CLK 支持范围 9Mhz-75Mhz，MIPI DSI 输出 CLK 支持范围 100Mbps/lane – 1.5Gbps/lane

2.2 模块源码文件介绍

- alkaid/sdk/interface/inclue/disp/mi_disp.h
  声明 DISP 所有的用户调用 API，定义了 DISP API 当前的 version num

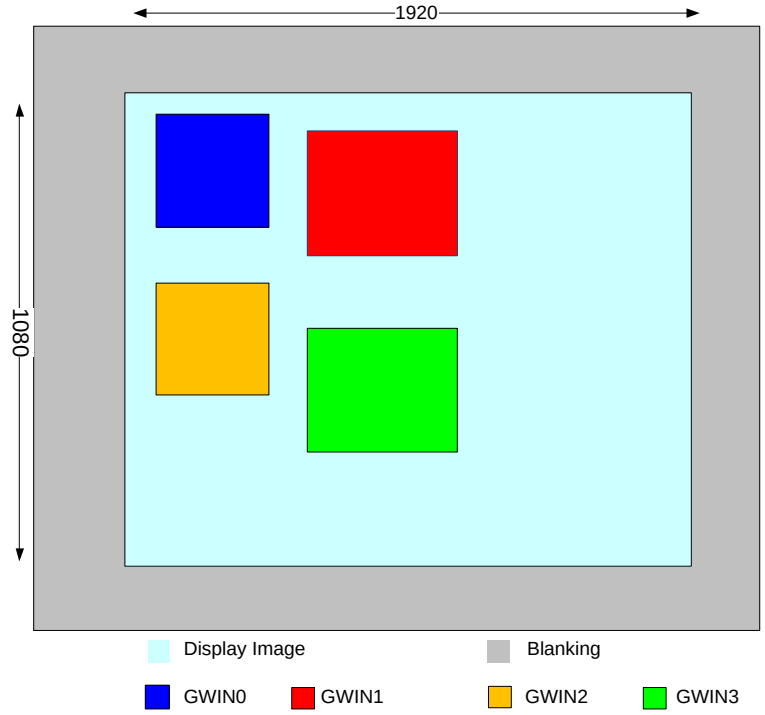
- alkaid/sdk/interface/inclue/disp/mi_disp_datatype.h
  定义了 DISP API 使用到的数据结构类型

- alkaid/sdk/interface/src/disp/disp_api.c
  DISP API 实现，最终编译生成 libmi_disp.so libmi_disp.a

3. 模块功能

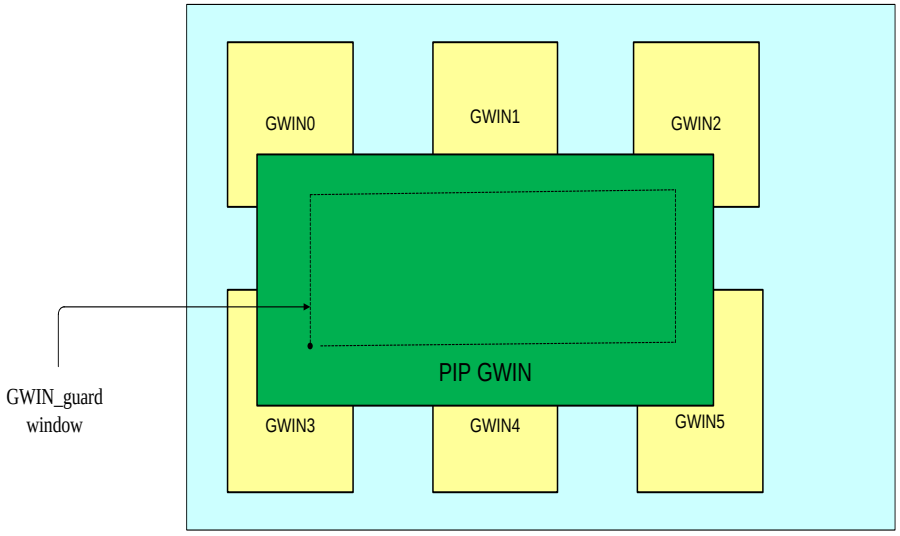
3.1 多路拼图

DISP video layer0 硬件有 16 个 mgwin，软件抽象为 16 个 input port，可以接收前端输出的 yuv 数据（NV12），16 个 input port 在 video layer0 上的显示位置可以任意设置，但 16 个 input port 的显示位置不能相互叠加。



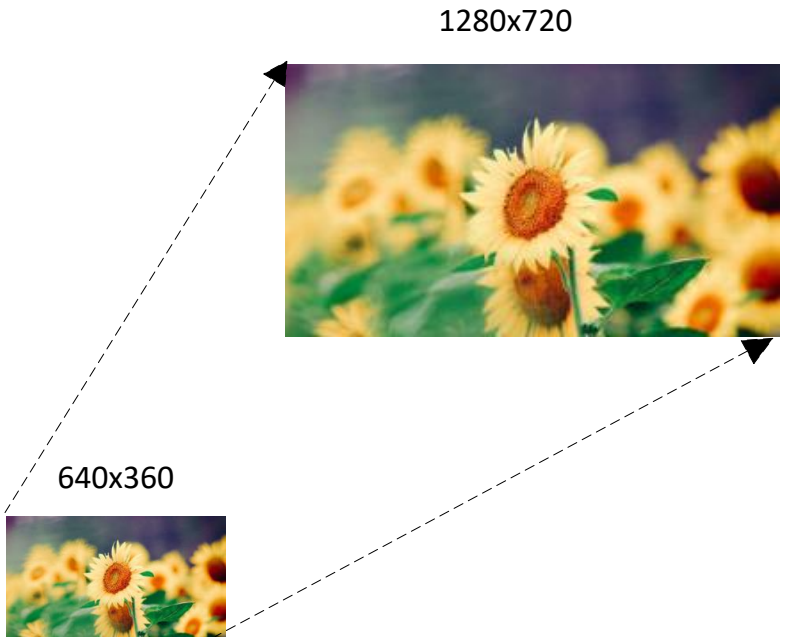
3.2 PIP

DISP video layer1 可以用于 PIP，将画面叠加到 video layer0 显示，默认 video layer1 显示优先级高于 video layer0，即如果两个 video layer 上都有内容，那么 video layer1 显示在上，video layer0 显示在下。



3.3 图像缩放

DISP 支持针对单个 input port 做 scaling up，H/V 方向放大比例可以达到 16 倍，来满足每个 window 对最终显示尺寸的需求。



相关 API：

MI_S32 MI_DISP_SetInputPortAttr(MI_DISP_LAYER DispLayer, MI_DISP_INPUTPORT LayerInputPort, const MI_DISP_InputPortAttr_t *pstInputPortAttr);

- ✧ DispLayer
  可以使用任意一个 video layer 上的任意一个 input port 做 scaling up，video layer 使用范围 0-1
- ✧ LayerInputPort
  可以使用任意一个 input port 做 scaling up。如果使用 video layer0，input port 使用范围 0-15；如果使用 video layer1，则只有一个 input port 可以使用。
- ✧ pstInputPortAttr

```
typedef struct MI_DISP_VidWin_Rect_s
{
    MI_U16 u16X;
    MI_U16 u16Y;
    MI_U16 u16Width;
    MI_U16 u16Height;
} MI_DISP_VidWinRect_t;

typedef struct MI_DISP_InputPortAttr_s
{
    MI_DISP_VidWinRect_t stDispWin;
    MI_U16 u16SrcWidth;
    MI_U16 u16SrcHeight;
```

} MI_DISP_InputPortAttr_t;

定义 input port 属性

u16SrcWidth：input image width

u16SrcHeight：input image height

u16X：在 layer 上的显示位置 H 方向偏移

u16Y：在 layer 上的显示位置 V 方向偏移

u16Width：放大后图像显示的宽，不放大等于 u16SrcWidth

u16Height：放大后图像显示的高，不放大等于 u16SrcHeight

同样如 3.1 所述，放大后的各个 window 之间不能叠加。

video layer size 需要等于最终输出 timing 的有效 width/height，图像才能全屏显示。比如，现在使用一个 1920x1080 的 panel 显示，那么输出 timing 的有效 width 等于 1920，height 等于 1080，如果只开启了一个 input port 并且希望该 port 全屏显示，那么应该满足：

u16Width = video layer width

u16Height = video layer height

video layer width = panel active width

video layer height = panel active height

3.4 图像裁剪

DISP 支持对 input image 做 crop，适用于 zoom in 场景，需要对局部区域放大显示。

相关 API：

MI_S32 MI_DISP_SetZoomInWindow(MI_DISP_LAYER DispLayer, MI_DISP_INPUTPORT LayerInputPort, MI_DISP_VidWinRect_t* pstZoomRect);

- DispLayer

  可以对任意 video layer 上的任意 input port 做 crop，layer id 取值范围 0-1

- LayerInputPort

  可以对任意 input port 做 crop，如果使用 video layer0，input port 取值范围 0-15，如果使用 video layer1，则只有一个 port 可以使用。

- pstZoomRect

  typedef struct MI_DISP_VidWin_Rect_s

  {

  ¦ MI_U16 u16X;

  ¦ MI_U16 u16Y;

```
    MI_U16 u16Width;
    MI_U16 u16Height;
} MI_DISP_VidWinRect_t;
```

设定 crop 属性：

u16X：crop H 方向起始点

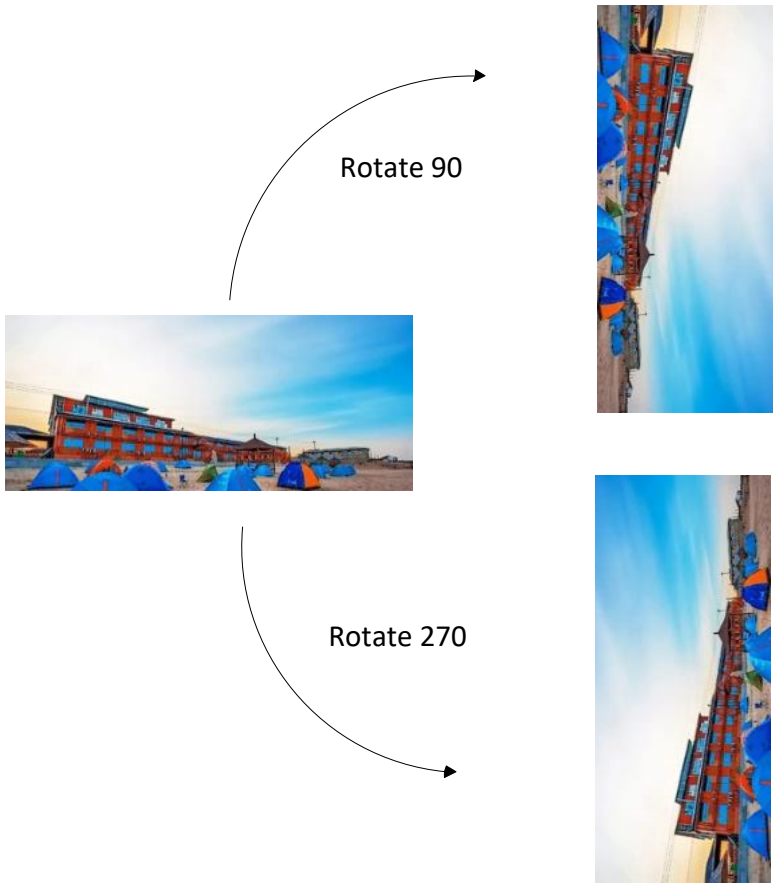
u16Y：crop V 方向起始点

u16Width：crop width

u16Height：crop height

3.5 图像旋转

DISP 支持以 layer 为单位做旋转 90 度或者 270 度，同时支持旋转加放大，以满足旋转后需要满屏显示的场景。



相关 API：

MI_S32 MI_DISP_SetVideoLayerRotateMode(MI_DISP_LAYER DispLayer, MI_DISP_RotateConfig_t *pstRotateConfig);

- ✧ DispLayer
  可以使用任意一个 video layer 做旋转，video layer 取值范围 0-1
- ✧ pstRotateConfig

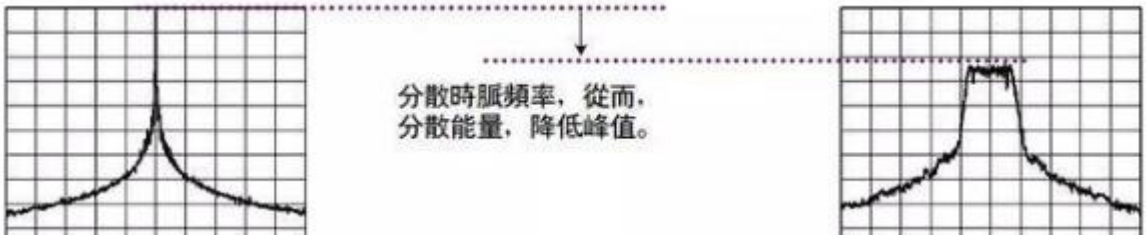
```
typedef enum
{
```

```
    E_MI_DISP_ROTATE_NONE,
    E_MI_DISP_ROTATE_90,
    E_MI_DISP_ROTATE_180,
    E_MI_DISP_ROTATE_270,
    E_MI_DISP_ROTATE_NUM,
}MI_DISP_RotateMode_e;
typedef struct MI_DISP_RotateConfig_s
{
    MI_DISP_RotateMode_e eRotateMode;
}MI_DISP_RotateConfig_t;
```

设置旋转角度，E_MI_DISP_ROTATE_90 旋转 90 度，E_MI_DISP_ROTATE_270 旋转 270 度，不支持旋转 180 度。

3.6 SSC

时钟展频通过频率调制的手段将集中在窄频带范围内的能量分散到设定的宽频带范围，通过降低时钟在基频和奇次谐波频率的幅度（能量），达到降低系统电磁辐射峰值的目的。



SSCG = OFF **SSCG = ON**

SSC 在屏参中配置，SSC step/span 有展频计算表中的公式来计算得到。

3.7 PQ

3.7.1 3x3 Matrix

DISP 支持通过 3x3 CSC Matrix 调节图像 Contrast、Hue、Luma、Saturation、Sharpness。

相关 API:

MI_S32 MI_DISP_IMPL_SetLcdParam(MI_DISP_DEV DispDev, MI_DISP_LcdParam_t *pstLcdParam)

- ✧ DispDev
  DISP 设备 ID 设为 0
- ✧ pstLcdParam

```
typedef struct MI_DISP_Csc_s
{
    MI_DISP_CscMattrix_e eCscMatrix;      /* eCscMatrix   使用第 4 组 Matrix*/
    MI_U32 u32Luma;                       /* luminance:    0 ~ 100 default: 50 */
    MI_U32 u32Contrast;                   /* contrast:     0 ~ 100 default: 50 */
    MI_U32 u32Hue;                        /* hue:      0 ~ 100 default: 50 */
    MI_U32 u32Saturation;                 /* saturation:   0 ~ 100 default: 40 */
} MI_DISP_Csc_t;
```

```
typedef struct MI_DISP_LcdParam_s
{
    MI_DISP_Csc_t stCsc;
    MI_U32 u32Sharpness;
} MI_DISP_LcdParam_t;
```

3.7.2 Color Tempeture

DISP 支持图像 R/G/B 分量的色温调节。

相关 API:

MI_S32 MI_DISP_DeviceSetColorTempeture(MI_DISP_DEV DispDev, MI_DISP_ColorTemperature_t *pstColorTempInfo);

- ✧ DispDev

  DISP 设备 ID 设为 0

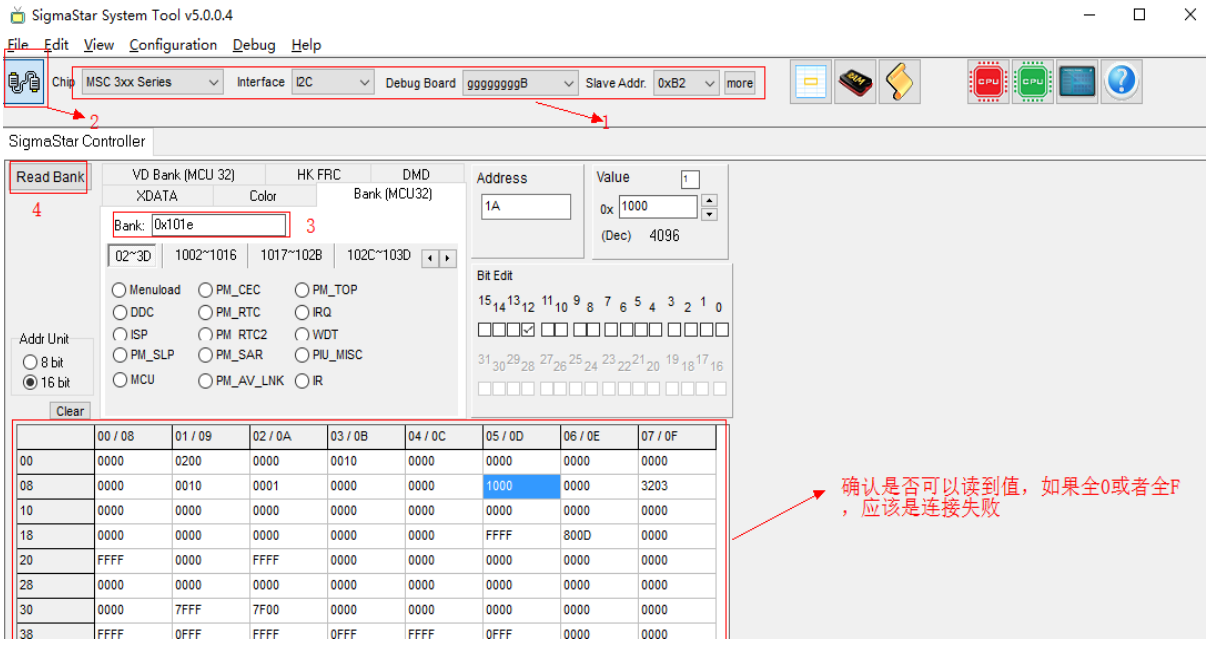
- ✧ pstColorTempInfo

```
typedef struct
{
    MI_U16 u16RedOffset;
    MI_U16 u16GreenOffset;
    MI_U16 u16BlueOffset;
    MI_U16 u16RedColor;   // 00~FF, 0x80 is no change
    MI_U16 u16GreenColor;// 00~FF, 0x80 is no change
    MI_U16 u16BlueColor; // 00~FF, 0x80 is no change
}MI_DISP_ColorTemperature_t;
```

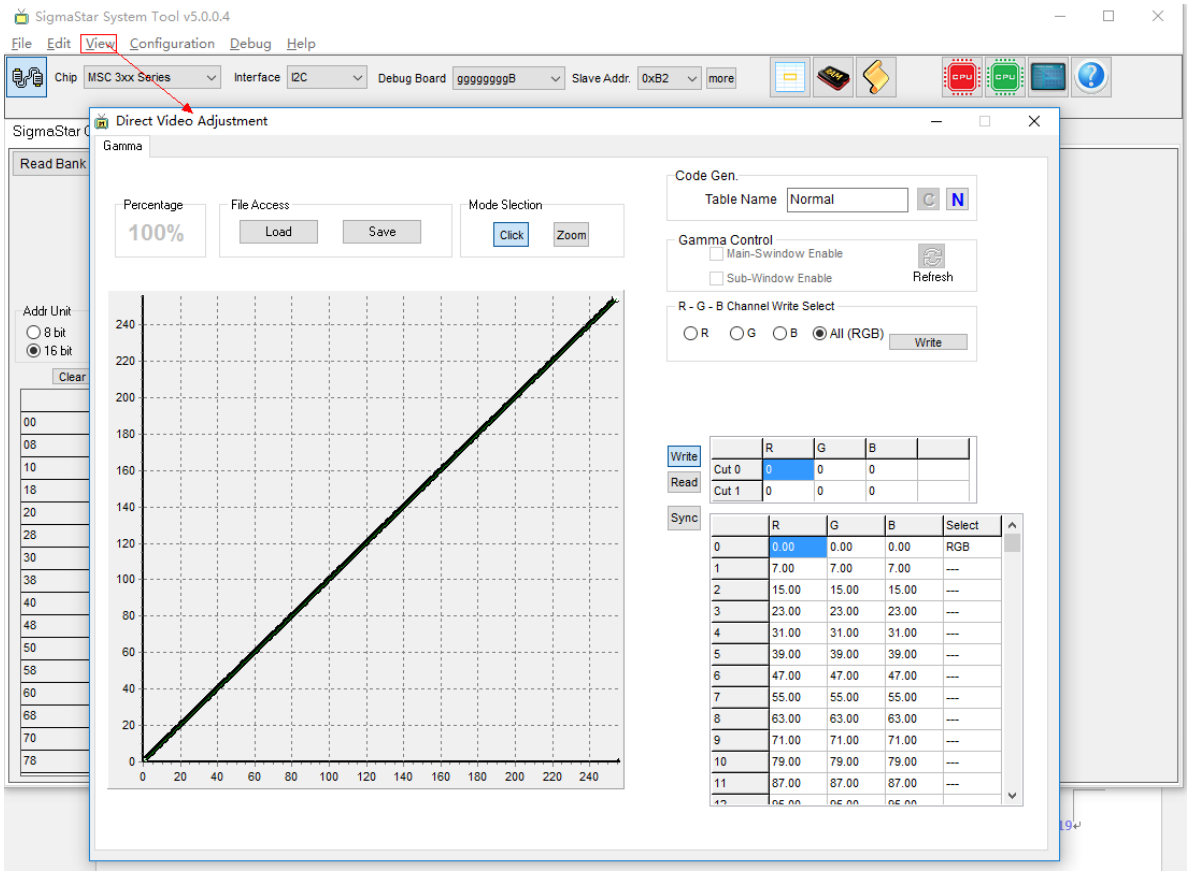
3.7.3 Gamma

方法一：使用 SigmaStar System Tool 调 GAMMA，参考以下步骤：

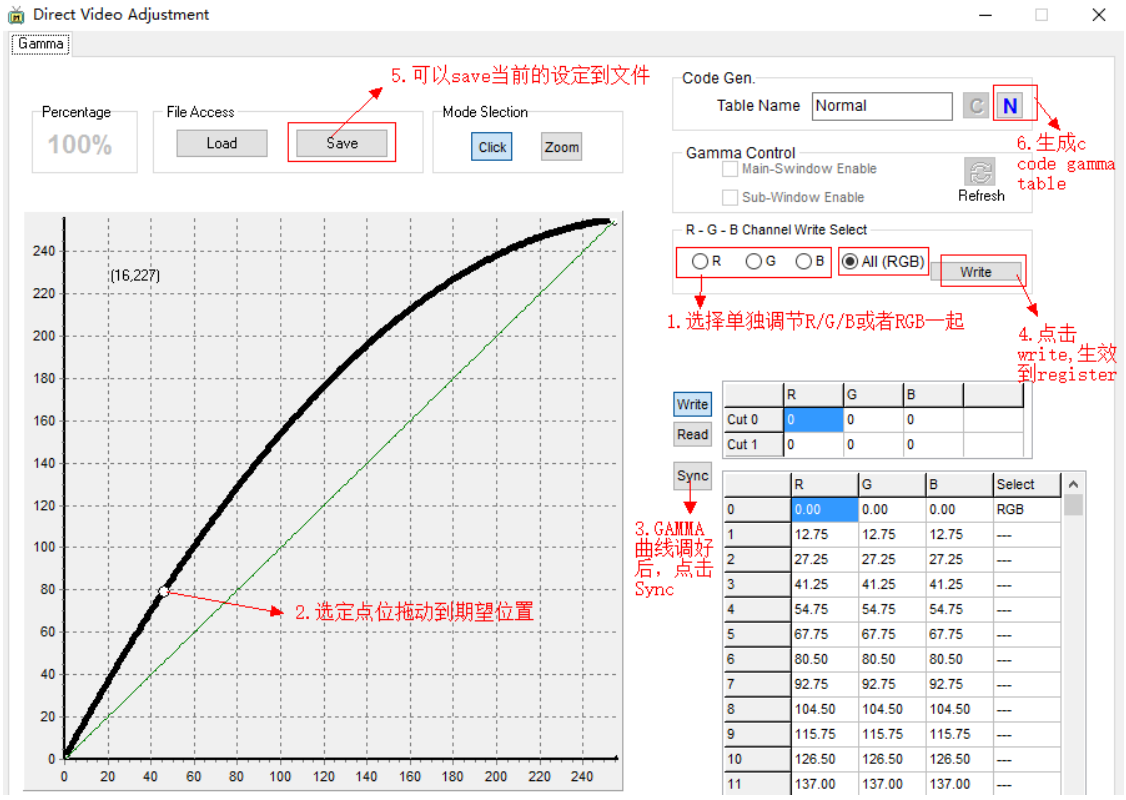
A. 连接到 chip



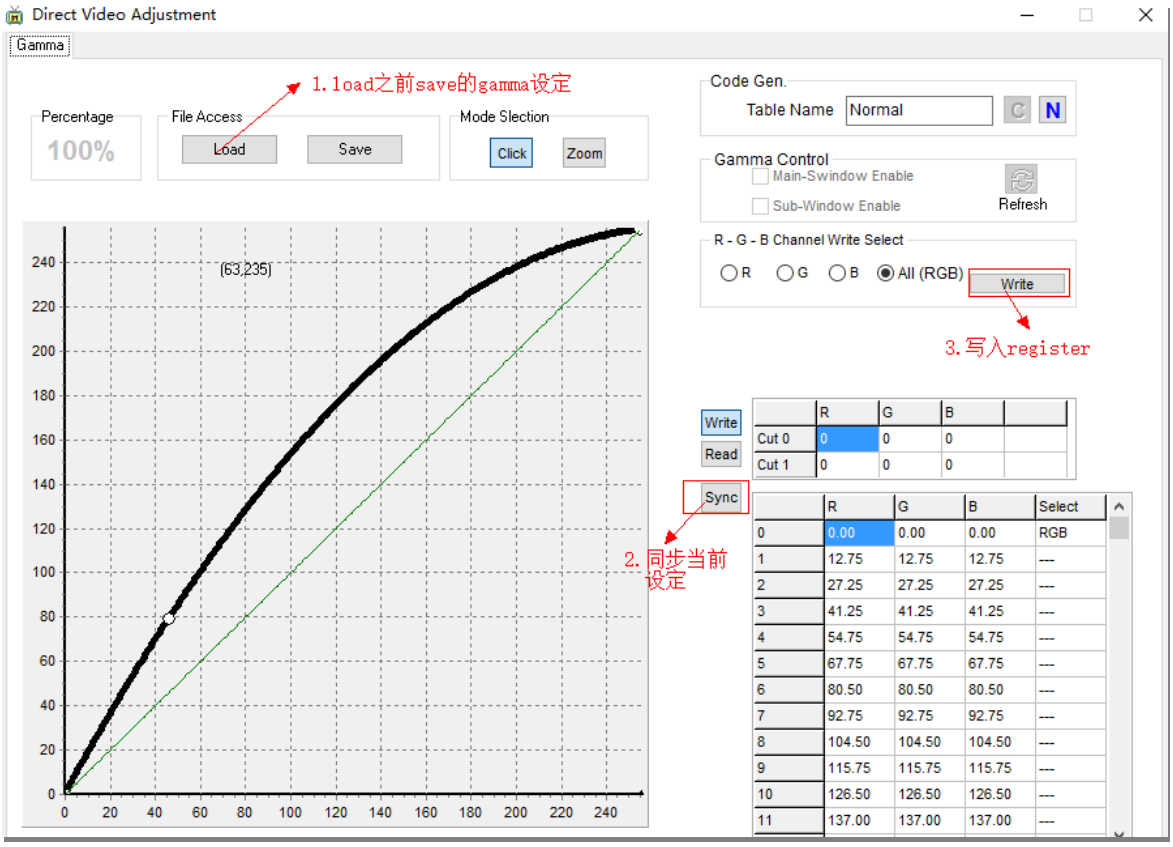
B. 打开 GAMMA 调整界面



C. GAMMA 调节

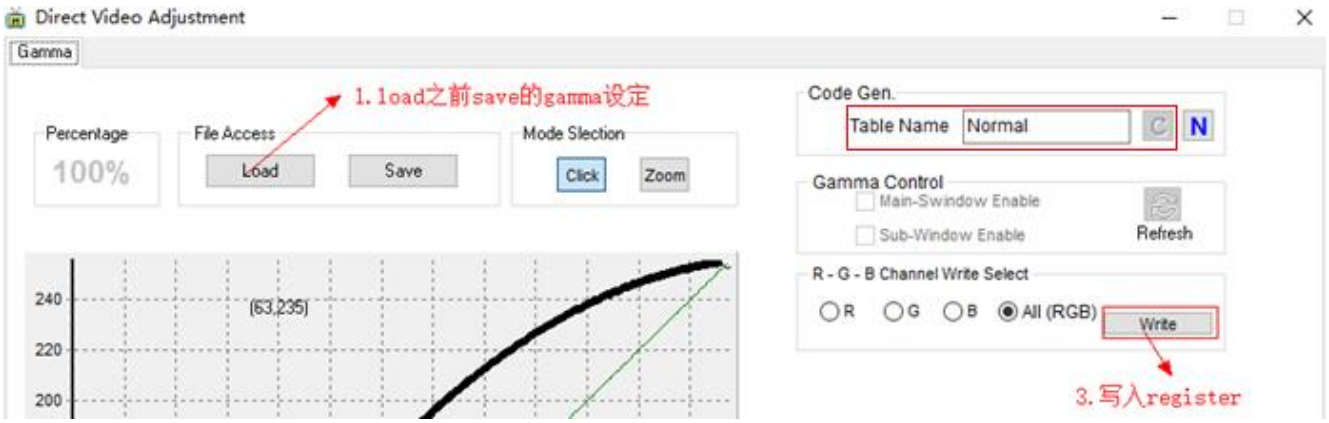


D. load gamma 设定



方法二：使用 DISP API 设定 GAMMA

可以将调好的 gamma 设定通过 SigmaStar System Tool 生成 c code gamma table。



SigmaStar System Tool 可以生成如下格式的 c code gamma tale

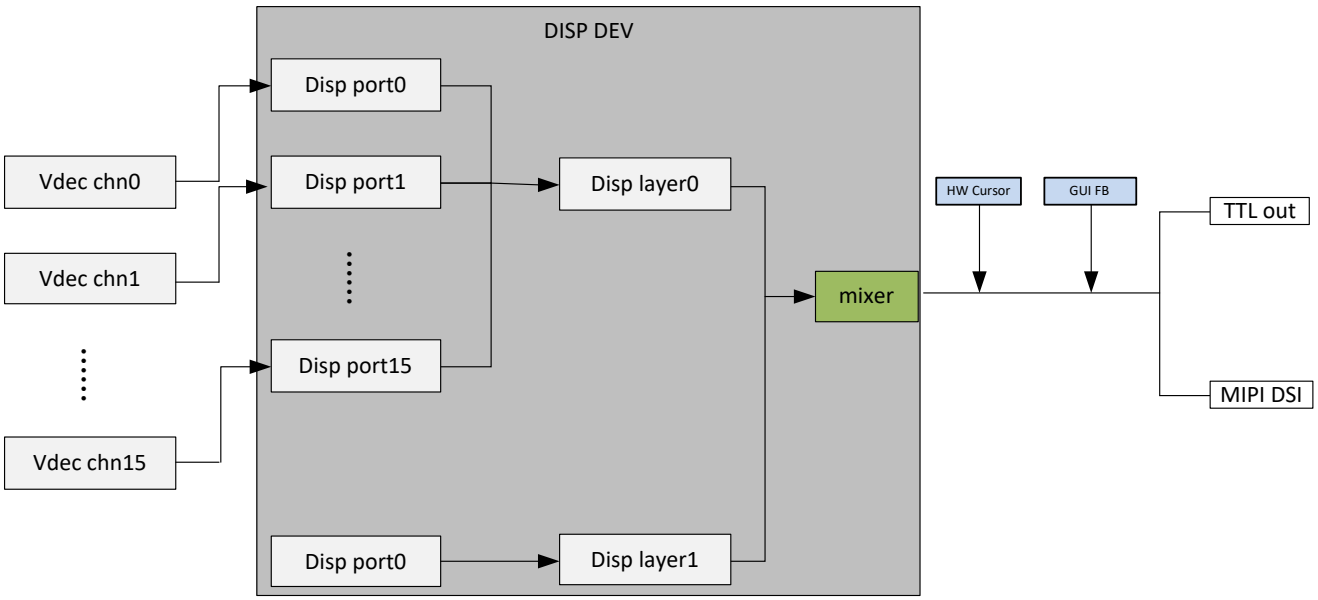
```
MI_U8 tnormalGammaR[] =
{
    0x00,0x07,0x0F,0x17,
    0x1F,0x27,0x2F,0x37,
```

```
¦   0x3F,0x47,0x4F,0x57,
¦   0x5F,0x67,0x6F,0x77,
¦   0x7F,0x87,0x8F,0x97,
¦   0x9F,0xA7,0xAF,0xB7,
¦   0xBF,0xC7,0xCF,0xD7,
¦   0xDF,0xE7,0xEF,0xF7,
¦   0xFF
};

MI_U8 tnormalGammaG[] =
{
¦   0x00,0x07,0x0F,0x17,
¦   0x1F,0x27,0x2F,0x37,
¦   0x3F,0x47,0x4F,0x57,
¦   0x5F,0x67,0x6F,0x77,
    0x7F,0x87,0x8F,0x97,
¦   0x9F,0xA7,0xAF,0xB7,
¦   0xBF,0xC7,0xCF,0xD7,
¦   0xDF,0xE7,0xEF,0xF7,
¦   0xFF
};

MI_U8 tnormalGammaB[] =
{
¦   0x00,0x07,0x0F,0x17,
¦   0x1F,0x27,0x2F,0x37,
¦   0x3F,0x47,0x4F,0x57,
¦   0x5F,0x67,0x6F,0x77,
¦   0x7F,0x87,0x8F,0x97,
¦   0x9F,0xA7,0xAF,0xB7,
¦   0xBF,0xC7,0xCF,0xD7,
¦   0xDF,0xE7,0xEF,0xF7,
¦   0xFF
};

调用 DISP API 设定 GAMMA
MI_DISP_GammaParam_t stGammaParam;
stGammaParam.bEn = TRUE;
stGammaParam.u16EntryNum = sizeof(tnormalGammaR);
stGammaParam.pu8ColorR = tnormalGammaR;
stGammaParam.pu8ColorG = tnormalGammaG;
stGammaParam.pu8ColorB = tnormalGammaB;
MI_DISP_DeviceSetGammaParam(devid, &stGammaParam);
```

3.8 输出设备

DISP 支持 LCD 以及 HDMI/VGA，LCD 和 HDMI/VGA 不能同时显示，HDMI/VGA 可以同时输出显示。LCD 支持 TTL PANEL 和 MIPI PANEL。本节主要介绍点亮 TTL 或 MIPI PANEL 的相关操作。

DISP 视频显示数据流程框图如下：

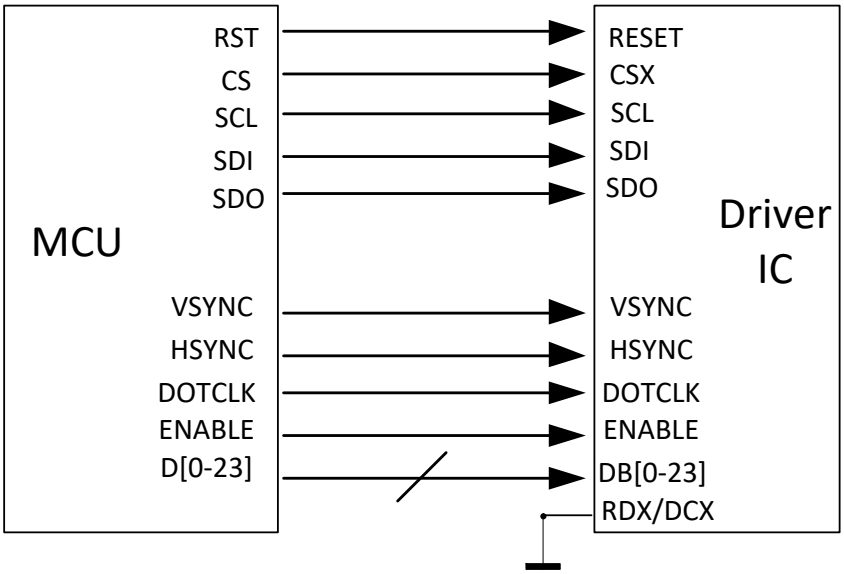


DISP video layer0 的 16 个 input port 可以绑定 vdec 的 16 个 channel 做 16 windows 拼图显示，DISP video layer1 的 port 可以做 PIP 使用，即最终 video layer1 和 video layer0 做 mixer，video layer1 显示优先级高于 video layer0。DISP 硬件拼图后根据用户选择的输出设备，将拼图后的视频数据按照一定的时序通过 TTL 接口送出 RGB data；如果是 MIPI 接口，支持 sync mode、sync event mode、burst mode 等 MIPI 标准传输模式。

GUI FB/Cursor 有独立的 HW enginer，需要操作 FB device 来进行 UI 的绘制，这部分不做详细介绍。

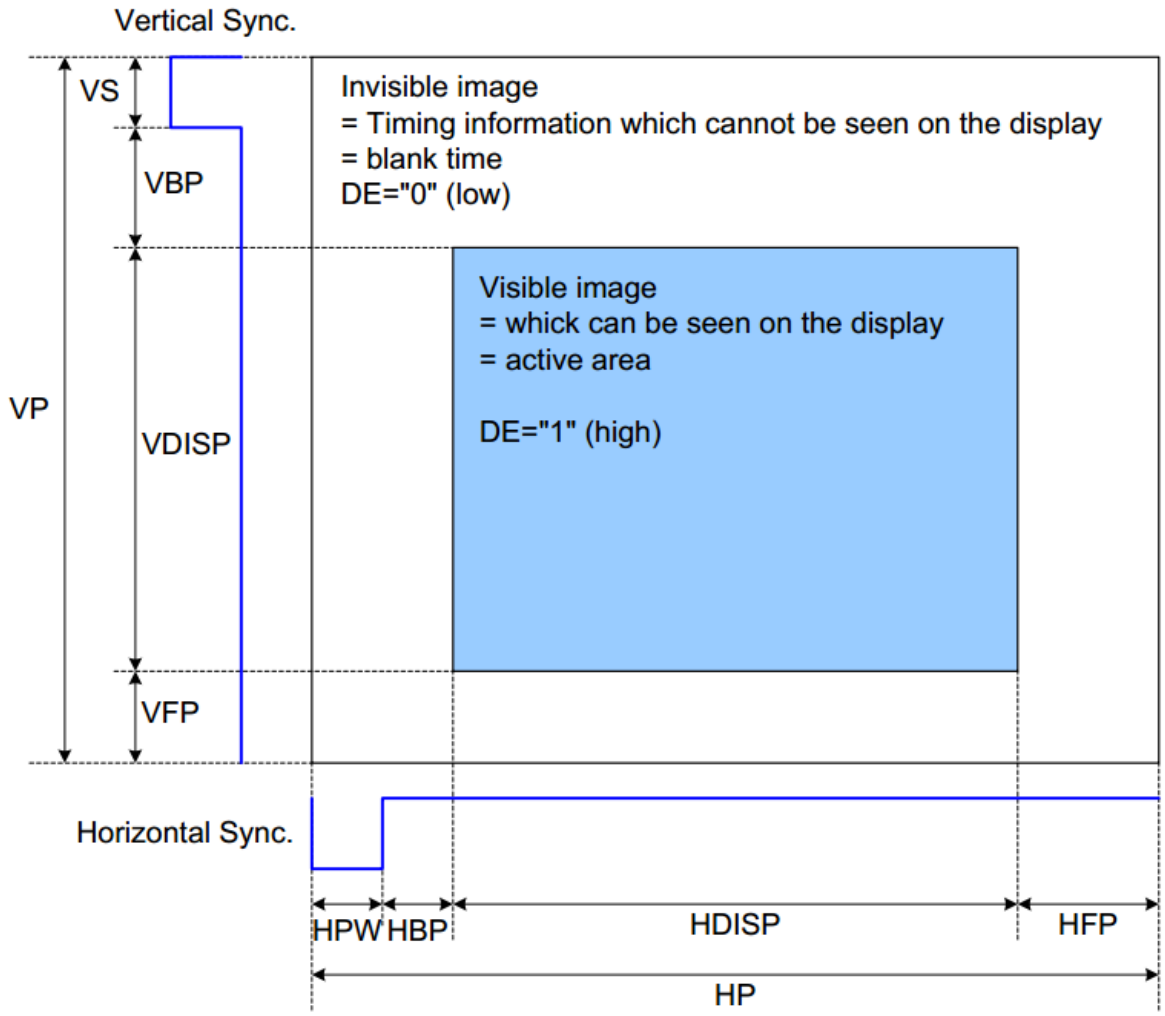
3.8.1 TTL

Parallel RGB Interface 有两种，DE mode 和 HV mode，当使能 DE 的时候，会使用 VSYNC、HSYNC、DOTCLK、DE、D[0-23]这些 pin，当使能 HV mode 的时候，会使用 VSYNC、HSYNC、DOTCLK、D[0-23]。



一些 panel driver IC 需要对其初始化，其实就是对其内部 register 的设定，一般会通过 SPI 或者 IIC 接口来通信，初始化需要的 cmd 和 data 一般由屏厂提供，发送时的数据格式需要参考 panel datasheet。

使用 Parallel RGB Interface 的 panel 需要 HSYNC、VSYNC、DOTCLK 来作为同步信号，RGB data 只在 timing 的特定区间有效。



行场信号中的 blanking 区间是不可见区域，只有 active 区间的 RGB data 才会最终显示出来，不同的 panel Driver IC 对于 blanking 区间的要求不同。

行信号中:

Htotal = HSYNC + HBP + HFP + H Active

场信号中：

Vtotal = VSYNC + VBP + VFP + V Active

最后计算像素时钟频率：

Pixel CLK = Htotal * Vtotal * fps

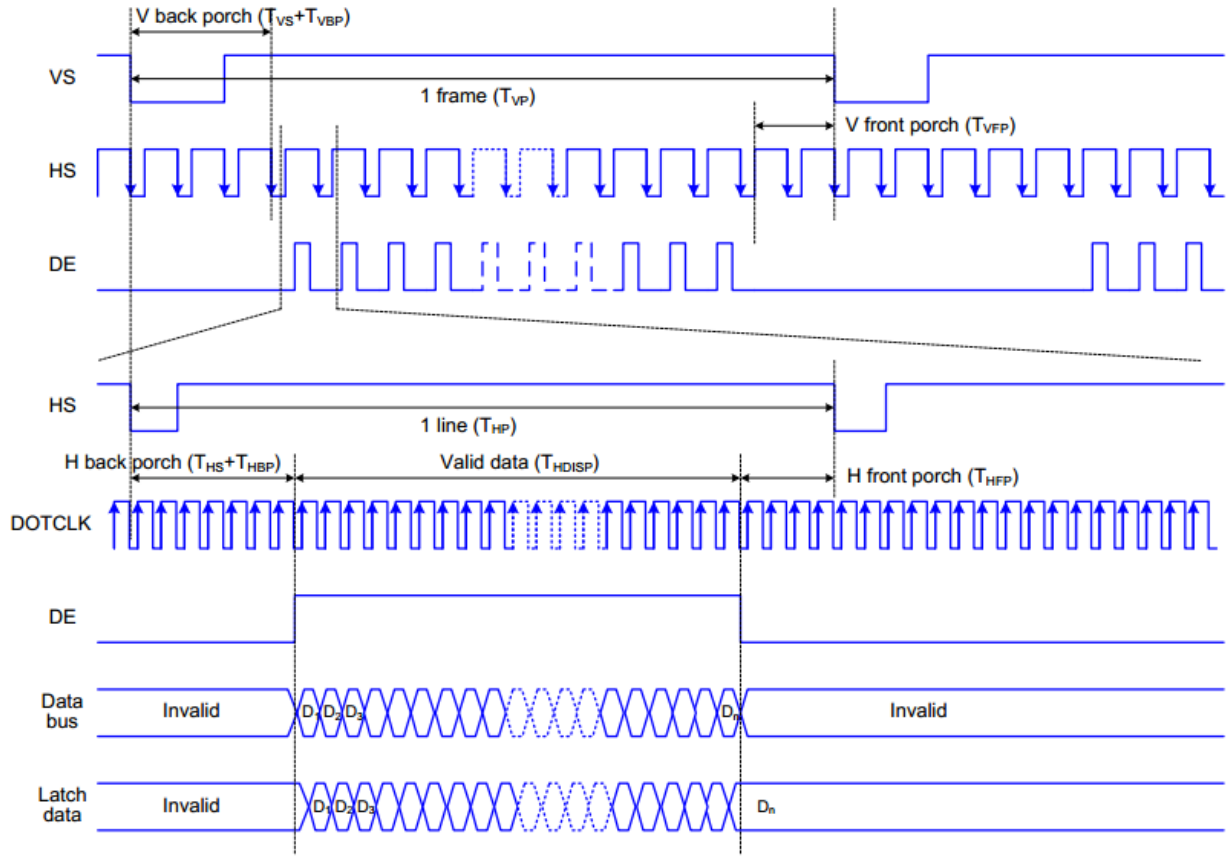
屏参中时序相关的配置主要是调整行信号中 HSYNC、HBP、HFP 和场信号中 VSYNC、VBP、VFP。panel spec 中会提供 blanking 区间各部分的长度要求。

blanking 区间是一个可调范围，最终计算出的 pixel clk 也是在一个范围内。

主控芯片一般会有时钟频率的限制，所以在点一个新的 panel 时，应先计算 panel 的 pixel clk，如果将要点的 panel 满足像素时钟频率的要求，则一般可以点起来。

Parallel RGB Interface Timing

The timing chart of RGB interface DE mode is shown as follows.



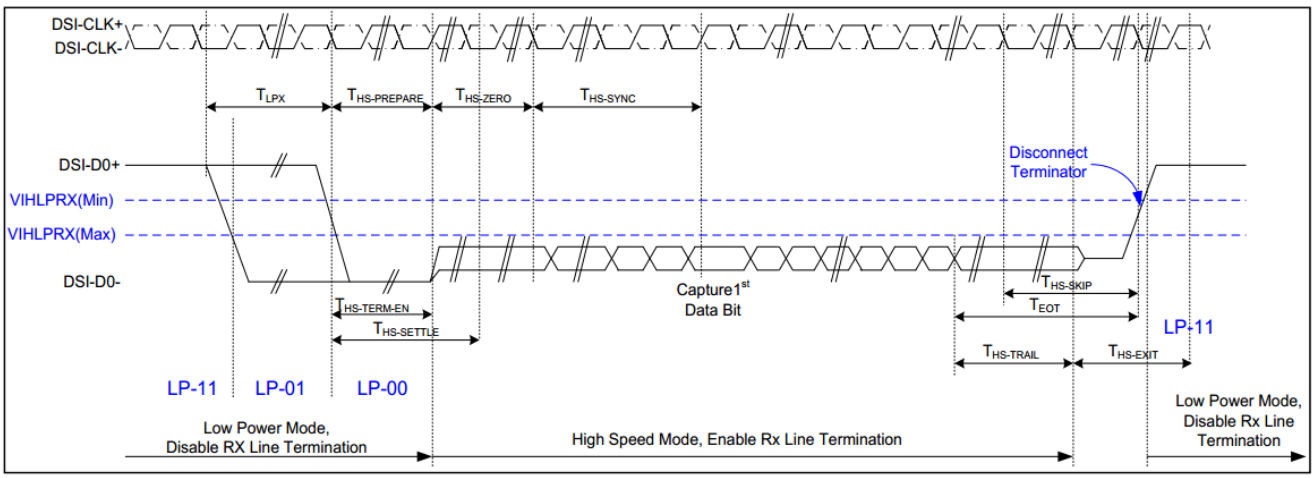
Note: The setting of front porch and back porch in host must match that in IC as this mode.

3.8.2 MIPI

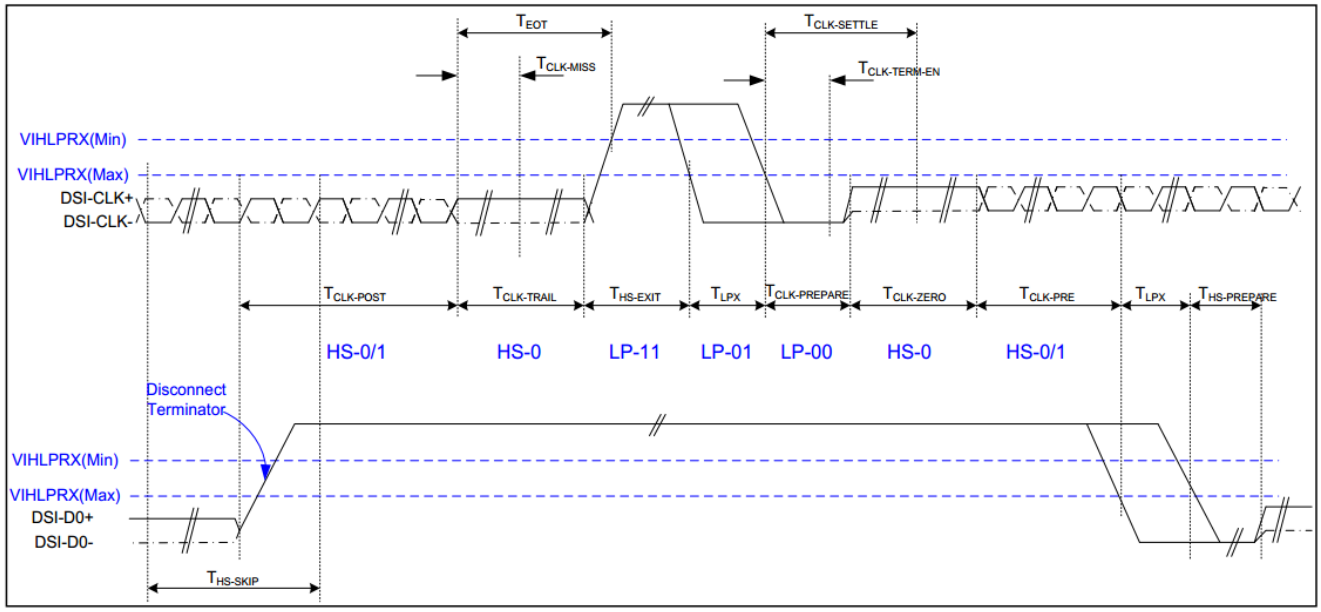
MIPI DSI 规格：

- 1-4 data lanes, 1 clock lane
- Level:
  - LP: 0~12V
  - HS: 100~300mV,
- HS: 80Mbps ~ 1.5Gbps/lane
- Pixel format:
  - 16 bpp (5,6,5 RGB) each pixel using two bytes
  - 18 bpp (6,6,6 RGB) packed
  - 18 bpp (6, 6, 6 RGB) loosely packed into three bytes
  - 24 bpp (8, 8, 8 RGB), each pixel using three bytes
- video mode: BURST_MODE/SYNC_EVENT/SYNC_PULSE
- data/clk chn swap
- data/clk chn P/N swap
- data clk skew adjustment

MIPI DPHY timing chart:



Data lanes-Low Power Mode to/from High Speed Mode Timing



Clock lanes- High Speed Mode to/from Low Power Mode Timing

Configure HS Timing Parameter:

- HS_TRAIL/HS_EXIT /HS_PRPR/HS_ZERO/CLK_PRPR/CLK_ZERO/CLK_POST/CLK_TRAIL

	Timing specification
$T_{HS-PREPARE}$ + $T_{HS-zero}$	145ns+10*UI
$T_{HS-PREPARE}$	40ns+4*UI ~ 85ns +6*UI
$T_{HS-ZERO}$	>60ns+4*UI ~105ns+6*UI

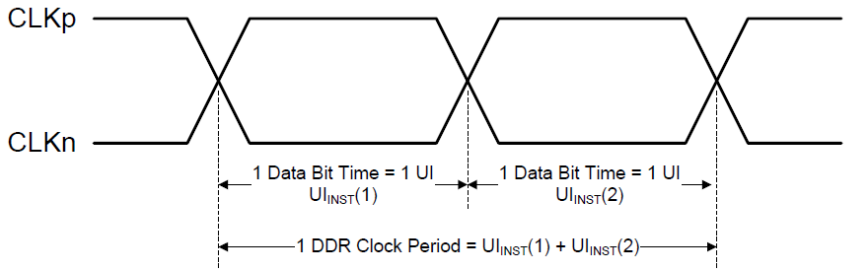
UI 表示时间间隔，等于时钟通道上任何 HS 状态的持续时间.

UI 的计算方法：

H_Total = HACT+HPW+HBP+HFP

V_Total = VACT+VPW+VBP+VFP

BitsPerPixel=24(RGB888)/18(RGB666)/16(RGB565)
Bitrate = (H_Total)*(V_Total)*FPS* BitsPerPixel/lane number
UI = 1/Bitrate



DSI clock channel timing

如果 Bitrate = 750Mps，UI = 1/Bitrate = 1.333ns，那么 HS timing parameter 计算结果为：

	Timing specification	**Absolute time**	**DA_HS_PREP value** （Absolute time/(8*UI)）
$T_{HS-PREPARE}$ + $T_{HS-zero}$	145ns+10*UI	> 158.33 ns	> 15
$T_{HS-PREPARE}$	40ns+4*UI ~ 85ns +6*UI	45.32 ~ 92.98 ns	5 ~ 8
$T_{HS-zero}$	>60ns+4*UI ~105ns+6*UI	> 112.98~ 65.32 ns	10 ~ 7

Configure LP Timing Parameter:

- CONT_DET/LPX/TA_GET/TA_SURE/TA_GO
  low power parameters 建议使用提供的默认值

4. 点屏配置

✧ **配置 DISP device**

MI_S32 MI_DISP_SetPubAttr(MI_DISP_DEV DispDev, const MI_DISP_PubAttr_t *pstPubAttr)
配置 DISP 设备属性

参数			描述
DispDev			DISP device ID
pstPubAttr	u32BgColor		背景色
	eIntfType		接口类型（点屏使用 E_MI_DISP_INTF_LCD）
	eIntfSync		输出 timing（点屏使用 E_MI_DISP_OUTPUT_USER）
	stSyncInfo	u16Vact	等于屏参中 u16Height
		u16Vbb	等于屏参中 u16VSyncBackPorch
		u16Vfb	等于屏参中 u16VTotal-(u16VSyncWidth+u16Height+u16VSyncBackPorch)
		u16Hact	等于屏参中 u16Width
		u16Hbb	等于屏参中 u16HSyncBackPorch
		u16Hfb	等于屏参中 u16HTotal-(u16HSyncWidth+u16Width+u16HSyncBackPorch)
		u16Hpw	等于屏参中 u16HSyncWidth
		u16Vpw	等于屏参中 u16VSyncWidth
		u32FrameRate	等于屏参中 u16DCLK*1000000/(u16HTotal*u16VTotal)

表中未列参数默认填 0。

MI_S32 MI_DISP_Enable(MI_DISP_DEV DispDev)
使能 DISP 设备

参数	描述
DispDev	DISP device ID

✧ **配置 DISP video layer**

MI_S32 MI_DISP_SetVideoLayerAttr(MI_DISP_LAYER DispLayer, const MI_DISP_VideoLayerAttr_t *pstLayerAttr)
设置 DISP video layer 属性

参数			描述
DispLayer			DISP video layer ID
pstLayerAttr	stVidLayerDispWin	u16X	0
		u16Y	0

		u16Width	等于屏参中 u16Width
		u16Height	等于屏参中 u16Height
	stVidLayerSize	u16Width	等于屏参中 u16Width
		u16Height	等于屏参中 u16Height
	ePixFormat	E_MI_SYS_PIXEL_FRAME_YUV_SEMIPLANAR_420	

MI_S32 MI_DISP_EnableVideoLayer(MI_DISP_LAYER DispLayer)
使能 DISP video layer

参数	描述
DispLayer	DISP video layer ID

✧ **配置 DISP input port**

MI_S32 MI_DISP_SetInputPortAttr(MI_DISP_LAYER DispLayer, MI_DISP_INPUTPORT LayerInputPort, const MI_DISP_InputPortAttr_t *pstInputPortAttr)
设置 DISP input port 属性

参数			描述
DispLayer			DISP video layer ID
LayerInputPort			DISP input port ID
pstInputPortAttr	stDispWin	u16X	Input port 显示位置的 x offset
		u16Y	Input port 显示位置的 y offset
		u16Width	Input port 显示 width(没有 scaling，则等于 u16SrcWidth)
		u16Height	Input port 显示 height(没有 scaling，则等于 u16SrcHeight)
	u16SrcWidth		Video source width
	u16SrcHeight		Video source height

MI_S32 MI_DISP_EnableInputPort(MI_DISP_LAYER DispLayer, MI_DISP_INPUTPORT LayerInputPort)
使能 DISP input port

参数	描述
DispLayer	DISP video layer ID
LayerInputPort	DISP input port ID

✧ **配置屏参**

MI_S32 MI_PANEL_Init(MI_PANEL_LinkType_e eLinkType)
设置 PANEL 类型

参数	描述
eLinkType	E_MI_PNL_LINK_TTL

	E_MI_PNL_LINK_MIPI_DSI

MI_S32 MI_PANEL_SetPanelParam(MI_PANEL_ParamConfig_t *pstParamCfg)

设置屏参

参数		描述
pstParamCfg	pPanelName	屏的型号
	u8Dither	1:enable Dither 0:disable Dither
	eLinkType	接口类型 E_MI_PNL_LINK_TTL E_MI_PNL_LINK_MIPI_DSI
	u8InvDCLK	Pixel clk 极性反转
	u8InvDE	DE 极性反转
	u8InvHSync	Hsync 极性反转
	u8InvVSync	Vsync 极性反转
	u16HSyncWidth	行同步信号脉宽
	u16HSyncBackPorch	行同步信号后肩
	u16VSyncWidth	场同步信号脉宽
	u16VSyncBackPorch	场同步信号后肩
	u16HStart	u16HSyncWidth+16HSyncBackPorch
	u16VStart	u16VSyncWidth+ u16VSyncBackPorch
	u16Width	行有效像素点数
	u16Height	场有效行数
	u16HTotal	u16HSyncWidth+16HSyncBackPorch+HsyncFrontPorch
	u16VTotal	u16VSyncWidth+ u16VSyncBackPorch+VsyncFrontPorch
	u16DCLK	u16HTotal*u16VTotal*fps
	u16SpreadSpectrumStep	时钟延展幅度调制（详见展频计算表）
	u16SpreadSpectrumSpan	时钟延展频率调制（详见展频计算表）
	eOutputFormatBitMode	输出 pixel format
	u8SwapOdd_RG	Swap Channel R 0:default 1:select B 2:select G 3:select R
	u8SwapEven_RG	Swap Channel G 0:default 1:select B 2:select G

		3:select R
	u8SwapOdd_GB	Swap Channel B 0:default 1:select B 2:select G 3:select R
	u8SwapEven_GB	Swap Rgb MSB/LSB 0:disable M/L swap 1:enable M/L swap
	eCh0	Chn0 lane selection(default:2) 0:select lane0 1:select lane1 2:select lane2 3:select lane3 4:select lane4 Selected lane 将作为 clk lane 输出
	eCh1	Chn1 lane selection(default:4) 0:select lane0 1:select lane1 2:select lane2 3:select lane3 4:select lane4
	eCh2	Chn2 lane selection(default:3) 0:selec1t lane0 1:select lane1 2:select lane2 3:select lane3 4:select lane4
	eCh3	Chn3 lane selection(default:1) 0:select lane0 1:select lane1 2:select lane2 3:select lane3 4:select lane4
	eCh4	Chn4 lane selection(default:0) 0:select lane0 1:select lane1 2:select lane2 3:select lane3 4:select lane4

表中未列参数默认填 0

如果是 MIPI panel，还需要对 MIPI DSI 进行配置，TTL panel 只配置屏参即可。

MI_S32 MI_PANEL_SetMipiDsiConfig(MI_PANEL_MipiDsiConfig_t *pstMipiDsiCfg)

参数		描述
pstMipiDsiCfg	u8HsTrail	Default: 0x05 60+4UI ~ MAX
	u8HsPrpr	Default: 0x05 40+4UI ~ 85+6UI
	u8HsZero	Default: 0x05 105+6UI ~ MAX
	u8ClkHsPrpr	Default: 0x05 38ns ~ 95ns
	u8ClkHsExit	Default: 0x05 100ns ~ max
	u8ClkTrail	Default: 0x05 60ns ~ max
	u8ClkZero	Default: 0x05 300ns-CLK_HS_PRPR
	u8ClkHsPost	Default: 0x05 60+52UI ~ max
	u8DaHsExit	Default: 0x05 100ns ~ max
	u8ContDet	Default:0
	u8Lpx	Default:16
	u8TaGet	Default:26 5LPX
	u8TaSure	Default:24 1LPX ~ 2LPX
	u8TaGo	Default:50 4LPX
	u16Hactive	Follow 屏参设定
	u16Hpw	
	u16Hbp	
	u16Hfp	
	u16Vactive	
	u16Vpw	
	u16Vbp	
	u16Vfp	
	u16Bllp	0

	u16Fps	Default:60
	enLaneNum	E_MI_PNL_MIPI_DSI_LANE_1 E_MI_PNL_MIPI_DSI_LANE_2 E_MI_PNL_MIPI_DSI_LANE_3 E_MI_PNL_MIPI_DSI_LANE_4
	enformat	E_MI_PNL_MIPI_DSI_RGB565 E_MI_PNL_MIPI_DSI_RGB666 E_MI_PNL_MIPI_DSI_LOOSELY_RGB666 E_MI_PNL_MIPI_DSI_RGB888
	enCtrl	E_MI_PNL_MIPI_DSI_CMD_MODE E_MI_PNL_MIPI_DSI_SYNC_PULSE E_MI_PNL_MIPI_DSI_SYNC_EVENT E_MI_PNL_MIPI_DSI_BURST_MODE

	pu8CmdBuf	Mipi panel 初始化 cmd buff 指针
		cmd buff 格式：
		cmd_buff[]=
		{
		Cmd, parameter cnt, parameter0, parameter1, … ,
		Cmd, parameter cnt, parameter0, parameter1, … ,
		Cmd, parameter cnt, parameter0, parameter1, … ,
		……
		}

	u32CmdBufSize	Cmd buff size = sizeof(cmd_buff)
	u16DataClkSkew	Date/clk 相位 范围：7-15
	u8PolCh0	Chn0 极性 0：default 1：positive 2：negative
	u8PolCh1	Chn1 极性 0：default 1：positive 2：negative
	u8PolCh2	Chn2 极性 0：default 1：positive 2：negative
	u8PolCh3	Chn3 极性 0：default 1：positive 2：negative

	u8PolCh4	Chn4 极性 0：default 1：positive 2：negative

✧ 初始化 PANEL

一些 panel 需要 cmd 来初始化屏驱。MIPI panel 的初始化 cmd 是通过 MIPI data lane0 来传输，cmd buff 填写参考上节说明；TTL panel 一般是通过 SPI 来发送 cmd/parameters 给屏驱，初始化需要的 cmd 要参考屏的数据手册，根据屏的初始化时序要求来下相应的初始化命令。

如果是采用 SPI 初始化的 panel，SDK 中有提供 SPI 发送数据的 API。

MI_S32 MI_PANEL_GPIO_Init(MI_PANEL_GpioConfig_t *pstGpioCfg)

初始化 panel 复位、panel 背光、panel enable 以及 SPI 使用到的引脚。

参数		描述
pstGpioCfg	u16GpioBL	Panel 背光
	u16GpioRST	Panel reset
	u16GpioSCL	SPI
	u16GpioSDO	
	u16GpioCS	
	u16GpioEN	Panel enable

MI_S32 MI_PANEL_SetGpioStatus(MI_U16 u16GpioNum, MI_BOOL bValue)

拉高/低某一个 GPIO

参数	描述
u16GpioNum	Gpio software num
bValue	0:low 1:high

MI_S32 MI_PANEL_SetCmd(MI_U32 u32Value, MI_U8 u8Bits)

通过 SPI 发送数据

参数	描述
u32Value	需要发送的数据
u8Bits	该数据的长度（单位 bit）

✧ video 显示

DISP/PANEL 初始化完成，送需要显示的 video data 给 DISP，一种方式是直接送 YUV data 到 DISP，第二种方式是 DISP 绑定其他可以输出 YUV data 的模块，前提是可以输出 YUV data 的模块已经初始化完成，并且有数据输出。

第一种方式需要用到的相关 API：

MI_S32 MI_SYS_ChnInputPortGetBuf (MI_SYS_ChnPort_t *pstChnPort,MI_SYS_BufConf_t *pstBufConf, MI_SYS_BufInfo_t *pstBufInfo, MI_SYS_BUF_HANDLE *pBufHandle, MI_S32 s32TimeOutMs)

MI_S32 MI_SYS_ChnInputPortPutBuf (MI_SYS_BUF_HANDLE bufHandle, MI_SYS_BufInfo_t *pstBufInfo, MI_BOOL bDropBuf)

第二种方式需要用到的相关 API：

MI_S32 MI_SYS_BindChnPort(MI_SYS_ChnPort_t *pstSrcChnPort, MI_SYS_ChnPort_t *pstDstChnPort,MI_U32 u32SrcFrmrate, MI_U32 u32DstFrmrate)

使用方法参考 MI SYS 相关文档。

5. 参考 DEMO

sdk/verify/feature/disp/disp_ut.c

使用方法：

A. 编译

- 修改 project/release/customer_tailor/nvr_default.mk，添加 verify_disp:=enable
- 修改 disp_ut.c，include 对应的屏参头文件
- cd sdk/verify/feature;
- make disp_clean;make disp
- cp disp/prog 到执行目录

B. 运行

./prog --interface lcd -l ttl -f ./YUV420SP_800_480.yuv -t yuv420 -n 1 -i 800_480 -c 0_0_800_480 -o 0_0_800_480

--interface //输出接口，点屏用 lcd
-l //panel 类型，ttl 或者 mipi
-f //文件路径
-t //pixel format
-n //chn num
-i //input size
-c //disp crop parameters
-o //disp show size

6. 调试方法和常见问题

调试手段：

a) procfs

cat /proc/mi_modules/mi_disp/mi_disp0

```
============================== Private DISP0 Info ==============================
 DevStatus          IrqNum          IrqCnt         BgColor
         1              57            1322          800080
 Interface       DevTiming       CscMatrix            Luma        Contrast             Hue      Saturation       Sharpness
       LCD         UNKNOWN               3              50              50              50              50               0
-------------------------------- Layer Info --------------------------------
   LayerId     BindedDevID      LayerWidth     LayerHeight
         0               0            1024             600
   LayerId    LayDispWidth   LayDispHeight      Toleration      rotatemode
         0            1024             600               0            NONE
--------------------------------- Layer0 InputPort Info ---------------------------------
    PortId      enable      Status       src_w       src_h      crop_x      crop_y      crop_w      crop_h      show_x      show_y      show_w      show_h
         0           1           0         200         120           0           0         200         120           0           0        1024         600
         1           0           0           0           0           0           0           0           0           0           0           0           0
         2           0           0           0           0           0           0           0           0           0           0           0           0
         3           0           0           0           0           0           0           0           0           0           0           0           0
         4           0           0           0           0           0           0           0           0           0           0           0           0
         5           0           0           0           0           0           0           0           0           0           0           0           0
         6           0           0           0           0           0           0           0           0           0           0           0           0
         7           0           0           0           0           0           0           0           0           0           0           0           0
         8           0           0           0           0           0           0           0           0           0           0           0           0
         9           0           0           0           0           0           0           0           0           0           0           0           0
        10           0           0           0           0           0           0           0           0           0           0           0           0
        11           0           0           0           0           0           0           0           0           0           0           0           0
        12           0           0           0           0           0           0           0           0           0           0           0           0
        13           0           0           0           0           0           0           0           0           0           0           0           0
        14           0           0           0           0           0           0           0           0           0           0           0           0
        15           0           0           0           0           0           0           0           0           0           0           0           0
    PortId      RecvBufCnt       RecvBuf_W       RecvBuf_H       Content_W       Content_H   RecvBufStride          PixFmt
         0             447             200             120             200             120             224 semiplanar_420
         1               0               0               0               0               0               0    yuv422_yuyv
         2               0               0               0               0               0               0    yuv422_yuyv
         3               0               0               0               0               0               0    yuv422_yuyv
         4               0               0               0               0               0               0    yuv422_yuyv
         5               0               0               0               0               0               0    yuv422_yuyv
         6               0               0               0               0               0               0    yuv422_yuyv
         7               0               0               0               0               0               0    yuv422_yuyv
         8               0               0               0               0               0               0    yuv422_yuyv
         9               0               0               0               0               0               0    yuv422_yuyv
        10               0               0               0               0               0               0    yuv422_yuyv
        11               0               0               0               0               0               0    yuv422_yuyv
        12               0               0               0               0               0               0    yuv422_yuyv
        13               0               0               0               0               0               0    yuv422_yuyv
        14               0               0               0               0               0               0    yuv422_yuyv
        15               0               0               0               0               0               0    yuv422_yuyv
    PortId    OnScreenTask     PendingTak    DropTaskCnt   LastDropTask  bClearAllTask             fps
         0        c4b2a038         (null)              0         (null)              0              21
         1          (null)         (null)              0         (null)              0               0
         2          (null)         (null)              0         (null)              0               0
         3          (null)         (null)              0         (null)              0               0
         4          (null)         (null)              0         (null)              0               0
         5          (null)         (null)              0         (null)              0               0
         6          (null)         (null)              0         (null)              0               0
         7          (null)         (null)              0         (null)              0               0
         8          (null)         (null)              0         (null)              0               0
         9          (null)         (null)              0         (null)              0               0
```

DISP procfs 主要用来分析数据流是否正常，DISP 相关设定是否合理和 DISP 工作状态。

数据流相关：

RecvBufCnt：分析 DISP 是否拿到 input buff

RecvBuf_W：input buff width

RecvBuf_H：input buff height

Content_W：input buff 有效 width

Content_H：input buff 有效 height

RecvBufStride：input buff stride

PixFmt：input buff pixel format

功能性设定：

crop_x/crop_y/crop_w/crop_h: input port crop parameters

show_x/show_y/show_w/show_h: display input port show size

rotatemode: rotate mode

CscMatrix/Luma/Contrast/Hue/Saturation/Sharpness: PQ

状态相关：

DevStatus：DISP device 使能状态

IrqCnt：DISP 硬件中断次数

enable：DISP input port 使能状态

DISP 没有正常显示的情况下，首先检查以上三个部分是否正常。

cat /proc/mi_modules/mi_panel/mi_panel0

```
------------------------------------ PANEL Dev0 Info -----------------------------------
  LVDS_POL        LVDS_CH      LINK_TYPE        TI_MODE
         0              0            TTL              1
    SW_ODD       SW_EVEN     SW_ODD_RB     SW_EVEN_RB
         0              0              0              0
   H_Total        V_Total          Width         Height        H_Start        V_Start
      1344            635           1024            600             98             27
       hbp            hspw            hfp            vbp            vspw            vfp
        46             48            226             23              4              8
      DClk      FrameRate       INV_DCLK         INV_DE      InvHSync      InvVSync
        51              0              0              0              0              0
SSC_Enable       SSC_Span       SSC_Step        TI_BIT         Format        chnswap
   disable            192             25          8BIT           8BIT      (0,1,2,3,4)
```

PANEL procfs 主要用来分析屏参设定是否生效。

b) register

跟显示相关需要关注的寄存器：

Bank	Addr	Description
101e	0d	[bit8-bit11] TTL mode 0001: SSR201/202 1100: SSR623 [bit12-bit13] MIPI TX mode 01: 4 lane
1038	54	[bit0-bit3] clk_mop [bit0] disable clk [bit1] inv clk [bit2-bit3] select clk source 00:320Mhz 01:384Mhz 10:288Mhz 11:clk_miu
	53	[bit0-bit3] clk_disp_432 [bit0] disable clk [bit1] inv clk [bit2-bit3] select clk source 00: 432Mhz [bit8-bit11] clk_disp_216 [bit8] disable clk [bit9] inv clk [bit10-bit11] select clk source

		00: 216Mhz 01: 108Mhz
	63	[bit0-bit5] clk_sc_pixel [bit0] disable clk [bit1] inv clk [bit2-bit5] select clk source 0000:240Mhz 0001:216Mhz 0010:192Mhz 0011:172Mhz 0100:144Mhz 0101:123Mhz 0110:108Mhz 0111:86Mhz 1000:72Mhz 1001:54Mhz 1010:lpll_clk
	6f	[bit0-bit4] clk_mipi_tx_dsi [bit0] disable clk [bit1] inv clk [bit2-bit4] select clk source 000:lpll_clk 001:160Mhz 010:144Mhz 011:108Mhz 100:216Mhz 101:240Mhz
1128	07	[bit0-bit3] pattern gen [bit0] select source of mace 0: from scaling patgen 1: from external video source [bit1-bit3] select pattern mode 000: 1-pix gray ramp 001: 16-pix gray ramp 010: 32-pix gray ramp 011: 64-pix gray ramp 100: 16-pix gray stick 101: 16-pix colorbar 110: 32-pix colorbar 111: 64-pix colorbar
1129	11	[bit0-bit12] h total
	12	[bit0-bit12] v total
	13	[bit0-bit12] hsync start
	14	[bit0-bit12] hsync end
	15	[bit0-bit12] vsync start
	16	[bit0-bit12] vsync end
	17	[bit0-bit12] H frame de start
	18	[bit0-bit12] H frame de end
	19	[bit0-bit12] V frame de start
	1a	[bit0-bit12] V frame de end
	1b	[bit0-bit15] no signal color [bit0-bit4] B channel

		[bit5-bit9] G channel [bit10-bit14] R channel [bit15] Forced to show no signal color
	7e	[bit0-bit5] rgb swap [bit0-bit1] swap for B channel [bit2-bit3] swap for G channel [bit4-bit5] swap for R channel [bit6-bit7] rgb mode 00:rgb 888 01:rgb 666 10:rgb 565-1 11:rgb 565-2 [bit8] MSB/LSB swap
1406	00	[bit0] Gwin0 enable
	10	[bit0] Gwin1 enable
	20	[bit0] Gwin2 enable
	30	[bit0] Gwin3 enable
	40	[bit0] Gwin4 enable
	50	[bit0] Gwin5 enable
	60	[bit0] Gwin6 enable
	70	[bit0] Gwin7 enable
1407	00	[bit0] Gwin8 enable
	10	[bit0] Gwin9 enable
	20	[bit0] Gwin10 enable
	30	[bit0] Gwin11 enable
	40	[bit0] Gwin12 enable
	50	[bit0] Gwin13 enable
	60	[bit0] Gwin14 enable
	70	[bit0] Gwin15 enable
1033	4e	[bit0-bit15] ssc [bit0-bit11] step [bit15] enable ssc
	4f	[bit0-bit13] ssc span

常见问题：

1 背光可以亮，但是没有画面

1.1 检查屏参是否满足 panel spec 要求

1.2 数据流是否正常，是否有 input buff 输入

1.3 DISP device/layer/input port 相关设定是否合理

1.4 panel init 是否成功，需要 SPI 初始化的 TTL panel 可以量测 SPI 信号是否正常，initial cmd 是否有正确传输

1.5 检查 Hsync/Vsync/DE/Dclk 极性是否满足 panel spec

1.6 MIPI panel 检查 clk lane/data lane 线序和软件设定是否一致

1.7 查看 register BK101e addr 0d 是否有切对 TTL mode/MIPI mode

1.8 检查 clk 相关 register 是否都有打开

1.9 MOP gwin 是否有打开

1.10 pattern gen 是否可以正常输出

2 分屏现象

如果是 H 方向，尝试加大 Hsync Front porch 和 Htotal，Hsync 和 Hsync back porch 保持不变

如果是 V 方向，尝试加大 Vsync Front porch 和 Vtotal，Vsync 和 Vsync back porch 保持不变

3 色偏

色偏

正常

3.1 检查 R/G/B 单色是否输出正常，如果是 TTL panel，可以关掉 3x3 matrix(BK1129_30 bit0=0,BK1129_3c bit0=0)，利用 BK1129_1b 输出 RGB 单色并量测 R/G/B channel 每根是否都有正常输出。

3.2 调整 VBP，VPW/VFP 维持不变

3.3 检查是否需要做 R/G/B channel swap 或者 RGB M/L swap。

7. TTL/MIPI 硬件连接

1、如下所示，芯片 TTL 输出默认硬件连接顺序

Pin Number	PAD_Name	reg_ttl_mode				
		1				
		28pin	Default			
Pin56	PAD_TTL0	TTL_DOUT[0]	R0	**G0**	**B0**	**R7/G7/B7**
Pin57	PAD_TTL1	TTL_DOUT[1]	R1	**G1**	**B1**	**R6/G6/B6**
Pin58	PAD_TTL2	TTL_DOUT[2]	R2	**G2**	**B2**	**R5/G5/B5**
Pin59	PAD_TTL3	TTL_DOUT[3]	R3	**G3**	**B3**	**R4/G4/B4**
Pin60	PAD_TTL4	TTL_DOUT[4]	R4	**G4**	**B4**	**R3/G3/B3**
Pin61	PAD_TTL5	TTL_DOUT[5]	R5	**G5**	**B5**	**R2/G2/B2**
Pin65	PAD_TTL6	TTL_DOUT[6]	R6	**G6**	**B6**	**R1/G1/B1**
Pin66	PAD_TTL7	TTL_DOUT[7]	R7	**G7**	**B7**	**R0/G0/B0**
Pin67	PAD_TTL8	TTL_DOUT[8]	**G0**	**B0**	R0	
Pin68	PAD_TTL9	TTL_DOUT[9]	**G1**	**B1**	R1	
Pin69	PAD_TTL10	TTL_DOUT[10]	**G2**	**B2**	R2	
Pin70	PAD_TTL11	TTL_DOUT[11]	**G3**	**B3**	R3	
Pin71	PAD_TTL12	TTL_DOUT[12]	**G4**	**B4**	R4	
Pin72	PAD_TTL13	TTL_DOUT[13]	**G5**	**B5**	R5	
Pin73	PAD_TTL14	TTL_DOUT[14]	**G6**	**B6**	R6	
Pin74	PAD_TTL15	TTL_DOUT[15]	**G7**	B7	R7	
Pin79	PAD_TTL16	TTL_DOUT[16]	**B0**	R0	**G0**	
Pin80	PAD_TTL17	TTL_DOUT[17]	**B1**	R1	**G1**	
Pin81	PAD_TTL18	TTL_DOUT[18]	**B2**	R2	**G2**	
Pin82	PAD_TTL19	TTL_DOUT[19]	**B3**	R3	**G3**	
Pin83	PAD_TTL20	TTL_DOUT[20]	**B4**	R4	**G4**	
Pin84	PAD_TTL21	TTL_DOUT[21]	**B5**	R5	**G5**	
Pin85	PAD_TTL22	TTL_DOUT[22]	**B6**	R6	**G6**	
Pin86	PAD_TTL23	TTL_DOUT[23]	**B7**	R7	**G7**	
Pin87	PAD_TTL24	TTL_CK				
Pin88	PAD_TTL25	TTL_HSYNC				
Pin89	PAD_TTL26	TTL_VSYNC				
Pin90	PAD_TTL27	TTL_DE				

2、如下表所示，芯片默认 MIPI 硬件连接顺序。

Pin Location	Ball Pin Name	Function
Pin65	PAD_TTL6	MIPI_TX_P_CH0
Pin66	PAD_TTL7	MIPI_TX_N_CH0
Pin67	PAD_TTL8	MIPI_TX_P_CH1
Pin68	PAD_TTL9	MIPI_TX_N_CH1
Pin69	PAD_TTL10	MIPI_TX_P_CH2
Pin70	PAD_TTL11	MIPI_TX_N_CH2
Pin71	PAD_TTL12	MIPI_TX_P_CH3
Pin72	PAD_TTL13	MIPI_TX_N_CH3
Pin73	PAD_TTL14	MIPI_TX_P_CH4
Pin74	PAD_TTL15	MIPI_TX_N_CH4