CV180X & CV181X MIPI 使用手册

Version: 1.1.2

Release date: 2023-04-13

目录

修订记录

Revision	Date	Description
1.0.0	2021/04/20	RD update by review
1.1.1	2021/06/21	更正 MIPI_TX ioctl
1.1.2	2023/04/13	添加 181X/180X 细节

1 声明

法律声明

本数据手册包含北京晶视智能科技有限公司（下称“晶视智能”）的保密信息。未经授权，禁止使用或披露本数据手册中包含的信息。如您未经授权披露全部或部分保密信息，导致晶视智能遭受任何损失或损害，您应对因之产生的损失/损害承担责任。

本文件内信息如有更改，恕不另行通知。晶视智能不对使用或依赖本文件所含信息承担任何责任。

本数据手册和本文件所含的所有信息均按“原样”提供，无任何明示、暗示、法定或其他形式的保证。晶视智能特别声明未做任何适销性、非侵权性和特定用途适用性的默示保证，亦对本数据手册所使用、包含或提供的任何第三方的软件不提供任何保证；用户同意仅向该第三方寻求与此相关的任何保证索赔。此外，晶视智能亦不对任何其根据用户规格或符合特定标准或公开讨论而制作的可交付成果承担责任。

联系我们

地址 北京市海淀区丰豪东路 9 号院中关村集成电路设计园（ICPARK）1 号楼

深圳市宝安区福海街道展城社区会展湾云岸广场 T10 栋

电话 +86-10-57590723 +86-10-57590724

邮编 100094（北京）518100（深圳）

官方网站 https://www.sophgo.com/

技术论坛 https://developer.sophgo.com/forum/index.html

2 MIPI 使用指南

2.1 概述

MIPI Rx 可接收差分与 DC(TTL) 接口数据, 并将数据转换成 pixel 数据后传给下一级的 ISP 模块。差分讯号支持 SubLVDS(Sub Low-Voltage Differential Signal), MIPI-CSI 与 HiSPi(High-Speed Serial Pixel Interface) 等视频输入。DC 讯号支持 Sensor RAW12, BT1120, BT656 与 BT601。

2.2 重要概念

· MIPI

移动行业处理器接口-Mobile Industry Processor Interface，MIPI 特指物理层使用 D-PHY 传输规范并使用 CSI-2 为协议层的通信接口。

· Lane

物理层用于连接发送端和接收端的一对高速差分线。一个 Lane 可传送时钟或数据。1C4D 指一个时钟 Lane 及 4 个资料 Lane。

· LVDS

低压差分信号 (Low Voltage differential Signaling)，这里的 LVDS 泛指 LVDS 发展的 sub-LVDS 与 HiSPi，通过同步码区分消隐区和有效数据。

· 同步码

MIPI-CSI 利用标准的短包 (Short Packet) 当做同步讯号。LVDS 讯号利用同包码 (Sync Code) 作为同步讯号。LVDS 有两种同步模式:

· 使用 SOF/EOF 表示一帧的开始与结束。

· 使用 SOL/EOL 表示行的开始与结束。

图 1-1 SOF/EOF/SOL/EOL 同步方式

<table>
<tr><td colspan="5">VBLANK</td></tr>
<tr><td>HBLANK</td><td>SOF</td><td>Active Line 1</td><td rowspan="7">EOL</td><td>HBLANK</td></tr>
<tr><td>HBLANK</td><td rowspan="6">SOL</td><td>Active Line 2</td><td>HBLANK</td></tr>
<tr><td>HBLANK</td><td>Active Line 3</td><td>HBLANK</td></tr>
<tr><td>⋮</td><td>⋮</td><td>⋮</td></tr>
<tr><td>HBLANK</td><td>Active Line P-1</td><td>HBLANK</td></tr>
<tr><td>HBLANK</td><td>Active Line P</td><td>EOF</td><td>HBLANK</td></tr>
<tr><td colspan="5">VBLANK</td></tr>
</table>

- 使用 SAV invalid 与 EAV invalid 表示 VBLANK 的开始与结束。使用 SAV valid 与 EAV valid 表示有效数据 (information line, H.OB 与 pixel data) 的开始与结束。

图 1-2 SAV/EAV 同步方式

<table>
<tr><td>HBLANK</td><td rowspan="4">SAV Invalid</td><td>VBLANK</td><td rowspan="4">EAV Invalid</td><td>HBLANK</td></tr>
<tr><td>HBLANK</td><td>VBLANK</td><td>HBLANK</td></tr>
<tr><td>HBLANK</td><td>⋮</td><td>HBLANK</td></tr>
<tr><td>⋮</td><td>VBLANK</td><td>⋮</td></tr>
<tr><td>HBLANK</td><td rowspan="5">SAV Valid</td><td>Frame Information Line</td><td rowspan="5">EAV Valid</td><td>HBLANK</td></tr>
<tr><td>HBLANK</td><td>OB/ effective pixel</td><td>HBLANK</td></tr>
<tr><td>HBLANK</td><td>OB/ effective pixel</td><td>HBLANK</td></tr>
<tr><td>HBLANK</td><td>⋮</td><td>HBLANK</td></tr>
<tr><td>HBLANK</td><td>OB/ effective pixel</td><td>HBLANK</td></tr>
<tr><td>HBLANK</td><td rowspan="4">SAV Invalid</td><td>VBLANK</td><td rowspan="4">EAV Invalid</td><td>HBLANK</td></tr>
<tr><td>HBLANK</td><td>VBLANK</td><td>HBLANK</td></tr>
<tr><td>HBLANK</td><td>⋮</td><td>HBLANK</td></tr>
<tr><td>HBLANK</td><td>VBLANK</td><td>HBLANK</td></tr>
</table>

- DOI

SONY 的交错式 WDR 模式，全称为 Digital Overlap。

2.3 功能描述

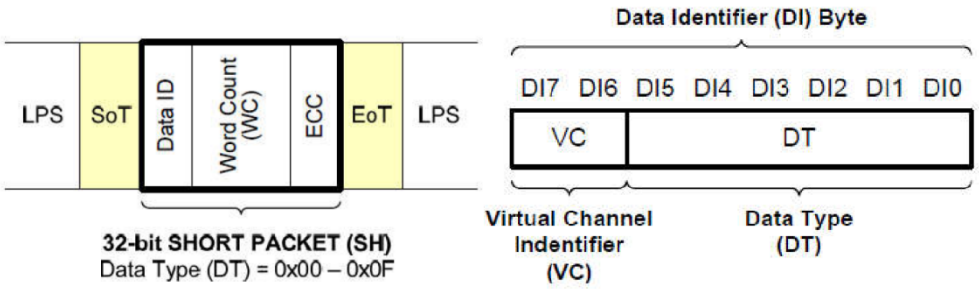
MIPI RX 支持一路最大 4 lanes 或两路最大 2 lanes 的 MIPI-CSI，sub-LVDS 与 HiSPi 差分视频输入接口，接口支持 lane 互换与差分讯号的 PN 互换。注意若使用两路差分讯号输入，脚位 MIPIRX# 必须 0 到 2 同一组输入，3 到 5 同一组输入。MIPI RX 也支持 sensor TTL，BT1120，BT656 与 BT601 等平行输入接口，支持除了 Clock 以外的 lane 功能互换。具体的 Lane 分布如下表:

表 1-1 支持 BT 接口分布表. BGA 封装支持 VI0~VI2, QFN 封装支持 VI0.

VIO	PAD NAME	VI1	PAD NAME	VI2	PAD NAME
VIO_CLK	MIPIRX4N	VI1_CLK	VIVO_CLK	VI2_CLK	VIDO_CLK
VI0_D[0]	MIPIRX4P	VI1_D[0]	VIVO_D0	VI2_D[0]	VIVO_D0
VI0_D[1]	MIPIRX3N	VI1_D[1]	VIVO_D1	VI2_D[1]	VIVO_D1
VI0_D[2]	MIPIRX3P	VI1_D[2]	VIVO_D2	VI2_D[2]	VIVO_D2
VI0_D[3]	MIPIRX2N	VI1_D[3]	VIVO_D3	VI2_D[3]	VIVO_D3
VI0_D[4]	MIPIRX2P	VI1_D[4]	VIVO_D4	VI2_D[4]	VIVO_D4
VI0_D[5]	MIPIRX1N	VI1_D[5]	VIVO_D5	VI2_D[5]	VIVO_D5
VI0_D[6]	MIPIRX1P	VI1_D[6]	VIVO_D6	VI2_D[6]	VIVO_D6
VI0_D[7]	MIPIRX0N	VI1_D[7]	VIVO_D7	VI2_D[7]	VIVO_D7
VI0_D[8]	MIPIRX0P	VI1_D[8]	VIVO_D8		
VI0_D[9]	MIPI_TXM0	VI1_D[9]	VIVO_D9		
VI0_D[10]	MIPI_TXP0	VI1_D[10]	VIVO_D10		
VI0_D[11]	MIPI_TXM1	VI1_D[11]	MIPIRX5N		
VI0_D[12]	MIPI_TXP1	VI1_D[12]	MIPIRX5P		
VI0_D[13]	MIPI_TXM2	VI1_D[13]	MIPIRX4N		
VI0_D[14]	MIPI_TXP2	VI1_D[14]	MIPIRX4P		
		VI1_D[15]	MIPIRX3N		
		VI1_D[16]	MIPIRX3P		
		VI1_D[17]	MIPIRX2N		
		VI1_D[18]	MIPIRX2P		

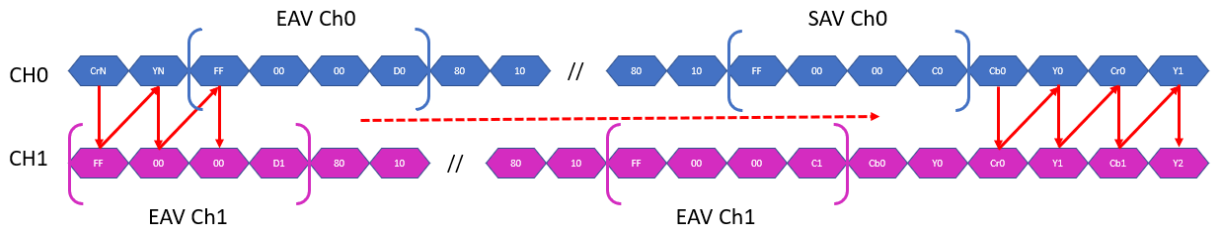
MIPI RX 支持 MIPI-CSI 的解多任务 (CSI Demux)，可接收不同 channel ID 的信道数据。

图 1-3 MIPI-CSI 的 channel ID



MIPI RX 支持 BT 接口的解多任务 (BT Demux)，输入端将不同信道的数据以 BT656 字节交错的格式打包，MIPI RX 可还原各通道再以对应的同步码解出有效数据。注意此模式只支持单沿取样 (SDR)。

图 1-4 BT demux 模式支持的输入



2.4 API 参考

MIPI Rx 提供对接 sensor 时序的功能。提供 ioctl 接口，可用的命令如下:

- CVI_MIPI_SET_HS_MODE :CVI_MIPI_SET_HS_MODE: 设置 MIPI 的 Lane 分布模式。
- CVI_MIPI_SET_DEV_ATTR：设置 MIPI 和并口设备属性。
- CVI_MIPI_RESET_SENSOR：复位 sensor。
- CVI_MIPI_UNRESET_SENSOR：撤销复位 sensor。
- CVI_MIPI_RESET_MIPI：复位 MIPI Rx。
- CVI_MIPI_ENABLE_SENSOR_CLOCK：打开 SENSOR 的时钟。
- CVI_MIPI_DISABLE_SENSOR_CLOCK：关闭 SENSOR 的时钟。
- CVI_MIPI_SET_CROP_TOP：舍弃每帧中的首 N 条资料。
- CVI_MIPI_SET_WDR_MANUAL：打开 WDR 手动模式。
- CVI_MIPI_SET_LVDS_FP_VS：设定 LVDS 中 VSYNC 生成的时间点。

MIPI Tx 提供对接显示屏、级联的功能。

其中单系统提供了 ioctl 接口，可通过传输以下相关命令进行相应的配置:

- CVI_VIP_MIPI_TX_SET_DEV_CFG：设置 MIPI Tx 设备的属性。
- CVI_VIP_MIPI_TX_GET_CMD：从 MIPI Tx 设备读取信息。
- CVI_VIP_MIPI_TX_SET_CMD：设备发送命令给 MIPI Tx 设备。
- CVI_VIP_MIPI_TX_ENABLE：启用 MIPI Tx 设备。
- CVI_VIP_MIPI_TX_DISABLE：禁用 MIPI Tx 设备。
- CVI_VIP_MIPI_TX_SET_HS_SETTLE：设定 MIPI Tx 在 HS mode 下的 settle timing。
- CVI_VIP_MIPI_TX_GET_HS_SETTLE：取得 MIPI Tx 在 HS mode 下的 settle timing。

双系统提供了 CVI_MSG_SendSync 接口，可通过传输以下相关命令进行相应的配置:

- MSG_CMD_MIPI_TX_SET_DEV_CFG：设置 MIPI Tx 设备的属性。
- MSG_CMD_MIPI_TX_GET_CMD：从 MIPI Tx 设备读取信息。

- MSG_CMD_MIPI_TX_SET_CMD：设备发送命令给 MIPI Tx 设备。
- MSG_CMD_MIPI_TX_ENABLE：启用 MIPI Tx 设备。
- MSG_CMD_MIPI_TX_DISABLE：禁用 MIPI Tx 设备。
- MSG_CMD_MIPI_TX_SET_HS_SETTLE：设定 MIPI Tx 在 HS mode 下的 settle timing。
- MSG_CMD_MIPI_TX_GET_HS_SETTLE：取得 MIPI Tx 在 HS mode 下的 settle timing。

2.4.1 CVI_MIPI_SET_HS_MODE

【描述】

相关功能被 CVI_MIPI_SET_DEV_ATTR 取代。

2.4.2 CVI_MIPI_SET_DEV_ATTR

【描述】

设置 MIPI 和并口设备属性。

【参数】

```
#define CVI_MIPI_SET_DEV_ATTR _IOW(CVI_MIPI_IOC_MAGIC, 0x01, struct combo_dev_
↪attr_t)
```

【定义】

struct combo_dev_attr_t 类型的指针。

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_vip_cif.h

【注意】

- 配置 CVI_MIPI_SET_DEV_ATTR 之前, 需要使用 ISP 接口打开 MIPI_RX 时钟. 详情请见 ISP 相关文件。
- 除了配置 CVI_MIPI_SET_DEV_ATTR 之前, 还需要配置以下接口。
- 复位 MIPI: 接口为 CVI_MIPI_RESET_MIPI。

- 打开 Sensor 的时钟: 接口为 CVI_MIPI_ENABLE_SENSOR_CLOCK
- 复位 Sensor: 接口为 CVI_MIPI_RESET_SENSOR
- 撤销复位 Sensor: 接口为 CVI_MIPI_UNRESET_SENSOR
- 推荐的配置流程如下:
  1. 打开 ISP 时钟。
  2. 复位对接的 Sensor。
  3. 复位 MIPI Rx。
  4. 配置 MIPI Rx 设备属性。
  5. 打开 Sensor 所连接的时钟。
  6. 撤销复位对接的 Sensor。
- 推荐的退出程序如下:
  1. 复位对接的 Sensor。
  2. 关闭 Sensor 所连接的时钟。
  3. 复位 MIPI Rx。
  4. 关闭 ISP 时钟。
- 操作 Sensor 复位信号线和时钟信号线会对所连接到该信号线的所有 Sensor 都产生效果。

【相关数据类型及接口】

- CVI_MIPI_RESET_MIPI
- CVI_MIPI_ENABLE_SENSOR_CLOCK 关闭 ISP 时钟。
- CVI_MIPI_DISABLE_SENSOR_CLOCK
- CVI_MIPI_RESET_SENSOR
- CVI_MIPI_UNRESET_SENSOR

2.4.3 CVI_MIPI_RESET_SENSOR

【描述】

复位 sensor

【定义】

```
#define CVI_MIPI_RESET_SENSOR _IOW(CVI_MIPI_IOC_MAGIC, 0x05, unsigned int)
```

【参数】

unsigned int。Sensor 复位信号线编号。

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_vip_cif.h

【注意】

- Sensor 复位信号绑定在对应的 dts。

```
mipi_rx: cif
{
  compatible = "cvitek,cif";
  reg = <0x0 0x0a0c2000 0x0 0x2000>, <0x0 0x0a0d0000 0x0 0x1000>,<0x0 0x0a0c4000 0x0 0x2000>;
  reg-names = "csi_mac0", "csi_wrap0", "csi_mac1";
  interrupts = <GIC_SPI 155 IRQ_TYPE_LEVEL_HIGH>,
               <GIC_SPI 156 IRQ_TYPE_LEVEL_HIGH>;
  interrupt-names = "csi0", "csi1";
  snsr-reset = <&porta 2 GPIO_ACTIVE_LOW>, <&porta 2 GPIO_ACTIVE_LOW>;
  resets = <&rst RST_CSIPHY0>, <&rst RST_CSIPHY1>,
        <&rst RST_CSIPHY0RST_APB>, <&rst RST_CSIPHY1RST_APB>;
  reset-names = "phy0", "phy1", "phy-apb0", "phy-apb1";
  clocks = <&clk CV181X_CLK_CAM0>, <&clk CV181X_CLK_CAM1>, <&clk CV181X_CLK_
↪SRC_VIP_SYS_2>;
  clock-names = "clk_cam0", "clk_cam1", "clk_sys_2";
};
```

2.4.4 CVI_MIPI_UNRESET_SENSOR

【描述】

撤销复位 sensor。

【定义】

```
#define CVI_MIPI_UNRESET_SENSOR _IOW(CVI_MIPI_IOC_MAGIC, 0x06, unsigned int)
```

【参数】

unsigned int。Sensor 复位信号线编号。

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_vip_cif.h

【注意】

· Sensor 复位信号绑定在对应的 dts。

```
mipi_rx: cif
 {
  compatible = "cvitek,cif";
  reg = <0x0 0x0a0c2000 0x0 0x2000>, <0x0 0x0a0d0000 0x0 0x1000>,
      <0x0 0x0a0c4000 0x0 0x2000>;
  reg-names = "csi_mac0", "csi_wrap0", "csi_mac1";
  interrupts = <GIC_SPI 155 IRQ_TYPE_LEVEL_HIGH>,
           <GIC_SPI 156 IRQ_TYPE_LEVEL_HIGH>;
  interrupt-names = "csi0", "csi1";
  snsr-reset = <&porta 2 GPIO_ACTIVE_LOW>, <&porta 2 GPIO_ACTIVE_LOW>;
  resets = <&rst RST_CSIPHY0>, <&rst RST_CSIPHY1>,
       <&rst RST_CSIPHY0RST_APB>, <&rst RST_CSIPHY1RST_APB>;
  reset-names = "phy0", "phy1", "phy-apb0", "phy-apb1";
  clocks = <&clk CV181X_CLK_CAM0>, <&clk CV181X_CLK_CAM1>, <&clk CV181X_CLK_
↪SRC_VIP_SYS_2>;
  clock-names = "clk_cam0", "clk_cam1", "clk_sys_2";
 };
```

2.4.5 CVI_MIPI_RESET_MIPI

【描述】

复位 MIPI_Rx。

【定义】

```
#define CVI_MIPI_RESET_MIPI _IOW(CVI_MIPI_IOC_MAGIC, 0x07, unsigned int)
```

【参数】

unsigned int。MIPI_Rx 设备号。

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_vip_cif.h

【注意】

无

2.4.6 CVI_MIPI_ENABLE_SENSOR_CLOCK

【描述】

打开 Sensor 的时钟。

【定义】

```
#define CVI_MIPI_ENABLE_SENSOR_CLOCK _IOW(CVI_MIPI_IOC_MAGIC, 0x10, unsigned␣
↪int)
```

【参数】

unsigned int。MIPI_Rx 设备号。

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_vip_cif.h

【注意】

无

2.4.7 CVI_MIPI_DISABLE_SENSOR_CLOCK

【描述】

关闭 Sensor 的时钟。

【定义】

```
#define CVI_MIPI_DISABLE_SENSOR_CLOCK _IOW(CVI_MIPI_IOC_MAGIC, 0x11, unsigned␣
↪int)
```

【参数】

unsigned int。MIPI_Rx 设备号。

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_vip_cif.h

【注意】

无

2.4.8 CVI_MIPI_SET_CROP_TOP

【描述】

舍弃每帧中的首 N 条资料

【定义】

```
#define CVI_MIPI_SET_WDR_MANUAL _IOW(CVI_MIPI_IOC_MAGIC, 0x21, struct crop_
↪top_s)
```

【参数】

struct crop_top_s

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_vip_cif.h

【注意】

无

2.4.9 CVI_MIPI_SET_WDR_MANUAL

【描述】

打开 WDR 手动模式。

【定义】

```
#define CVI_MIPI_SET_WDR_MANUAL _IOW(CVI_MIPI_IOC_MAGIC, 0x21, struct manual_
↪wdr_s)
```

【参数】

struct manual_wdr_s

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_vip_cif.h

【注意】

无

2.4.10 CVI_MIPI_SET_LVDS_FP_VS

【描述】

设定 sub-LVDS 中 VSYNC 生成的时间点。

【定义】

```
#define CVI_MIPI _SET_LVDS_FP_VS _IOW(CVI_MIPI_IOC_MAGIC, 0x22, struct vsync_
↪gen_s)
```

【参数】

struct vsync_gen_s

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_vip_cif.h

【注意】

无

2.4.11 CVI_VIP_MIPI_TX_SET_DEV_CFG

【描述】

设置 MIPI Tx 设备的属性。

【定义】

```
#define CVI_VIP_MIPI_TX_SET_DEV_CFG _IOW(CVI_VIP_MIPI_TX_IOC_MAGIC, 0x01,␣
↪struct combo_dev_cfg_s)
```

【参数】

MIPI Tx 设备属性结构体，详见 combo_dev_cfg_s 结构体说明。

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_mipi_tx.h

【注意】

无

2.4.12 MSG_CMD_MIPI_TX_SET_DEV_CFG

该双系统命令的描述、参数、返回值、处理器差异、需求、注意项皆与单系统命令 CVI_VIP_MIPI_TX_SET_DEV_CFG 相同。

【定义】

```
typedef enum tagMSG_MIPI_TX_CMD_E {
  MSG_CMD_MIPI_TX_SET_DEV_CFG = 0,
  MSG_CMD_MIPI_TX_SET_CMD,
  MSG_CMD_MIPI_TX_GET_CMD,
```

(下页继续)

(续上页)

```
    MSG_CMD_MIPI_TX_ENABLE,
    MSG_CMD_MIPI_TX_DISABLE,
    MSG_CMD_MIPI_TX_SET_HS_SETTLE,
    MSG_CMD_MIPI_TX_GET_HS_SETTLE,
    MSG_CMD_MIPI_TX_BUFF
} MSG_MIPI_TX_CMD_E;
```

双系统后续命令的定义都来自该枚举，不再叙述。

2.4.13 CVI_VIP_MIPI_TX_GET_CMD

【描述】

从 MIPI Tx 设备读取信息。

【定义】

```
#define CVI_VIP_MIPI_TX_GET_CMD _IOWR(CVI_VIP_MIPI_TX_IOC_MAGIC, 0x04,␣
↪struct get_cmd_info_s)
```

【参数】

详见 get_cmd_info_s 结构体说明

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_mipi_tx.h

【注意】

无

2.4.14 MSG_CMD_MIPI_TX_GET_CMD

该双系统命令的描述、参数、返回值、处理器差异、需求、注意项皆与单系统命令 CVI_VIP_MIPI_TX_GET_CMD 相同。

【定义】

见 MSG_CMD_MIPI_TX_SET_DEV_CFG 命令的定义说明。

2.4.15 CVI_VIP_MIPI_TX_SET_CMD

【描述】

设备发送命令给 MIPI Tx 设备。

【定义】

单系统:

```
#define CVI_VIP_MIPI_TX_SET_CMD _IOW(CVI_VIP_MIPI_TX_IOC_MAGIC, 0x02, struct␣
↪cmd_info_s)
```

【参数】

详见 cmd_info_s 结构体说明。

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_mipi_tx.h

【注意】

无

2.4.16 MSG_CMD_MIPI_TX_SET_CMD

该双系统命令的描述、参数、返回值、处理器差异、需求、注意项皆与单系统命令 CVI_VIP_MIPI_TX_SET_CMD 相同。

【定义】

见 MSG_CMD_MIPI_TX_SET_DEV_CFG 命令的定义说明。

2.4.17 CVI_VIP_MIPI_TX_ENABLE

【描述】

启用 MIPI Tx 设备。

【定义】

```
#define CVI_VIP_MIPI_TX_ENABLE _IO(CVI_VIP_MIPI_TX_IOC_MAGIC, 0x03)
```

【参数】

无

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_mipi_tx.h

【注意】

无

2.4.18 MSG_CMD_MIPI_TX_ENABLE

该双系统命令的描述、参数、返回值、处理器差异、需求、注意项皆与单系统命令 CVI_VIP_MIPI_TX_ENABLE 相同。

【定义】

见 MSG_CMD_MIPI_TX_SET_DEV_CFG 命令的定义说明。

2.4.19 CVI_VIP_MIPI_TX_DISABLE

【描述】

禁用 MIPI Tx 设备。

【定义】

```
#define CVI_VIP_MIPI_TX_DISABLE _IO(CVI_VIP_MIPI_TX_IOC_MAGIC, 0x05)
```

【参数】

无

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_mipi_tx.h

【注意】

无

2.4.20 MSG_CMD_MIPI_TX_DISABLE

该双系统命令的描述、参数、返回值、处理器差异、需求、注意项皆与单系统命令 CVI_VIP_MIPI_TX_DISABLE 相同。

【定义】

见 MSG_CMD_MIPI_TX_SET_DEV_CFG 命令的定义说明。

2.4.21 CVI_VIP_MIPI_TX_SET_HS_SETTLE

【描述】

设定 MIPI Tx 在 HS mode 下的 settle timing。

【定义】

```
#define CVI_VIP_MIPI_TX_SET_HS_SETTLE _IOW(CVI_VIP_MIPI_TX_IOC_MAGIC, 0x06,
↪ struct hs_settle_s)
```

【参数】

详见 hs_settle_s 结构体说明。

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_mipi_tx.h

【注意】

2.4.22 MSG_CMD_MIPI_TX_SET_HS_SETTLE

该双系统命令的描述、参数、返回值、处理器差异、需求、注意项皆与单系统命令 CVI_VIP_MIPI_TX_SET_HS_SETTLE 相同。

【定义】

见 MSG_CMD_MIPI_TX_SET_DEV_CFG 命令的定义说明。

2.4.23 CVI_VIP_MIPI_TX_GET_HS_SETTLE

【描述】

取得 MIPI Tx 在 HS mode 下的 settle timing。

【定义】

```
#define CVI_VIP_MIPI_TX_GET_HS_SETTLE _IOWR(CVI_VIP_MIPI_TX_IOC_MAGIC,␣
↪0x06, struct hs_settle_s)
```

【参数】

详见 hs_settle_s 结构体说明。

【返回值】

返回值	描述
0	成功
-1	失败，并设置 errno

【处理器差异】

无

【需求】

头文件: cvi_mipi_tx.h

【注意】

2.4.24 MSG_CMD_MIPI_TX_GET_HS_SETTLE

该双系统命令的描述、参数、返回值、处理器差异、需求、注意项皆与单系统命令 CVI_VIP_MIPI_TX_GET_HS_SETTLE 相同。

【定义】

见 MSG_CMD_MIPI_TX_SET_DEV_CFG 命令的定义说明。

2.5 数据类型

MIPI RX 相关数据类型定义如下:

- CVI_MIPI_IOC_MAGIC :MIPI Rx ioctl 命令的幻数。
- MIPI_LANE_NUM :MIPI Rx 的 MIPI 设备支持的最大 Lane 数。
- WDR_VC_NUM : 定义最多支持的 Virtual Channel 数量。
- SYNC_CODE_NUM : 定义 LVDS 每个 Virtual Channel 的同步码数量。
- input_mode_e :MIPI Rx 输入接口类型。
- img_size_s :MIPI Rx 输入数据每帧的大小。
- rx_mac_clk_e :MAC 的支持工作时钟。
- cam_pll_freq_e :MIPI-RX 输出的 Sensor 参考时钟。
- mclk_pll_s :MIPI-RX 输出的 Sensor 参考时钟设定。
- raw_data_type_e :MIPI Rx 输入数据格式。
- mipi_wdr_mode_e :MIPI-CSI WDR 模式。
- wdr_mode_e :LVDS/HISPI WDR 模式。
- lvds_sync_mode_e :LVDS 同步模式。
- lvds_bit_endian : 比特位大小端模式。
- lvds_vsync_type_e :LVDS 在 WDR 模式的同步方式。
- lvds_fid_type_e :LVDS frame identification ID 类型
- lvds_vsync_type_s: LVDS WDR 同步参数
- lvds_dev_attr_s: SubLVDS/HiSPi 设备属性
- dphy_s: MIPI DPHY 属性
- mipi_demux_info_s: MIPI 解多任务模式属性
- mipi_dev_attr_s: MIPI-CSI 设备属性
- manual_wdr_attr_s: 手动 WDR 模式参数
- ttl_pin_func_e: TTL/BT 接口的配置功能
- ttl_src_e: TTL/BT 接口输入来源
- bt_demux_mode_e: BT 解多任务模式的信道数量
- bt_demux_sync_s: BT 解多任务模式的同步码配置
- bt_demux_attr_s: BT 解多任务模式的设备属性
- ttl_dev_attr_s: TTL/BT 设备属性
- combo_dev_attr_s: combo 设备属性
- crop_top_s: 舍弃开头的行数据属性
- manual_wdr_s: 手动 WDR 模式设定

- vsync_gen_s: SUBLVDS 垂直同步信号属性

MIPI TX 相关数据类型定义如下:

- LANE_MAX_NUM: 一个 MIPI Tx 设备支持的最大 Lane 数
- output_mode_e: MIPI Tx 输出模式
- video_mode_e: MIPI Tx 视频模式
- output_format_e: MIPI Tx 输出格式
- sync_info_s: MIPI Tx 设备同步信息
- combo_dev_cfg_s: MIPI Tx 设备属性
- cmd_info_s: 从 MIPI Tx 设备取回信息
- get_cmd_info_s: 从 MIPI Tx 设备取回信息
- hs_settle_s: MIPI Tx 设备 HS mode 下的 settle 信息

2.5.1 CVI_MIPI_IOC_MAGIC

【说明】

MIPI Rx ioctl 命令的幻数

【定义】

```
#define CVI_MIPI_IOC_MAGIC    'm'
```

【处理器差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无

2.5.2 MIPI_LANE_NUM

【说明】

一个 MIPI Rx 设备支持的最大 Lane 数。

【定义】

```
#define MIPI_LANE_NUM  4
```

【处理器差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无

2.5.3 WDR_VC_NUM

【说明】

定义最多支持的 Virtual Channel 数量

【定义】

```
#define WDR_VC_NUM  2
```

【处理器差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无

2.5.4 SYNC_CODE_NUM

【说明】

定义 LVDS 每个 Virtual Channel 的同步码数量。

【定义】

```
#define SYNC_CODE_NUM  4
```

【处理器差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无

2.5.5 BT_DEMUX_NUM

【说明】

定义使用 bt demux 功能时支持最大的信道数量。

【定义】

```
#define BT_DEMUX_NUM  4
```

【处理器差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无

2.5.6 MIPI_DEMUX_NUM

【说明】

定义使用 mipi demux 功能时支持最大的 virtual channel 数量。

【定义】

```
#define MIPI_DEMUX_NUM  4
```

【处理器差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无

2.5.7 input_mode_e

【说明】

MIPI Rx 输入接口类型。

【定义】

```
enum input_mode_e
 {
    INPUT_MODE_MIPI = 0,
    INPUT_MODE_SUBLVDS,
    INPUT_MODE_HISPI,
    INPUT_MODE_CMOS,
    INPUT_MODE_BT1120,
    INPUT_MODE_BT601_19B_VHS,
    INPUT_MODE_BT656_9B,
    INPUT_MODE_CUSTOM_0,
    INPUT_MODE_BT_DEMUX,
    INPUT_MODE_BUTT
 };
```

【处理器差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无

2.5.8 img_size_s

【说明】

MIPI Rx 输入数据每帧的大小。

【定义】

```
struct img_size_s {
    unsigned int    width;
    unsigned int    height;
};
```

【处理器差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无

2.5.9 rx_mac_clk_e

【说明】

MAC 的支持工作时钟。

【定义】

```
enum rx_mac_clk_e {
    RX_MAC_CLK_200M = 0,
 RX_MAC_CLK_400M,
    RX_MAC_CLK_500M,
 RX_MAC_CLK_600M,
 RX_MAC_CLK_BUTT,
};
```

【处理器差异】

无。

【注意事项】

MAC 时钟与支持的 MIPI 时钟关系请参考 1.7.2。

【相关数据类型及接口】

无

2.5.10 cam_pll_freq_e

【说明】

MIPI-RX 输出的 Sensor 参考时钟。

【定义】

```
enum cam_pll_freq_e {
  CAMPLL_FREQ_NONE = 0,
  CAMPLL_FREQ_37P125M,
  CAMPLL_FREQ_25M,
  CAMPLL_FREQ_27M,
 CAMPLL_FREQ_24M,
 CAMPLL_FREQ_26M,
  CAMPLL_FREQ_NUM
};
```

【处理器差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无

2.5.11 mclk_pll_s

【说明】

MIPI-RX 输出的 Sensor 参考时钟设定。

【定义】

```
struct mclk_pll_s {
  unsigned int      cam;
  enum cam_pll_freq_e   freq;
};
```

【成员】

成员	描述
cam	0: 输出为 CAM_MCLK0 1: 输出为 CAM_MCLK1
freq	输出的 Sensor 参考时钟

【处理器差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无

2.5.12 raw_data_type_e

【说明】

MIPI Rx 输入数据格式。

【定义】

```
enum raw_data_type_e {
    RAW_DATA_8BIT = 0,
    RAW_DATA_10BIT,
    RAW_DATA_12BIT,
    YUV422_8BIT,    /* MIPI-CSI only */
    YUV422_10BIT,   /* MIPI-CSI only*/
    RAW_DATA_BUTT
};
```

【处理器差异】

无。

【注意事项】

YUV422_8BIT 与 YUV422_10BIT 只支持 MIPI-CSI 格式。

【相关数据类型及接口】

无

2.5.13 mipi_wdr_mode_e

【说明】

MIPI-CSI WDR 模式。

【定义】

```
enum mipi_wdr_mode_e {
    CVI_MIPI_WDR_MODE_NONE = 0,
    CVI_MIPI_WDR_MODE_VC,
    CVI_MIPI_WDR_MODE_DT,
    CVI_MIPI_WDR_MODE_DOL,
    CVI_MIPI_WDR_MODE_MANUAL,  /* SOI case */
    CVI_MIPI_WDR_MODE_BUTT
};
```

【成员】

成员	描述
CVI_MIPI_WDR_MODE_NONE	线性模式
CVI_MIPI_WDR_MODE_VC	MIPI-CSI Virtual Channel 模式
CVI_MIPI_WDR_MODE_DT	MIPI-CSI Data Type 模式
CVI_MIPI_WDR_MODE_DOL	Sony DOL LI 模式
CVI_MIPI_WDR_MODE_MANUAL	WDR 手动模式

【处理器差异】

无。

【注意事项】

- CVI_MIPI_WDR_MODE_VC 适用于使用 MIPI-CSI Virtual Channel ID 分辨长曝线与短曝线的 Sensor。
- CVI_MIPI_WDR_MODE_DT 适用于使用 MIPI-CSI Data Type ID 分辨长曝线与短曝线的 Sensor。注意注意长曝与短曝必须在同一帧开始与结束。
- CVI_MIPI_WDR_MODE_DOL 适用于使用 SONY MIPI-CSI Line Information 模式。
- CVI_MIPI_WDR_MODE_MANUAL 使用自定义的规则决定长曝线与短曝线。

【相关数据类型及接口】

无

2.5.14 wdr_mode_e

【说明】

Sub-LVDS/HISPI WDR 模式。

【定义】

```
enum wdr_mode_e {
    CVI_WDR_MODE_NONE = 0,
    CVI_WDR_MODE_2F,
    CVI_WDR_MODE_3F,
    CVI_WDR_MODE_DOL_2F,
    CVI_WDR_MODE_DOL_3F,
    CVI_WDR_MODE_DOL_BUTT
};
```

【成员】

成员	描述
CVI_WDR_MODE_NONE	线性模式
CVI_WDR_MODE_2F	一般双曝 WDR
CVI_WDR_MODE_3F	一般三曝 WDR
CVI_WDR_MODE_DOL_2F	Sony DOL-2 WDR
CVI_WDR_MODE_DOL_3F	Sony DOL-3 WDR

【处理器差异】

无。

【注意事项】

- CVI_WDR_MODE_2F 适用于一般 MIPI-CSI/HiSPi 的交错式 WDR。
- CVI_WDR_MODE_DOL_2F 适合用 Sony Sub-LVDS DOL-2 WDR
- CV181x 不支援 CVI_WDR_MODE_3F 与 CVI_WDR_MODE_DOL_3
- CV180X 不支持 WDR mode

【相关数据类型及接口】

无

2.5.15 lvds_sync_mode_e

【说明】

LVDS 同步模式。

【定义】

```
enum lvds_sync_mode_e {
    LVDS_SYNC_MODE_SOF = 0,
    LVDS_SYNC_MODE_SAV,
    LVDS_SYNC_MODE_BUTT
};
```

【成员】

成员	描述
LVDS_SYNC_MODE_SOF	SOF, EOF, SOL, EOL。请参考图 1-1
LVDS_SYNC_MODE_SAV	Invalid SAV, invalid EAV, valid SAV, valid EAV。请参考图 1-2

【处理器差异】

无。

【注意事项】

- LVDS_SYNC_MODE_SOF 适用于 HiSPi Packetize-SP 模式。
- LVDS_SYNC_MODE_SAV 适用于 sub-LVDS 与 HiSPi Streaming-SP 模式
- 当输入为 INPUT_MODE_HISPI 并且同步模式为 LVDS_SYNC_MODE_SAV。MIPI-Rx 切换到 HiSPi Streaming-SP 模式。

【相关数据类型及接口】

无

2.5.16 lvds_bit_endian

【说明】

比特位大小端模式。

【定义】

```
enum lvds_bit_endian {
    LVDS_ENDIAN_LITTLE = 0,
    LVDS_ENDIAN_BIG,
    LVDS_ENDIAN_BUTT
};
```

【处理器差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无

2.5.17 lvds_vsync_type_e

【说明】

LVDS 在 WDR 模式的同步方式。

【定义】

```
enum lvds_vsync_type_e {
    LVDS_VSYNC_NORMAL = 0,
    LVDS_VSYNC_SHARE,
    LVDS_VSYNC_HCONNECT,
    LVDS_VSYNC_BUTT
};
```

【成员】

成员	描述
LVDS_VSYNC_NORMAL	长短曝光帧有独立的 SOF-EOF, SOL-EOL 或 invalid-SAV-invalid-EAV, valid SAV-valid EAV。
LVDS_VSYNC_SHARE	长短曝光帧共享一对 SOF-EOF 标识，短曝光的起始几行用固定值填充。
LVDS_VSYNC_HCONNECT	长短曝光帧共享一对 SAV-EAV 标识，长短曝光帧之间是固定周期的消隐。

图 1-3 LVDS_VSYNC_NORMAL 同步方式.

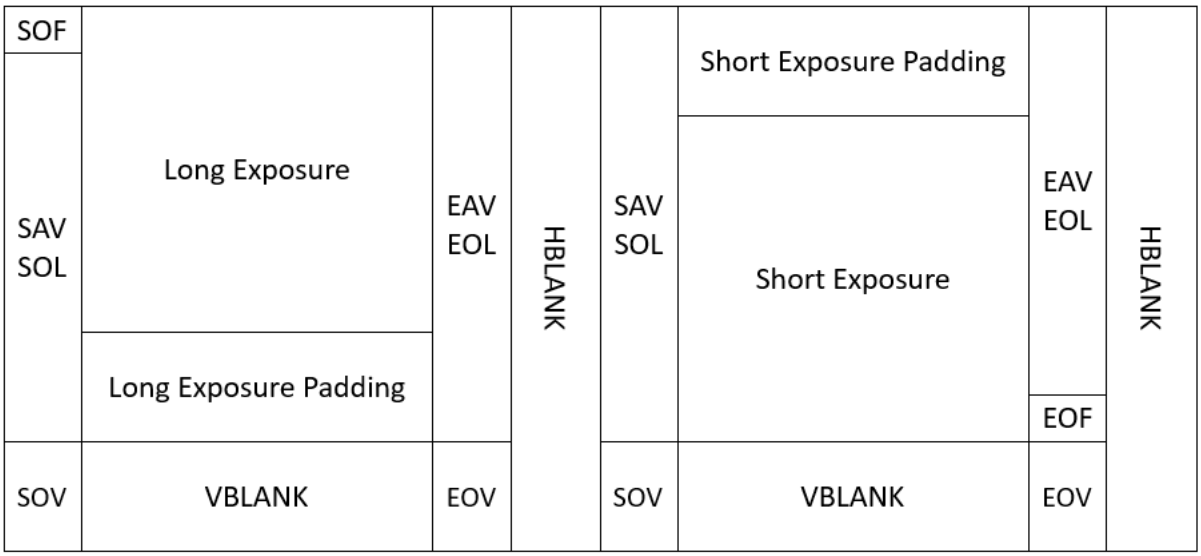
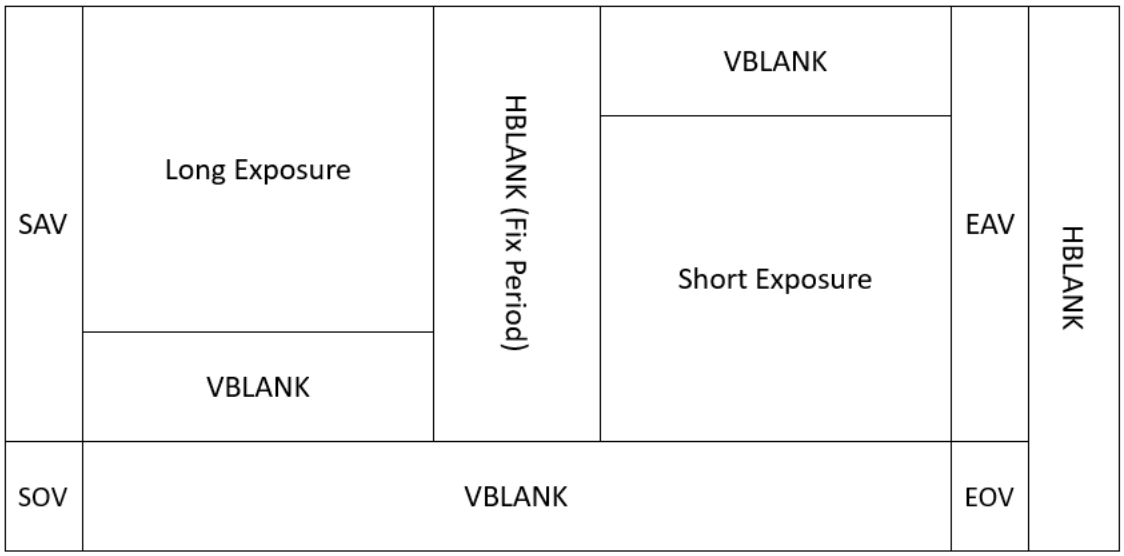


图 1-4 LVDS_VSYNC_HCONNECT 同步方式.



【处理器差异】

无。

【注意事项】

- LVDS_VSYNC_NORMAL 适用于 SONY sub-LVDS DOL-2 pattern 1 与 HiSPi Packtized-SP WDR 模式。
- LVDS_VSYNC_SHARE 适用于 HiSPi Streaming-SP WDR 模式。
- LVDS_VSYNC_HCONNECT 适用于 SONY sub-LVDS DOL-2 pattern 2。

【相关数据类型及接口】

无

2.5.18 lvds_fid_type_e

【说明】

LVDS frame identification ID 类型。

【定义】

```
enum lvds_fid_type_e {
    LVDS_FID_NONE = 0,
    LVDS_FID_IN_SAV,
    LVDS_FID_BUTT,
};
```

【成员】

成员	描述
LVDS_FID_NONE	不使用 frame identification id。
LVDS_FID_IN_SAV	FID 插入在 SAV 第 4 个字段中。

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.19 lvds_fid_type_s

【说明】

LVDS frame identification ID 类型。

【定义】

```
struct lvds_fid_type_s {
        enum lvds_fid_type_e            fid;
};
```

【成员】

成员	描述
fid	LVDS DOL 模式下的 frame identification 类型

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.20 lvds_vsync_type_s

【说明】

LVDS WDR 同步参数。

【定义】

```
struct lvds_vsync_type_s {
        enum lvds_vsync_type_e  sync_type;
        unsigned short                  hblank1;
        unsigned short                  hblank2;
};
```

【处理器差异】

无。

【注意事项】

- 当 sync_type 为 LVDS_VSYNC_HCONNECT 时，需要配置 hblank1 和 hblank2，表示长短曝间的消隐区长度。

【相关数据类型及接口】

无

2.5.21 lvds_dev_attr_s

【说明】

LVDS/SubLVDS/HiSPi 设备属性。

【定义】

```
struct lvds_dev_attr_s {
    enum wdr_mode_e             wdr_mode;
    enum lvds_sync_mode_e       sync_mode;
    enum raw_data_type_e        raw_data_type;
    enum lvds_bit_endian        data_endian;
    enum lvds_bit_endian        sync_code_endian;
    short                       lane_id[MIPI_LANE_NUM+1];
    short  sync_code[MIPI_LANE_NUM][WDR_VC_NUM+1][SYNC_CODE_NUM];
    struct lvds_vsync_type_s    vsync_type;
    struct lvds_fid_type_s      fid_type;
    char                        pn_swap[MIPI_LANE_NUM+1];
};
```

【成员】

成员	描述
wdr_mode	WDR 模式
sync_mode	LVDS 同步模式
raw_data_type	传输的数据类型
data_endian	数据大小端模式
sync_code_endian	同步码大小端模式
lane_id	发送端 (sensor) 和接收端 (MIPI Rx) lane 的对应关系
sync_code	每个 Virtual Channel 有 4 个同步码，根据同步模式不同，分别表示 SOF/EOF/SOL/EOL 的同步码或者 invalid SAV/invalid EAV/ valid SAV/valid EAV 的同步码。
vsync_type	vsync 类型，当 wdr_mod 为 DOL 模式并且 sync_mode 为 LVDS_SYNC_MODE_SAV 时，需要配置 vsync 的类型。
fid_type	frame identification 类型
pn_swap	Positive/negative line 是否交换

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.22 dphy_s

【说明】

MIPI RX DPHY 属性。

【定义】

```
struct mipi_dev_attr_s {
    unsigned char       enable;
    unsigned char       hs_settle;
};
```

【成员】

成员	描述
enable	开启 MIPI RX DPHY 属性设定
hs_settle	hs settle time

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.23 mipi_demux_info_s

【说明】

MIPI CSI 使用 Virtual Channel 解多任务的属性设定。

【定义】

```
struct mipi_demux_info_s {
    unsigned int            demux_en;
    unsigned char           vc_mapping[MIPI_DEMUX_NUM];
};
```

【成员】

成员	描述
demux_en	开启 mipi virtual channel 解多任务
vc_mapping	设定 ISP channel 与 mipi virtual channel 对应关系. 例如 vc_mapping = {0, 2, 3, 1}, ISP ch0 代表 vc=0; ch1 代表 vc=2; ch2 代表 vc=3; ch3 代表 vc=1.

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.24 mipi_dev_attr_s

【说明】

MIPI-CSI 设备属性。

【定义】

```
struct mipi_dev_attr_s {
    enum raw_data_type_e        raw_data_type;
    short                       lane_id[MIPI_LANE_NUM+1];
    enum mipi_wdr_mode_e        wdr_mode;
    short                       data_type[WDR_VC_NUM];
    char                        pn_swap[MIPI_LANE_NUM+1];
};
```

【成员】

成员	描述
wdr_mode	WDR 模式
raw_data_type	传输的数据类型
lane_id	发送端 (sensor) 和接收端 (MIPI Rx) lane 的对应关系
data_type	当 WDR 模式为 CVI_MIPI_WDR_MODE_DT 时, 每个 WDR frames 对应的 data type。
pn_swap	Positive/negative line 是否交换。

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.25 manaul_wdr_attr_s

【说明】

手动 WDR 模式参数。

【定义】

```
struct manaul_wdr_attr_s {
        unsigned int                manual_en;
        unsigned short               l2s_distance;
        unsigned short               lsef_length;
        unsigned int                discard_padding_lines;
        unsigned int                update;
};
```

【成员】

成员	描述
manual_en	手动 WDR 开关
l2s_distance	一帧中首条长曝到首列短曝的距离, 单位为行数。
lsef_length	长曝/短曝的行数。
discard_padding_lines	Sensor 是否有把 padding 行当有效行输出。
update	是否强制更新设定。若否, 设定会等下一张的同步讯号来才更新。

【处理器差异】

无。

【注意事项】

- HiSPi 输入并且 sync_type 为 LVDS_VSYNC_SHARE 时，需打开手动 WDR 模式并设定参数。
- MIPI-CSI 输入并且 wdr mode 为 CVI_MIPI_WDR_MODE_MANUAL 时，需打开手动 WDR 模式并设定参数。

【相关数据类型及接口】

无

2.5.26 ttl_pin_func_e

【说明】

TTL/BT 接口的配置功能.

【定义】

```
enum ttl_pin_func_e {
    TTL_PIN_FUNC_VS,
    TTL_PIN_FUNC_HS,
    TTL_PIN_FUNC_VDE,
    TTL_PIN_FUNC_HDE,
    TTL_PIN_FUNC_D0,
    TTL_PIN_FUNC_D1,
    TTL_PIN_FUNC_D2,
    TTL_PIN_FUNC_D3,
    TTL_PIN_FUNC_D4,
    TTL_PIN_FUNC_D5,
    TTL_PIN_FUNC_D6,
    TTL_PIN_FUNC_D7,
    TTL_PIN_FUNC_D8,
    TTL_PIN_FUNC_D9,
    TTL_PIN_FUNC_D10,
    TTL_PIN_FUNC_D11,
    TTL_PIN_FUNC_D12,
    TTL_PIN_FUNC_D13,
    TTL_PIN_FUNC_D14,
    TTL_PIN_FUNC_D15,
    TTL_PIN_FUNC_NUM,
};
```

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.27 ttl_src_e

【说明】

TTL/BT 接口输入来源.

【定义】

```
enum ttl_src_e {
    TTL_VI_SRC_VI0 = 0,
    TTL_VI_SRC_VI1,
```

(下页继续)

(续上页)

```
    TTL_VI_SRC_VI2,       /* BT demux */
    TTL_VI_SRC_NUM
};
```

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

参考表 1-1.

2.5.28 bt_demux_mode_e

【说明】

BT 解多任务模式的信道数量。

【定义】

```
enum bt_demux_mode_e {
    BT_DEMUX_DISABLE = 0,
    BT_DEMUX_2,
    BT_DEMUX_3,
    BT_DEMUX_4,
};
```

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.29 bt_demux_sync_s

【说明】

BT 解多任务模式的同步码设定。

【定义】

```
struct bt_demux_sync_s {
    unsigned char         sav_vld;
    unsigned char         sav_blk;
```

(下页继续)

(续上页)

```
    unsigned char          eav_vld;
    unsigned char          eav_blk;
};
```

【成员】

成员	描述
sav_vld	有效数据区间的 SAV
sav_blk	空白区间的 SAV
eav_vld	有效数据区间的 EAV
eav_blk	空白区间的 EAV

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.30 bt_demux_attr_s

【说明】

BT 解多任务模式的属性设定。

【定义】

```
struct bt_demux_attr_s {
    signed char                 func[TTL_PIN_FUNC_NUM];
    unsigned short               v_fp;
    unsigned short               h_fp;
    unsigned short               v_bp;
    unsigned short               h_bp;
    enum bt_demux_mode_e            mode;
    unsigned char               sync_code_part_A[3];    /* sync code 0~2 */
    struct bt_demux_sync_s         sync_code_part_B[BT_DEMUX_NUM]; /* sync code 3 */
    char                        yc_exchg;
};
```

【成员】

成员	描述
func	BT 接口对应输入源 TTL_VI_SRC_VI2 的 lane number. Index 为 BT 逻辑功能, value 请参考表 1-1. 例如, func[TTL_PIN_FUNC_D0] = 5, 代表 BT 讯号的 D0 接到 VI2_D[5], 即 pad name VIVO_D5。
v_fp	垂直方向的 front porch
h_fp	水平方向的 front porch
v_bp	垂直方向的 back porch
h_bp	水平方向的 back porch
mode	BT 解多任务模式的信道数量
sync_code	BT 解多任务模式的 0~2 同步码
sync_code	BT 解多任务模式的同步码 3
yc_exchg	BIT0~BIT3 分别代表 CH0~CH3 的 Y Cb/Cr bytes 顺序互换. 1 为互换, 0 为不互换.

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.31 ttl_dev_attr_s

【说明】

TTL/BT 接口的属性设定。

【定义】

```
struct ttl_dev_attr_s {
    enum ttl_src_e              vi;
    signed char                 func[TTL_PIN_FUNC_NUM];
    unsigned short              v_bp;
    unsigned short              h_bp;
};
```

【成员】

成员	描述
vi	TTL/BT 接口的输入来源, 允许值为 TTL_VI_SRC_VI0 或 TTL_VI_SRC_VI1
func	BT 接口对应输入源的 lane number. Index 为 BT 逻辑功能, value 请参考表 1-1. 例如, vi = TTL_VI_SRC_VI1 时, func[TTL_PIN_FUNC_D0] = 5, 代表 BT 讯号的 D0 接到 VI1_D[5], 即 pad name VIVO_D5.
v_bp	垂直方向的 back porch
h_bp	水平方向的 back porch

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.32 combo_dev_attr_s

【说明】

combo 设备属性，由于 MIPI Rx 能够对接 CSI-2，sub-LVDS，HiSPi 等时序，所以将 MIPI Rx 称为 combo 设备。

【定义】

```
struct combo_dev_attr_s {
    enum input_mode_e          input_mode;
    enum rx_mac_clk_e           mac_clk;
    struct mclk_pll_s          mclk;
    union {
        struct mipi_dev_attr_s  mipi_attr;
        struct lvds_dev_attr_s  lvds_attr;
        struct ttl_dev_attr_s   ttl_attr;
        struct bt_demux_attr_s  bt_demux_attr;
    };
    unsigned int               devno;
    struct img_size_s          img_size;
    struct manaul_wdr_attr_s       wdr_manu;
};
```

【成员】

成员	描述
input_mode	输入接口类型
mac_clk	MIPI RX MAC 时钟设定
mclk	MIPI RX 输出的 Sensor 参考时钟设定
mipi_attr	如果 input_mode 配置为 INPUT_MODE_MIPI, 则必须配置 mipi_attr
lvds_attr	如果 input_mode 配置为 INPUT_MODE_SUBLVDS/INPUT_MODE_HISPI, 则必须配置 lvds_attr
devno	MIPI-Rx 设备号
img_size	输入帧大小
wdr_manu	手动 WDR 属性

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.33 crop_top_s

【说明】

舍弃开头的行资料。

【定义】

```
struct crop_top_s {
        unsigned int                devno;
        unsigned int                crop_top;
        unsigned int                update;
};
```

【成员】

成员	描述
devno	MIPI-Rx 设备号
crop_top	开头要拾弃的行数
update	是否强制更新设定。若否，设定会等下一张的同步讯号来才更新。

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.34 manual_wdr_s

【说明】

手动 WDR 模式设定。

【定义】

```
struct manual_wdr_s {
        unsigned int                devno;
        struct manaul_wdr_attr_s        attr;
};
```

【成员】

成员	描述
devno	MIPI-Rx 设备号
attr	手动 WDR 属性

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.35 vsync_gen_s

【说明】

手动 WDR 模式设定。

【定义】

```
struct vsync_gen_s {
    unsigned int        devno;
    unsigned int        distance_fp;
};
```

【成员】

成员	描述
devno	MIPI-Rx 设备号
distance_fp	当 input_mode 为 INPUT_MODE_SUBLVDS 时，产生垂直同步信号的时间点。

【处理器差异】

无。

【注意事项】

- Sub-LVDS 不自带垂直同步信号，所以 MIPI-Rx 须自行生成送给 ISP。distance_fp 可调整垂直同步时间点以达到加大 front porch 的作用。

【相关数据类型及接口】

无

2.5.36 LANE_MAX_NUM

【说明】

一个 MIPI Tx 设备支持的最大 Lane 数。

【定义】

```
#define MIPI_LANE_NUM  4
```

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.37 output_mode_e

【说明】

MIPI Tx 输出模式。

【定义】

```
enum output_mode_e
{
 OUTPUT_MODE_CSI          = 0x0,     /* csi mode */ OUTPUT_MODE_DSI_VIDEO     =␣
↪0x1,     /* dsi video mode */
 OUTPUT_MODE_DSI_CMD      = 0x2,     /* dsi command mode */
 OUTPUT_MODE_BUTT
} output_mode_t;
```

【处理器差异】

无

【注意事项】

无

【相关数据类型及接口】

无

2.5.38 video_mode_e

【说明】

MIPI Tx 视频模式。

【定义】

```
enum video_mode_e {
  BURST_MODE                = 0x0,
  NON_BURST_MODE_SYNC_PULSES    = 0x1,
  NON_BURST_MODE_SYNC_EVENTS    = 0x2,
};
```

【处理器差异】

无

【注意事项】

无

【相关数据类型及接口】

无

2.5.39 output_format_e

【说明】

LVDS 在 WDR 模式的同步方式。

【定义】

```
enum output_format_e {
  OUT_FORMAT_RGB_16_BIT       = 0x0,
  OUT_FORMAT_RGB_18_BIT       = 0x1,
  OUT_FORMAT_RGB_24_BIT       = 0x2,
  OUT_FORMAT_RGB_30_BIT       = 0x3,
  OUT_FORMAT_YUV420_8_BIT_NORMAL = 0x4,
  OUT_FORMAT_YUV420_8_BIT_LEGACY = 0x5,
  OUT_FORMAT_YUV422_8_BIT     = 0x6,

  OUT_FORMAT_BUTT
};
```

【处理器差异】

处理器差异	是否支持
CV181x	不支持 YUV420
CV180x	不支持 YUV420

【注意事项】

无

【相关数据类型及接口】

无

2.5.40 sync_info_s

【说明】

MIPI Tx 设备同步信息。

【定义】

```
struct sync_info_s {
  unsigned short  vid_hsa_pixels;
  unsigned short  vid_hbp_pixels;
  unsigned short  vid_hfp_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;
  bool            vid_vsa_pos_polarity;
  bool            vid_hsa_pos_polarity;
};
```

【成员】

成员	描述
vid_hsa_pixels	Horizontal sync-pluse 像素个数
vid_hbp_pixels	Horizontal back-porch 像素个数
vid_hfp_pixels	Horizontal front-porch 像素个数
vid_hline_pixels	Horizontal image active 像素个数
vid_vsa_lines	Vertical sync-pluse 行数
vid_hbp_pixels	Vertical back-porch 行数
vid_hbp_pixels	Vertical front-porch 行数
vid_active_pixels	Vertical image active 行数
edpi_cmd_size	写内存命令字节数。video mode 时该值无效，command mode 时该值设为 hact。
vid_vsa_pos_polarity;	Horizontal sync-pluse polarity
vid_hsa_pos_polarity;	Vertical sync-pluse polarity

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

无

2.5.41 combo_dev_cfg_s

【说明】

MIPI Tx 设备属性。

【定义】

```
struct combo_dev_cfg_s {
    unsigned int            devno;
    enum mipi_tx_lane_id    lane_id[LANE_MAX_NUM];
    enum output_mode_e      output_mode;
    enum video_mode_e       video_mode;
    enum output_format_e    output_format;
    struct sync_info_s      sync_info;
    unsigned int            phy_data_rate;
    unsigned int            pixel_clk;
    bool                    lane_pn_swap[LANE_MAX_NUM];
};
```

【成员】

成员	描述
devno	devno
lane_id	发送端 (vo) 和接收端 (MIPI Tx) Lane 的对应关系; 未使用的 Lane 设置为-1。
output_mode	MIPI Tx 输出模式。
video_mode	MIPI Tx 视频模式。
output_format	MIPI Tx 输出格式。
sync_info	MIPI Tx 设备的同步信息。
phy_data_rate	MIPI Tx 输出速率，MIPI Tx 高速模式每个信道（lane）的速率范围的描述。
pixel_clk	像素时钟。单位为 KHz
lane_pn_swap	Lane 设置上是否 P/N 要互换

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

CVI_VIP_MIPI_TX_SET_DEV_CFG, MSG_CMD_MIPI_TX_SET_DEV_CFG

2.5.42 cmd_info_s

【说明】

给 MIPI Tx 设备初始化信息。

【定义】

```
struct cmd_info_s {
    unsigned int        devno;
    unsigned short      data_type;
    unsigned short      cmd_size;
#ifdef __arm__
    unsigned char       *cmd;
    unsigned int        padding;
#else
    unsigned char       *cmd;
#endif
};
```

【成员】

成员	描述
devno	MIPI Tx 设备号
data_type	命令数据类型
cmd_size	命令数据字节数，范围：(0,128)。
cmd_data	命令数据地址指针，需要用户分配。

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

CVI_VIP_MIPI_TX_SET_CMD, MSG_CMD_MIPI_TX_SET_CMD

2.5.43 get_cmd_info_s

【说明】

从 MIPI Tx 设备取回信息。

【定义】

```
struct get_cmd_info_s {
    unsigned int        devno;
    unsigned short      data_type;
    unsigned short      data_param;
    unsigned short      get_data_size;
#ifdef __arm__
```

(下页继续)

(续上页)

```
    unsigned int        padding1;
    unsigned char       *get_data;
    unsigned int        padding2;
#else
    unsigned int        padding1;
    unsigned char       *get_data;
#endif
};
```

【成员】

成员	描述
devno	MIPI Tx 设备号
data_type	命令数据类型
data_param	数据参数，低八比特为第一个参数，高八比特为第二个参数、不用时填 0
get_data_size	预期获取的数据字节数，范围：(0,4)。
get_data	获取到的数据存放地址指针，需要用户分配。

【处理器差异】

无。

【注意事项】

无

【相关数据类型及接口】

CVI_VIP_MIPI_TX_GET_CMD, MSG_CMD_MIPI_TX_GET_CMD

2.5.44 hs_settle_s

【说明】

MIPI Tx 设备 HS mode 下的 settle 信息。

【定义】

```
struct hs_settle_s {
    unsigned char       prepare;
    unsigned char       zero;
    unsigned char       trail;
};
```

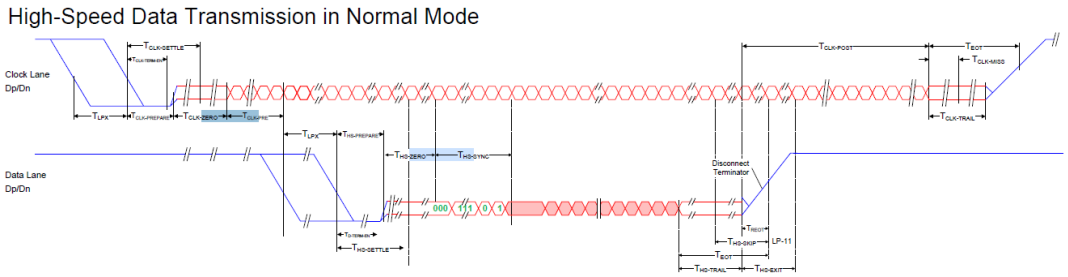
【成员】

成员	描述
prepare	为 LP->HS 后的前置准备 T 数
zero	在进入 HS 后，在输出数据前为 0 的 T 数
trail	在 HS->LP 时，要结束 HS 前，数据结束后额外的 T 数

【处理器差异】

无。

【注意事项】



【相关数据类型及接口】

无

2.6 Proc 信息

2.6.1 MIPI_RX Proc 信息

MIPI_Rx 在正常工作下 proc 信息中的各种错误中断计数应为 0。若否，请检查 MIPI_Rx 相关属性是否配置正确。

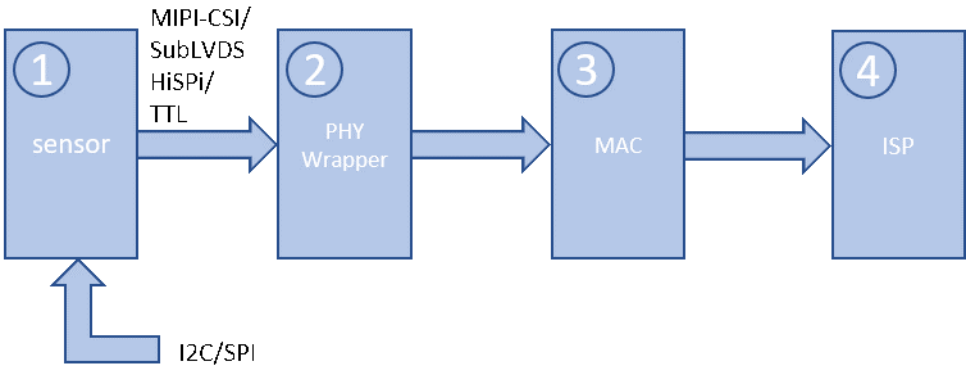
【调试信息】

```
Module: [MIPI_RX], Build Time[#1 SMP PREEMPT Thu Apr 29 11:18:57 CST 2021]

----------Combo DEV ATTR----------
 Devno  WorkMode  DataType   WDRMode        LinkId       PN Swap  SyncMode  DataEndian
↪SyncCodeEndian
    0      MIPI     RAW12      NONE  2, 3, 1, 4, 0  1, 1, 1, 1, 1      N/A       N/A          N/A

----------MIPI info-------------------
Devno EccErr CrcErr HdrErr WcErr fifofull   decode
   0      0      0      0     2        0    raw12
Physical:       D0       D1       D2       D3       D4       D5
                 0        0        0        0        0        0
 Digital:       D0       D1       D2       D3    CK_HS  CK_ULPS  CK_STOP   CK_ERR   Deskew
          hs_idle  hs_idle  hs_idle  hs_idle        1        0        0        0     idle
```

【调试信息分析】

- MIPI Rx 通过 PHY Wrapper 接收 Sensor 的 MIPI-CSI/SubLVDS/HiSPi/TTL 讯号，由 MAC 内对应的接口进行同步头检测与对齐。
- MAC 将各 Lane 的数据合并成 Pixel 数据，并将数据发送给后级的 ISP。
- PHY Wrapper 由 sensor 的 pixel clock 提供时钟。MAC 内的时钟与后级的 ISP 相同。
- 若要操作数据的 Crop，可由后级的 ISP 来调试。

图 1-5 数据示例图



【参数说明】

参数		描述
MIPI DEV ATTR	Devno	MIPI 设备号
	WorkMode	MIPI 设备工作模式: MIPI/SUBLVDS/HISPI/CMOS 等模式
	DataType	RAW8/RAW10/RAW12 等类型
	WDRMode	**WDR 模式:** · NONE · VC · DT · DOL · MANUAL
	LaneId	Lane id
	PN Swap	PN 讯号交换
LVDS DEV ATTR	Devno	MIPI 设备号
	WorkMode	MIPI 设备工作模式: MIPI/SUBLVDS/HISPI/CMOS 等模式
	DataType	RAW8/RAW10/RAW12 等类型
	WDRMode	**WDR 模式:** · NONE · 2To1 · 3To1 · DOL2To1 · DOL3To1
	LaneId	Lane id
	PN Swap	PN 讯号交换

下页继续

表 2.1 – 续上页

参数		描述
	SyncMode	**LVDS 的同步码模式:** · SOF · SAV
	DataEndian	Data 的比特位大小端模式
	SyncCodeEndian	同步码的比特位大小端模式
MIPI Info (仅 MIPI 模式可见)	Devno	MIPI 设备号
	EccErr	ECC 错误的中断计数
	CrcErr	CRC 错误的中断计数
	HdrErr	HDR Flag 错误的中断计数
	WcErr	Word Count 错误的中断计数
	fifofull	Fifofull 的中断计数
	Physical: D0	MIPIRX_PAD0 收到的资料
	Physical: D1	MIPIRX_PAD1 收到的资料
	Physical: D2	MIPIRX_PAD2 收到的资料
	Physical: D3	MIPIRX_PAD3 收到的资料
	Physical: D4	MIPIRX_PAD4 收到的资料
	Physical: D5	MIPIRX_PAD5 收到的资料
	Digital: D0	Sensor data lane 0 state
	Digital: D1	Sensor data lane 1 state
	Digital: D2	Sensor data lane 2 state
	Digital: D3	Sensor data lane 3 state
	CK_HS/CK_ULPS/CK_STOP/CK_HDR	Sensor clock lane state
	Deskew	Deskew 结果

2.7 FAQ

2.7.1 1.6.1. Land id 如何配置

Land id 的配置对应 mipi_dev_attr_s 中的 short lane_id[MIPI_LANE_NUM+1] 或者 lvds_dev_attr_s 中的 short lane_id[MIPI_LANE_NUM+1]，其中 lane_id 数组的索引号表示的是 Sensor 的 Lane ID，索引号 0 表示 sensor clock，索引号 1 表示 sensor lane 0。land_id 数组的值表示的是 MIPI-Rx 的 Lane ID，0 表示 MIPIRX1_PAD0，1 表示 MIPIRX1_PAD1。未使用的 lane 将其对应的 lane_id 配置为-1。

下面举例说明，例如 MIPI 与 SENSOR 的引脚硬件连接如下表所示。

SENSOR 管脚	MIPI Lane 管脚
Clock Lane (index = 0)	MIPIRX1_PAD0 (value = 0)
Lane 0 (index = 1)	MIPIRX1_PAD1 (value = 1)
Lane 1 (index = 2)	MIPIRX1_PAD2 (value = 2)
Lane 2 (index = 3)	MIPIRX1_PAD3 (value = 3)
Lane 3 (index = 4)	MIPIRX1_PAD4 (value = 4)

MIPI 的最大 Lane 数加上 Clock 为 5，所以 lane_id 配置如下:

```
//索引sensor_clk,  sensor_lane0,  sensor_lane1,  sensor_lane2,   sensor_lane3
//        ↓            ↓          ↓          ↓           ↓
land_id={MIPIRX1_PAD0  MIPIRX1_PAD1  MIPIRX1_PAD2  MIPIRX1_PAD3 MIPIRX1_PAD4
↪}
```

2.7.2 MIPI 频率说明

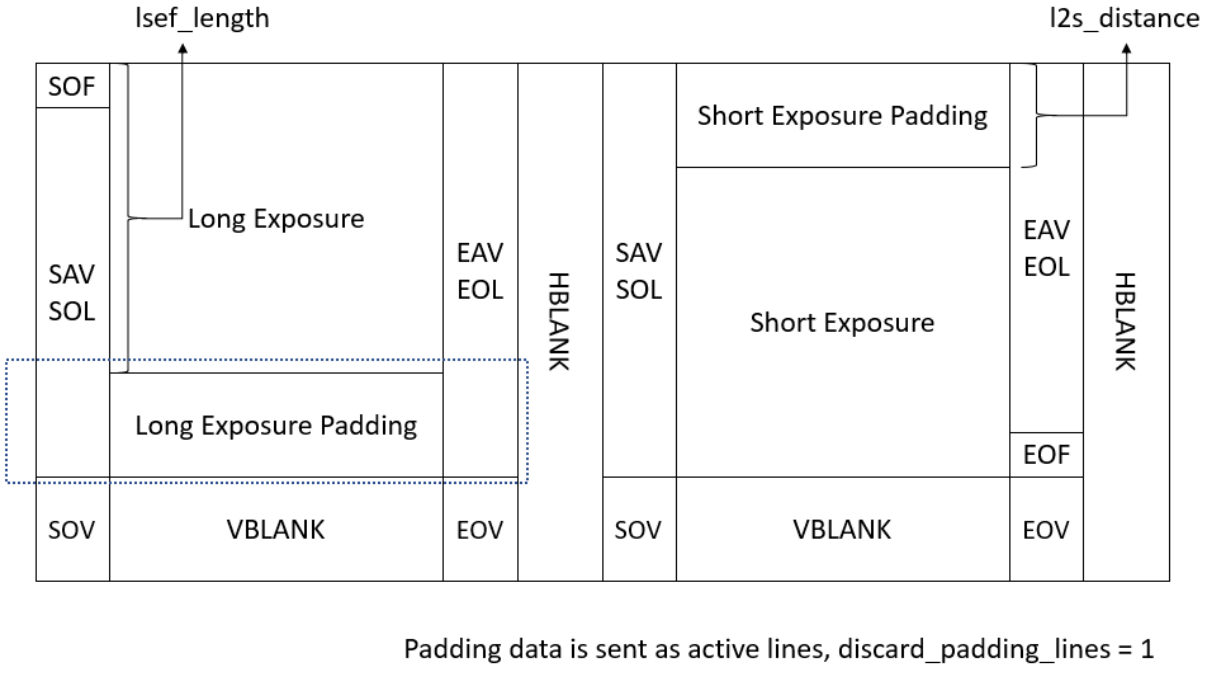
使用以下公式计算 MIPI 每 Lane 最高频率与 VI MAC 的工作频率:

MAC_Freq * pixel_width = lane_num * MIPI_Freq * 2。其中 MAC_Freq 为 VI MAC 的工作频率，pixel_width 为像素位宽，lane_num 为 MIPI lane 个数，MIPI_Freq 为每条 lane 的工作频率。若 MAC clock 为 400M，pixel_width = 12，lane_num = 4，可支持最快 MIPI_Freq = 400 * 12 / (4 * 2) = 600MHz。

2.7.3 Manual WDR 模式使用说明

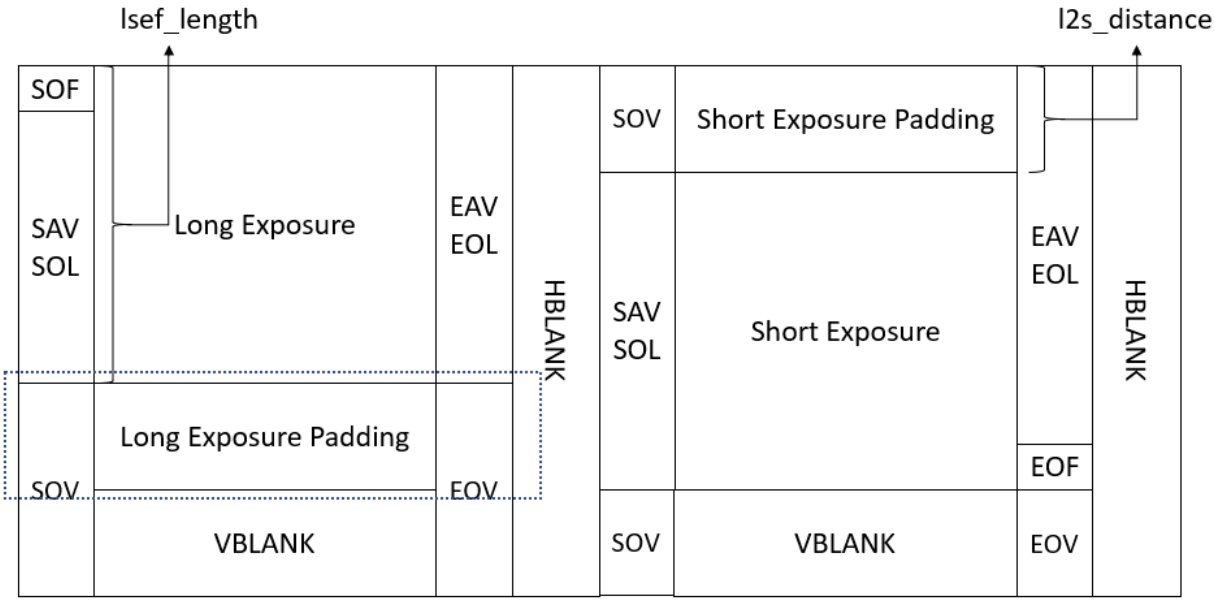
当手动 WDR 模式打开后，MIPI-Rx 会把收下来的数据以行为单位，遵遁以下规则分配给长曝帧与短曝帧。

- l2s_distance: 从第一行到第 l2s_distance 行都是长曝数据，第 l2s_distance+1 行开始长曝与短曝交错分配。
- lsef_length: 第 lsef_length+1 行开始都是短曝帧数据。直到下个垂直同步信号为止。
- 当 discard_padding_lines=1 时，1 到 l2s_distance 行分配方式为 {长-ignore-长-ignore …}，第 l2s_distance+1 到 lsef_length 行分配方式为 {长-短-长-短…}。第 lsef_length+1 行到下个垂直同步信号分配方式为 {短-ignore-短-ignore …}。



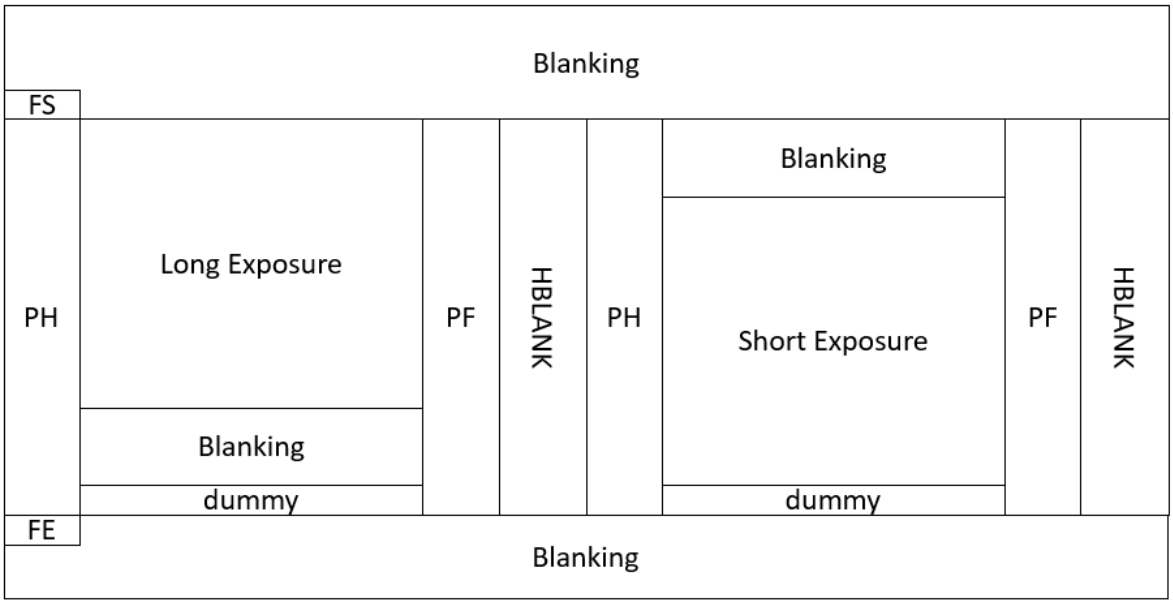
Padding data is sent as active lines, discard_padding_lines = 1

- 当 discard_padding_lines=0 时，1 到 l2s_distance 行分配方式为 {长-长-长…}，第 l2s_distance+1 到 lsef_length 行分配方式为 {长-短-长-短…}。第 lsef_length+1 行到下个垂直同步信号分配方式为 {短-短-短…}。



Padding data is sent as blanking lines, discard_padding_lines = 0

- MIPI-Rx 必须确保发送给 ISP 的长曝帧与短曝帧行数是一致的。
- 调整 sensor 短曝长度，有可能 l2s_distance 需要一起调整。
- 有些 sensor 可能会在送完短曝有效数据后带 dummy 行。这会造成长曝与短曝的行数不一致。可将 l2s_distance 设成 0，lsef_length 设成最大值 0x1FFF，discard_padding_lines 为 1 即收下两张带有效与 dummy 数据的长短曝，再用 ISP crop 有效位置即可。



discard_padding_lines = 1
l2s_distance = 0
lsef_length = 1FFF