



HiMPP MIPI

使用指南

文档版本 03

发布日期 2017-04-10

版权所有 © 深圳市海思半导体有限公司 2016-2017。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

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

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

深圳市海思半导体有限公司

地址：深圳市龙岗区坂田华为基地华为电气生产中心 邮编：518129

网址：<http://www.hisilicon.com>

客户服务电话：+86-755-28788858

客户服务传真：+86-755-28357515

客户服务邮箱：support@hisilicon.com



前 言

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3516A	V100
Hi3516D	V100
Hi3518E	V200
Hi3518E	V201
Hi3516C	V200
Hi3519	V100
Hi3519	V101
Hi3516C	V300
Hi3559	V100
Hi3556	V100
Hi3516A	V200

说明

- 未有特殊说明，Hi3516D 与 Hi3516A 一致。
- 未有特殊说明，Hi3518EV201、Hi3516CV200 和 Hi3518EV200 一致。
- 未有特殊说明，Hi3556V100、Hi3559V100、Hi3516AV200 和 Hi3519V101 一致。

读者对象



本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师



符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。
 警告	表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
 注意	表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
 窍门	表示能帮助您解决某个问题或节省您的时间。
 说明	表示是正文的附加信息，是对正文的强调和补充。

修订记录

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

文档版本 03 (2017-04-10)

1.4 小节，HI_MIPI_SET_CROP 涉及修改

1.5 小节，input_mode_t 的【定义】及【注意】涉及修改

文档版本 02 (2017-02-14)

1.4 小节，新增 HI_MIPI_SET_OUTPUT_CLK_EDGE 和 HI_MIPI_SET_OUTPUT_MSB

1.6 小节涉及修改

文档版本 01(2016-12-20)

第 1 次正式版本发布。

新增 Hi3559V100 相关内容。

文档版本 00B02(2016-07-12)

第 2 次临时版本发布。

新增 Hi3516CV300 相关内容。



1.4 小节，HI_MIPI_SET_DEV_ATTR 内容涉及修改。新增 API 参考
HI_MIPI_RESET_SENSOR、HI_MIPI_UNRESET_SENSOR、HI_MIPI_RESET_MIPI 和
HI_MIPI_UNRESET_MIPI。

1.5 小节，新增数据类型 COMBO_DEV、HI_MIPI_IOC_MAGIC、phy_cmv_e 和
phy_cmv_t。

文档版本 00B01 (2016-06-20)

第 1 次临时版本发布。



目 录

前 言.....	i
1 HiMPP MIPI 使用指南	1
1.1 概述	1
1.2 重要概念	1
1.3 功能描述	3
1.4 API 参考	3
1.5 数据类型	12
1.6 PROC 信息	38
1.7 FAQ	43
1.7.2 LVDS mode sync code 如何配置	44
1.7.3 MIPI 频率说明	50
1.7.4 Sensor 复位	50



1 HiMPP MIPI 使用指南

1.1 概述

MIPI Rx 通过低电压差分信号接收原始视频数据，将接收到的串行差分信号（serial differential signal）转化为 DC（Digital Camera）时序后传递给下一级模块 ViCAP（Video Capture）

MIPI Rx 支持 MIPI D-PHY、LVDS（Low-Voltage Differential Signal）、HiSPi（High-Speed Serial Pixel Interface）等串行视频信号输入，同时兼容 DC 视频接口。

1.2 重要概念

- **MIPI**
MIPI 的全称是 Mobile Industry Processor Interface(移动行业处理器接口)，本文描述的 MIPI 接口特指物理层使用 D-PHY 传输规范，协议层使用 CSI-2 的通信接口。
- **LVDS**
LVDS 的全称是 Low Voltage differential Signaling(低压差分信号)，通过同步码区分消隐区和有效数据。
- **Lane**
用于连接发送端和接收端的一对高速差分线，即可以是时钟 Lane，也可以是数据 Lane。
- **Link**
发送端和接收端之间的时钟 Lane 和至少一个数据 Lane 组成一个 Link。
- **同步码**
MIPI 接口使用 CSI-2 里面的短包进行同步，LVDS 使用同步码区分有效数据和消隐区。LVDS 有两种同步方式：
 - 使用 SOF/EOF 表示帧起始和结束，使用 SOL/EOL 表示行的起始和结束。同步方式如图 1-1 所示。



图1-1 SOF/EOF/SOL/EOL 同步方式

V.BLK				
H.BLK	SOF	Effective Pixel	EOL	H.BLK
H.BLK	SOL	Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
:		:		:
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
H.BLK	Effective Pixel	EOF	H.BLK	
V.BLK				

- 使用 SAV(Invalid) EAV(Invalid)表示消隐区的无效数据开始和结束，使用 SAV(Valid) EAV(Valid)表示有效像素数据的开始和结束。

每个同步码由 4 个字段组成，每个字段的位宽与像素数据位宽保持一致。前 3 个字段为固定基准码字，第 4 个字段由 sensor 厂家确定。

由于不同的 sensor 可能会有不同的同步码，所以需要根据 sensor 配置同步码。同步方式如图 1-2 所示。

图1-2 SAV/EAV 同步方式

H.BLK	SAV (Invalid line)	V.BLK	EAV (Invalid line)	H.BLK
H.BLK		V.BLK		H.BLK
H.BLK		V.BLK		H.BLK
H.BLK	SAV (Valid line)	H.OB / effective pixel	EAV (Valid line)	H.BLK
H.BLK		H.OB / effective pixel		H.BLK
H.BLK		H.OB / effective pixel		H.BLK
:		:		:
H.BLK		H.OB / effective pixel		H.BLK
H.BLK		H.OB / effective pixel		H.BLK
H.BLK		H.OB / effective pixel		H.BLK
H.BLK		H.OB / effective pixel		H.BLK
H.BLK		H.OB / effective pixel		H.BLK
H.BLK		H.OB / effective pixel		H.BLK
:	SAV (Invalid line)	V.BLK	EAV (Invalid line)	:
H.BLK		V.BLK		H.BLK
H.BLK		V.BLK		H.BLK
H.BLK		V.BLK		H.BLK

- DOL



DOL 的全称是 Digital Overlap，指 SONY 的 WDR 功能。

1.3 功能描述

MIPI Rx 是一个支持多种差分视频输入接口的采集单元，通过 combo-PHY 接收 MIPI/LVDS/sub-LVDS/HiSPi/DC 接口的数据，通过不同的功能模式配置，MIPI Rx 可以支持多种速度和分辨率的数据传输需求，支持多种外部输入设备。

表1-1 最大支持 lane 的个数

芯片类型	定义
Hi3516A	MIPI Rx 最大支持 1Link/4Lane MIPI 输入或 2Link/8Lane LVDS 输入。
Hi3518EV200/Hi3516CV300	MIPI Rx 最大支持 1Link/4Lane MIPI 输入或 1Link/4Lane LVDS 输入。
Hi3519V100/Hi3519V101	MIPI Rx 最大支持 2Link/8Lane MIPI 输入或 3Link/12Lane LVDS 输入。

以上输入管脚可复用为单端 DC/BT.1120 通道输入，从而可以用更少的芯片管脚提供更好的兼容性。

1.4 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：

- 支持所有芯片
 - `HI_MIPI_SET_DEV_ATTR`：设置 MIPI 设备属性。
 - `HI_MIPI_SET_OUTPUT_CLK_EDGE`：设置 PHY 输出数据的随路时钟相位。
 - `HI_MIPI_SET_OUTPUT_MSB`：设置 LVDS/HISPI 同步码的大小端。
- 仅支持 Hi3519V100/Hi3516CV300/Hi3519V101
 - `HI_MIPI_SET_PHY_CMVMODE`：设置共模电压模式。
 - `HI_MIPI_RESET_SENSOR`：复位 sensor。
 - `HI_MIPI_UNRESET_SENSOR`：撤销复位 sensor。
 - `HI_MIPI_RESET_MIPI`：复位 MIPI Rx。
 - `HI_MIPI_UNRESET_MIPI`：撤销复位 MIPI Rx。
- 仅支持 Hi3519V100 和 Hi3516CV300
 - `HI_MIPI_SET_CROP`：设置 MIPI CROP 属性。



HI_MIPI_SET_DEV_ATTR

【描述】

设置 MIPI Rx 设备属性。

【定义】

```
#define HI_MIPI_SET_DEV_ATTR                                _IOW(HI_MIPI_IOC_MAGIC, 0x01,  
combo_dev_attr_t)
```

【参数】

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

【芯片差异】

芯片类型	内部流程
Hi3516A	接口内部包含如下流程： 1. 复位 sensor 2. 复位 MIPI Rx 3. 撤销复位 MIPI Rx 4. 配置 MIPI RX 参数 5. 撤销复位 sensor
Hi3518EV200	
Hi3519V100	
Hi3519V101	配置 MIPI Rx 参数。
Hi3516CV300	

【需求】

头文件：hi_mipi.h

【注意】

Hi3519V100/Hi3519V101/Hi3516CV300:

- 除了配置 HI_MIPI_SET_DEV_ATTR 之外，还需要配置以下接口。
- 复位 sensor、撤销复位 sensor、复位 MIPI Rx 和撤销复位 MIPI Rx 的操作被剥离为独立的接口，对应的接口如下：



- 复位 sensor: [HI_MIPI_RESET_SENSOR](#)
- 撤销复位 sensor: [HI_MIPI_UNRESET_SENSOR](#)
- 复位 MIPI Rx: [HI_MIPI_RESET_MIPI](#)
- 撤销复位 MIPI Rx: [HI_MIPI_UNRESET_MIPI](#)
- 推荐的配置流程如下:
 1. 复位 MIPI Rx
 2. 复位 sensor
 3. 配置 MIPI Rx 设备属性
 4. 撤销复位 MIPI Rx
 5. 撤销复位 sensor

【相关数据类型及接口】

- [HI_MIPI_RESET_SENSOR](#)
- [HI_MIPI_UNRESET_SENSOR](#)
- [HI_MIPI_RESET_MIPI](#)
- [HI_MIPI_UNRESET_MIPI](#)

HI_MIPI_SET_OUTPUT_CLK_EDGE

【描述】

设置 PHY 输出数据的随路时钟相位，0 为时钟上升沿输出数据，1 为时钟下降沿输出数据。

【定义】

```
#define HI_MIPI_SET_OUTPUT_CLK_EDGE          _IOW(HI_MIPI_IOC_MAGIC, 0x02,  
clk_edge)
```

【参数】

clk_edge 类型的指针。

【返回值】

返回值	描述
0	成功。
-1	失败，并设置 errno

【芯片差异】

芯片类型	是否支持
Hi3516A	支持
Hi3518EV200	支持



芯片类型	是否支持
Hi3519V100	支持
Hi3519V101	支持
Hi3516CV300	支持

【需求】

头文件：hi_mipi.h

【注意】

无。

HI_MIPI_SET_OUTPUT_MSB

【描述】

设置 LVDS/HISPI 同步码的大小端，0 为小端模式，1 为大端模式。

【定义】

```
#define HI_MIPI_SET_OUTPUT_MSB                _IOW(HI_MIPI_IOC_MAGIC, 0x03,  
output_msb)
```

【参数】

output_msb 类型的指针。

【返回值】

返回值	描述
0	成功。
-1	失败，并设置 errno

【芯片差异】

芯片类型	是否支持
Hi3516A	支持
Hi3518EV200	支持
Hi3519V100	支持
Hi3519V101	支持
Hi3516CV300	支持



【需求】

头文件：hi_mipi.h

【注意】

该接口一般无需设置，在设置 mipi 属性接口中可以进行设置大小端。

HI_MIPI_SET_PHY_CMVMODE

【描述】

设置共模电压模式。

【定义】

```
#define HI_MIPI_SET_PHY_CMVMODE          _IOW(HI_MIPI_IOC_MAGIC, 0x04,  
phy_cmvm_t)
```

【参数】

phy_cmvm_t 类型的指针。

【返回值】

返回值	描述
0	成功。
-1	失败，并设置 errno

【芯片差异】

芯片类型	是否支持
Hi3516A	不支持
Hi3518EV200	不支持
Hi3519V100	支持
Hi3519V101	支持
Hi3516CV300	支持

【需求】

头文件：hi_mipi.h

【注意】

无。



HI_MIPI_SET_CROP

【描述】

设置 MIPI CROP 配置。

【定义】

Hi3519V100:

```
#define HI_MIPI_SET_CROP _IOW(HI_MIPI_IOC_MAGIC, 0x05, img_rect_t)
```

Hi3516CV300:

```
#define HI_MIPI_SET_CROP IOW(HI_MIPI_IOC_MAGIC, 0x09, img_rect_t)
```

【参数】

[img_rect_t](#) 类型的指针。

【返回值】

返回值	描述
0	成功。
-1	失败，并设置 errno

【芯片差异】

芯片类型	是否支持
Hi3516A	不支持
Hi3518EV200	不支持
Hi3519V100	支持
Hi3519V101	不支持
Hi3516CV300	支持

【需求】

头文件：hi_mipi.h

【注意】

- 在 [HI_MIPI_SET_DEV_ATTR](#) 接口后调用此接口，Crop 才会生效。
([HI_MIPI_SET_DEV_ATTR](#) 默认关闭 CROP 功能)。
- 需要根据 sensor 实际输出宽高进行 crop 配置，(x+width, y+height) 不能超过实际图像分辨率。
- Crop 的图像宽度必须是有效 lane 个数的整数倍。



HI_MIPI_RESET_SENSOR

【描述】

复位 sensor。

【定义】

```
#define HI_MIPI_RESET_SENSOR _IOW(HI_MIPI_IOC_MAGIC, 0x05, COMBO_DEV)
```

【参数】

COMBO_DEV 设备号。

【返回值】

返回值	描述
0	成功。
-1	失败，并设置 errno

【芯片差异】

芯片类型	是否支持
Hi3516A	不支持
Hi3518EV200	不支持
Hi3519V100	支持
Hi3519V101	支持
Hi3516CV300	支持

【需求】

头文件：hi_mipi.h

【注意】

无。

HI_MIPI_UNRESET_SENSOR

【描述】

撤销复位 sensor。

【定义】

```
#define HI_MIPI_UNRESET_SENSOR _IOW(HI_MIPI_IOC_MAGIC, 0x06,  
COMBO_DEV)
```



【参数】

COMBO_DEV 设备号。

【返回值】

返回值	描述
0	成功。
-1	失败，并设置 errno

【芯片差异】

芯片类型	是否支持
Hi3516A	不支持
Hi3518EV200	不支持
Hi3519V100	支持
Hi3519V101	支持
Hi3516CV300	支持

【需求】

头文件：hi_mipi.h

【注意】

无。

HI_MIPI_RESET_MIPI

【描述】

复位 MIPI_Rx。

【定义】

```
#define HI_MIPI_RESET_MIPI    _IOW(HI_MIPI_IOC_MAGIC, 0x07, COMBO_DEV)
```

【参数】

COMBO_DEV 设备号。

【返回值】

返回值	描述
0	成功。
-1	失败，并设置 errno



【芯片差异】

芯片类型	是否支持
Hi3516A	不支持
Hi3518EV200	不支持
Hi3519V100	支持
Hi3519V101	支持
Hi3516CV300	支持

【需求】

头文件：hi_mipi.h

【注意】

无。

HI_MIPI_UNRESET_MIPI

【描述】

撤销复位 MIPI_Rx。

【定义】

```
#define HI_MIPI_UNRESET_MIPI _IOW(HI_MIPI_IOC_MAGIC, 0x08, COMBO_DEV)
```

【参数】

COMBO_DEV 设备号。

【返回值】

返回值	描述
0	成功。
-1	失败，并设置 errno

【芯片差异】

芯片类型	是否支持
Hi3516A	不支持
Hi3518EV200	不支持



Hi3519V100	支持
Hi3519V101	支持
Hi3516CV300	支持

【需求】

头文件：hi_mipi.h

【注意】

无。

1.5 数据类型

MIPI Rx 相关数据类型定义如下：

- [HI_MIPI_IOC_MAGIC](#)：MIPI Rx ioctl 命令的幻数。
- [COMBO_DEV](#)：MIPI Rx 设备类型。
- [COMBO_MAX_LINK_NUM](#)：设备最大的支持的 Link 数量。
- [LANE_NUM_PER_LINK](#)：每个 Link 最多支持的 Lane。
- [COMBO_MAX_LANE_NUM](#)：设备最大支持的 Lane 数量。
- [MIPI_LANE_NUM](#)：MIPI 接口支持的 Lane 数量。
- [LVDS_LANE_NUM](#)：LVDS/HiSPi 接口支持的 Lane 数量。
- [COMBO_MAX_DEV_NUM](#)：MIPI Rx 设备的数量。
- [WDR_VC_NUM](#)：定义最多支持的 Virtual Chnnael 数量。
- [SYNC_CODE_NUM](#)：每个 Virtual Channel 的同步码数量。
- [input_mode_t](#)：MIPI Rx 输入接口类型。
- [phy_clk_share_e](#)：PHY1 与 PHY2 是否与 PHY0 共用时钟。
- [raw_data_type_e](#)：传输的 raw 数据比特数。
- [mipi_wdr_mode_e](#)：MIPI WDR 模式。
- [mipi_dev_attr_t](#)：MIPI 设备属性。
- [wdr_mode_e](#)：LVDS WDR 模式。
- [lvds_sync_mode_e](#)：LVDS 同步方式。
- [lvds_bit_endian](#)：比特位大小端模式。
- [lvds_vsync_type_e](#)：LVDS VSYNC 类型。
- [lvds_vsync_type_t](#)：LVDS VSYNC 参数。
- [lvds_fid_type_e](#)：Frame identification Id 类型。
- [lvds_fid_type_t](#)：Frame indentification Id 配置信息。
- [lvds_dev_attr_t](#)：LVDS/SubLVDS/HiSPi 设备属性。



- [phy_cmve](#): PHY 共模电压模式。
- [phy_cmvt](#): PHY 共模电压配置信息
- [combo_dev_attr_t](#): combo 设备属性。
- [img_rect_t](#): crop 属性。
- [img_size_t](#): 图像宽高属性。

HI_MIPI_IOC_MAGIC

【说明】

MIPI Rx ioctl 命令的幻数。

【定义】

```
#define HI_MIPI_IOC_MAGIC    'm'
```

【成员】

无

【芯片差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无

COMBO_DEV

【说明】

MIPI Rx 设备类型。

【定义】

```
typedef unsigned int COMBO_DEV;
```

【芯片差异】

无。

【注意事项】

无。

【相关数据类型及接口】

- [combo_dev_attr_t](#)
- [HI_MIPI_SET_DEV_ATTR](#)
- [HI_MIPI_RESET_SENSOR](#)



- [HI_MIPI_UNRESET_SENSOR](#)
- [HI_MIPI_RESET_MIPI](#)
- [HI_MIPI_UNRESET_MIPI](#)

COMBO_MAX_LINK_NUM

【说明】

设备最大的支持的 Link 数量。

【芯片差异】

芯片类型	定义
Hi3516A	#define COMBO_MAX_LINK_NUM 2
Hi3518EV200	#define COMBO_MAX_LINK_NUM 2 (只有一个 Link, 为了兼容 Hi3516A 所以定义为 2)
Hi3519V100	#define COMBO_MAX_LINK_NUM 3
Hi3519V101	#define COMBO_MAX_LINK_NUM 3
Hi3516CV300	#define COMBO_MAX_LINK_NUM 1

【注意事项】

无。

【相关数据类型及接口】

无。

LANE_NUM_PER_LINK

【说明】

每个 Link 最多支持的 Lane。

【定义】

```
#define LANE_NUM_PER_LINK 4
```

【芯片差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无。



COMBO_MAX_LANE_NUM

【说明】

设备最大支持的 Lane 数量。

【芯片差异】

芯片类型	定义
Hi3516A	#define COMBO_MAX_LANE_NUM 8
Hi3518EV200	#define COMBO_MAX_LANE_NUM 8 (只有 4 个 Lane，为了兼容 Hi3516A 所以定义为 8)
Hi3519V100	#define COMBO_MAX_LANE_NUM 12
Hi3519V101	#define COMBO_MAX_LANE_NUM 12
Hi3516CV300	#define COMBO_MAX_LANE_NUM 4

【注意事项】

无。

【相关数据类型及接口】

无。

MIPI_LANE_NUM

【说明】

MIPI 接口支持的 Lane 数量。

【芯片差异】

芯片类型	定义
Hi3516A	#define MIPI_LANE_NUM COMBO_MAX_LANE_NUM
Hi3518EV200	#define MIPI_LANE_NUM COMBO_MAX_LANE_NUM
Hi3519V100	#define MIPI_LANE_NUM (LANE_NUM_PER_LINK * 2)
Hi3519V101	#define MIPI_LANE_NUM (LANE_NUM_PER_LINK * 2)
Hi3516CV300	#define MIPI_LANE_NUM (LANE_NUM_PER_LINK * 1)

【注意事项】

无。

【相关数据类型及接口】

无。



LVDS_LANE_NUM

【说明】

LVDS/HiSPi 接口支持的 Lane 数量。

【定义】

```
#define LVDS_LANE_NUM COMBO_MAX_LANE_NUM
```

【芯片差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无。

COMBO_MAX_DEV_NUM

【说明】

MIPI Rx 设备的数量。

【芯片差异】

芯片类型	定义
Hi3516A	无
Hi3518EV200	无
Hi3519V100	#define COMBO_MAX_DEV_NUM 1
Hi3519V101	#define COMBO_MAX_DEV_NUM 2
Hi3516CV300	#define COMBO_MAX_DEV_NUM 1

【注意事项】

无。

【相关数据类型及接口】

无。

WDR_VC_NUM

【说明】

定义最多支持的 Virtual Chnnael 数量

【定义】



```
#define WDR_VC_NUM          4
```

【芯片差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无。

SYNC_CODE_NUM

【说明】

每个 Virtual Channel 的同步码数量

【定义】

```
#define SYNC_CODE_NUM      4
```

【芯片差异】

无

【注意事项】

无。

【相关数据类型及接口】

无。

input_mode_t

【说明】

MIPI Rx 输入接口类型

【定义】

Hi3516A/Hi3518EV200

```
typedef enum
```

```
{
```

```
    INPUT_MODE_MIPI          = 0x0,          /* mipi */
    INPUT_MODE_SUBLVDS        = 0x1,          /* SUB_LVDS */
    INPUT_MODE_LVDS           = 0x2,          /* LVDS */
    INPUT_MODE_HISPI          = 0x3,          /* HISPI */
    INPUT_MODE_CMOS_18V        = 0x4,          /* CMOS 1.8V */
    INPUT_MODE_CMOS_33V        = 0x5,          /* CMOS 3.3V */
    INPUT_MODE_BT1120          = 0x6,          /* CMOS 3.3V */
    INPUT_MODE_BYPASS          = 0x7,          /* MIPI Bypass */
```



```
    INPUT_MODE_BUTT  
}input_mode_t;
```

Hi3519V100/Hi3519V101

```
typedef enum  
{  
    INPUT_MODE_MIPI          = 0x0,          /* CSI-2 */  
    INPUT_MODE_SUBLVDS       = 0x1,          /* SUB_LVDS */  
    INPUT_MODE_LVDS          = 0x2,          /* LVDS */  
    INPUT_MODE_HISPI         = 0x3,          /* HISPI */  
    INPUT_MODE_CMOS           = 0x4,          /* CMOS */  
    INPUT_MODE_BT1120        = 0x5,          /* CMOS */  
    INPUT_MODE_BUTT  
} input_mode_t;
```

Hi3516CV300

```
typedef enum  
{  
    INPUT_MODE_MIPI          = 0x0,          /* CSI-2 */  
    INPUT_MODE_SUBLVDS       = 0x1,          /* SUB_LVDS */  
    INPUT_MODE_LVDS          = 0x2,          /* LVDS */  
    INPUT_MODE_HISPI         = 0x3,          /* HISPI */  
    INPUT_MODE_CMOS           = 0x4,          /* CMOS */  
    INPUT_MODE_BT1120        = 0x5,          /* CMOS */  
    INPUT_MODE_BUTT  
} input_mode_t;
```

【芯片差异】

芯片类型	input_mode_t
Hi3516A/Hi3518EV200	CMOS 模式定义为 INPUT_MODE_CMOS_18V、 INPUT_MODE_CMOS_33V。
Hi3519V100/Hi3519V101/Hi3516CV300	MIPI Rx 能够识别 1.8V 和 3.3V 的 CMOS, 所以 CMOS 模式统一定义为 INPUT_MODE_CMOS

【注意事项】

无。

【相关数据类型及接口】



无。

phy_clk_share_e

【说明】

PHY1 与 PHY2 是否与 PHY0 共用时钟

【定义】

```
typedef enum
{
    PHY_CLK_SHARE_NONE = 0x0,
    PHY_CLK_SHARE_PHY0 = 0x1,      /* PHY share clock with PHY0 */
    PHY_CLK_SHARE_BUTT = 0x2,
} phy_clk_share_e;
```

【成员】

成员名称	描述
PHY_CLK_SHARE_NONE	PHY1 和 PHY2 使用独立的时钟
PHY_CLK_SHARE_PHY0	PHY1 和 PHY2 共用 PHY0 时钟

【芯片差异】

芯片类型	是否支持 PHY 共用时钟
Hi3516A	不支持
Hi3518EV200	不支持
Hi3519V100	支持
Hi3519V101	支持
Hi3516CV300	不支持

【注意事项】

无。

【相关数据类型及接口】

无。

raw_data_type_e

【说明】



传输的 raw 数据比特数

【定义】

Hi3516A/Hi3518EV200

```
typedef enum
{
    RAW_DATA_8BIT = 1,
    RAW_DATA_10BIT,
    RAW_DATA_12BIT,
    RAW_DATA_14BIT,
    RAW_DATA_BUTT
}raw_data_type_e;
```

Hi3519V100/Hi3519V101/Hi3516CV300

```
typedef enum
{
    RAW_DATA_8BIT = 0,
    RAW_DATA_10BIT,
    RAW_DATA_12BIT,
    RAW_DATA_14BIT,
    RAW_DATA_16BIT,
    RAW_DATA_BUTT
} raw_data_type_e;
```

【芯片差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无。

mipi_wdr_mode_e

【说明】

MIPI WDR 模式。

【定义】

```
typedef enum
{
    HI_MIPI_WDR_MODE_NONE = 0x0,
    HI_MIPI_WDR_MODE_VC   = 0x1,    /* Virtual Channel */
    HI_MIPI_WDR_MODE_DT   = 0x2,    /* Data Type */
    HI_MIPI_WDR_MODE_DOL  = 0x3,    /* DOL Mode */
}
```



```
HI_MIPI_WDR_MODE_BUTT  
} mipi_wdr_mode_e;
```

【成员】

成员名称	描述
HI_MIPI_WDR_MODE_NONE	线性模式
HI_MIPI_WDR_MODE_VC	使用 Packet header 中的 Virtual Channel 区分长短曝光帧
HI_MIPI_WDR_MODE_DT	使用 Packet header 中的自定义 Data type 区分长短曝光帧
HI_MIPI_WDR_MODE_DOL	表示 DOL 模式 WDR，使用 Packet header 之后的一个 pixel 识别长短曝光帧

【芯片差异】

芯片类型	是否支持 mipi_wdr_mode_e
Hi3516A	不支持
Hi3518EV200	不支持
Hi3519V100	支持
Hi3519V101	支持
Hi3516CV300	支持

【注意事项】

无

【相关数据类型及接口】

无。

mipi_dev_attr_t

【说明】

mipi 设备属性。

【定义】

Hi3516A/Hi3518EV200:

```
typedef struct  
{  
    raw_data_type_e    raw_data_type;
```



```
short lane_id[MIPI_LANE_NUM];  
}mipi_dev_attr_t;
```

Hi3519V100/Hi3519V101/Hi3516CV300:

```
typedef struct  
{  
    raw_data_type_e raw_data_type;  
    mipi_wdr_mode_e wdr_mode;  
    short lane_id[MIPI_LANE_NUM];  
  
    union  
    {  
        short data_type[WDR_VC_NUM];  
    };  
} mipi_dev_attr_t;
```

【成员】

成员名称	描述
raw_data_type	传输的 raw data 比特数
lane_id	发送端(sensor)和接收端(MIPI Rx) lane 的对应关系 未使用的 lane 设置为-1
wdr_mode	MIPI WDR 模式
data_type	当 wdr_mode 为 HI_MIPI_WDR_MODE_DT 时，需要设置 data_type，表示不同曝光长度数据对应的 Data Type。

【芯片差异】

芯片类型	支持的 raw_data_type
Hi3516A	支持 10bit/12bit/14bit
Hi3518EV200	支持 8bit/10bit/12bit/14bit
Hi3519V100/Hi3519V101/Hi3516CV300	支持 8bit/10bit/12bit/14bit/16bit

芯片类型	wdr_mode
Hi3516A/Hi3518EV200	不支持设置 wdr_mode，默认使用 Virtual Channel 区分不同曝光长度的数据。



Hi3519V100/Hi3519 V101/Hi3516CV300	支持设置 wdr_mode, 除了支持 Virtual Channel 模式之外, 还支持 DataType WDR 和 Sony DOL 模式 WDR。
------------------------------------	---

【注意事项】

无。

【相关数据类型及接口】

- raw_data_type_e
- mipi_wdr_mode_e
- HI_MIPI_SET_DEV_ATTR

wdr_mode_e

【说明】

LVDS WDR 模式

【定义】

```
typedef enum
{
    HI_WDR_MODE_NONE      = 0x0,
    HI_WDR_MODE_2F        = 0x1,
    HI_WDR_MODE_3F        = 0x2,
    HI_WDR_MODE_4F        = 0x3,
    HI_WDR_MODE_DOL_2F    = 0x4,
    HI_WDR_MODE_DOL_3F    = 0x5,
    HI_WDR_MODE_DOL_4F    = 0x6,
    HI_WDR_MODE_BUTT
} wdr_mode_e;
```

【成员】

成员名称	描述
HI_WDR_MODE_NONE	线性模式
HI_WDR_MODE_2F	2 合一 WDR
HI_WDR_MODE_3F	3 合一 WDR
HI_WDR_MODE_4F	4 合一 WDR
HI_WDR_MODE_DOL_2F	DOL 模式 2 合一 WDR
HI_WDR_MODE_DOL_3F	DOL 模式 3 合一 WDR
HI_WDR_MODE_DOL_4F	DOL 模式 4 合一 WDR



【芯片差异】

无。

【注意事项】

- DOL WDR 模式需要配置为 HI_WDR_MODE_DOL_2F/
HI_WDR_MODE_DOL_3F/ HI_WDR_MODE_DOL_4F。
- Built-in WDR 模式和帧合成 WDR 模式都需要配置为 HI_WDR_MODE_NONE。

【相关数据类型及接口】

无。

lvds_sync_mode_e

【说明】

LVDS 同步方式。

表1-2 LVDS 同步方式

sync_mode	同步方式
LVDS_SYNC_MODE_SOL/ 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

【芯片差异】

芯片类型	定义
Hi3516A/Hi3518EV200	<pre>typedef enum { LVDS_SYNC_MODE_SOL = 0, LVDS_SYNC_MODE_SAV, LVDS_SYNC_MODE_BUTT } lvds_sync_mode_e;</pre>
Hi3519V100/Hi3519V101/Hi3516CV300	<pre>typedef enum { LVDS_SYNC_MODE_SOF = 0, LVDS_SYNC_MODE_SAV, LVDS_SYNC_MODE_BUTT } lvds_sync_mode_e;</pre>



【注意事项】

LVDS_SYNC_MODE_SOF 与 LVDS_SYNC_MODE_SOL 的含义相同，Hi3519V100 更名为 LVDS_SYNC_MODE_SOF。

【相关数据类型及接口】

无。

lvds_bit_endian

【说明】

比特位大小端模式

【定义】

```
typedef enum
{
    LVDS_ENDIAN_LITTLE  = 0x0,
    LVDS_ENDIAN_BIG     = 0x1,
    LVDS_ENDIAN_BUTT
}lvds_bit_endian;
```

【芯片差异】

无。

【注意事项】

无。

【相关数据类型及接口】

无。

lvds_vsync_type_e

【说明】

LVDS VSYNC 类型。

【定义】

```
typedef enum
{
    LVDS_VSYNC_NORMAL    = 0x00,
    LVDS_VSYNC_SHARE     = 0x01,
    LVDS_VSYNC_HCONNECT  = 0x02,
    LVDS_VSYNC_BUTT
} lvds_vsync_type_e;
```

【成员】



成员名称	描述
LVDS_VSYNC_NORMAL	长短曝光帧有独立的 SOF-EOF、SOL-EOL 或者 invalid SAV-invalid EAV, valid SAV-valid EAV。
LVDS_VSYNC_SHARE	长短曝光帧共用一对 SOF-EOF 标识, 短曝光的起始几行用固定值填充。
LVDS_VSYNC_HCONNECT	长短曝光帧共用一对 SAV-EAV 标识, 长短曝光帧之间是固定周期的消隐。

LVDS_VSYNC_SHARE 同步方式:

SOF	Long Exposure	EOL	Horizontal Blanking	SOL	Padding	EOL	Horizontal Blanking
SOL					Short Exposure		
	Padding					EOF	
SOV	V.BLK	EOV	-	SOV	V.BLK	EOV	-

LVDS_VSYNC_HCONNECT 同步方式:

SAV	Long Exposure frame	Horizontal Blanking(fix period)	V.BLK	Horizontal Blanking(fix period)	V.BLK	EAV	Horizontal Blanking
			Short ExposureFrame1		Short Exposure Frame2		
	V.BLK		V.BLK				
	V.BLK						

【芯片差异】

芯片类型	是否支持 lvds_vsync_type_e
Hi3516A/Hi3518EV200	不支持



Hi3519V100/Hi3519V101	支持
Hi3516CV300	支持

【注意事项】

无。

【相关数据类型及接口】

[lvds_vsync_type_t](#)

lvds_vsync_type_t

【说明】

LVDS vsync 参数

【定义】

```
typedef struct
{
    lvds\_vsync\_type\_e sync_type;

    unsigned short hblank1;
    unsigned short hblank2;
} lvds_vsync_type_t;
```

【芯片差异】

芯片类型	是否支持 lvds_vsync_type_t
Hi3516A/Hi3518EV200	不支持
Hi3519V100/Hi3519V101	支持
Hi3516CV300	支持

【注意事项】

当 sync_type 为 LVDS_VSYNC_HCONNECT 时，需要配置 hblank1 和 hblank2，表示 Hconnect 的消隐区长度。

【相关数据类型及接口】

[lvds_vsync_type_e](#)

lvds_fid_type_e

【说明】



Frame identification Id 类型

【定义】

```
typedef enum
{
    LVDS_FID_NONE      = 0x00,
    LVDS_FID_IN_SAV    = 0x01,    /* frame identification id in SAV 4th */
    LVDS_FID_IN_DATA   = 0x02,    /* frame identification id in first data
    */
    LVDS_FID_BUTT
} lvds_fid_type_e;
```

【成员】

成员名称	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，DOL 4 个字段的同步码需要将 fid_type 配置为 LVDS_FID_IN_SAV。
LVDS_FID_IN_DATA	FID 作为 Frame information column 插入在同步码之后的第一个像素之前，DOL 5 个字段的同步码需要将 fid_type 配置为 LVDS_FID_IN_DATA。

【芯片差异】

芯片类型	是否支持 lvds_fid_type_e
Hi3516A/Hi3518EV200	不支持
Hi3519V100/Hi3519V101	支持
Hi3516CV300	支持

【注意事项】

无。

【相关数据类型及接口】

无。

lvds_fid_type_t

【说明】

Frame indentification Id 配置信息。

【定义】



```
typedef struct
{
    lvds_vsync_type_e fid;

    HI_BOOL output_fil;
} lvds_fid_type_t;
```

【成员】

成员名称	描述
fid	LVDS DOL 模式下的 Frame identification Id 类型
output_fil	DOL 模式中的 Frame information line 紧接在 V-Blanking 之后输出，Frame ID 是 Frame information line 中的第一个像素值。 Frame information line 中并不包含有效的视频数据，如果 output_fil 设置为 HI_TRUE，Frame information line 会输出到后端设备。如果 output_fil 设置为 HI_FALSE，MIPI Rx 将丢弃这一行数据。

【芯片差异】

芯片类型	是否支持 lvds_fid_type_t
Hi3516A/Hi3518EV200	不支持
Hi3519V100/Hi3519V101	支持
Hi3516CV300	支持

【注意事项】

无。

【相关数据类型及接口】

lvds_fid_type_e

lvds_dev_attr_t

【说明】

LVDS/SubLVDS/HiSPi 设备属性。

【定义】

Hi3516A/Hi3518EV200:

```
typedef struct
{
```



```
img_size_t      img_size;
wdr_mode_e      wdr_mode;
lvds_sync_mode_e sync_mode;
raw_data_type_e raw_data_type;

lvds_bit_endian data_endian;
lvds_bit_endian sync_code_endian;
short           lane_id[LVDS_LANE_NUM];

unsigned short   sync_code[LVDS_LANE_NUM][WDR_VC_NUM][SYNC_CODE_NUM];
} lvds_dev_attr_t;
```

Hi3519V100/Hi3516CV300:

```
typedef struct
{
    img_size_t      img_size;
    raw_data_type_e raw_data_type;
    wdr_mode_e      wdr_mode;
    lvds_sync_mode_e sync_mode;
    lvds_vsync_type_t vsync_type;
    lvds_fid_type_t  fid_type;

    lvds_bit_endian data_endian;
    lvds_bit_endian sync_code_endian;
    short           lane_id[LVDS_LANE_NUM];

    unsigned short   sync_code[LVDS_LANE_NUM][WDR_VC_NUM][SYNC_CODE_NUM];
} lvds_dev_attr_t;
```

Hi3519V101:

```
typedef struct
{
    raw_data_type_e raw_data_type;
    wdr_mode_e      wdr_mode;
    lvds_sync_mode_e sync_mode;
    lvds_vsync_type_t vsync_type;
    lvds_fid_type_t  fid_type;

    lvds_bit_endian data_endian;
    lvds_bit_endian sync_code_endian;
    short           lane_id[LVDS_LANE_NUM];

    unsigned short   sync_code[LVDS_LANE_NUM][WDR_VC_NUM][SYNC_CODE_NUM];
```



```
} lvds_dev_attr_t;
```

【成员】

成员名称	描述
img_size	sensor 输入图像宽高，图像宽度必须是有效 lane 个数的整数倍。
wdr_mode	WDR 模式。
sync_mode	LVDS 同步模式。
raw_data_type	传输的 raw data 比特数。
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 未使用的 lane 设置为-1 lane id 的配置方式请参考“ Lane id 如何配置 ”。
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，当 wdr_mode 为 DOL 模式，并且 sync_mode 为 LVDS_SYNC_MODE_SAV 时，需要配置。

【芯片差异】

芯片类型	raw_data_type
Hi3516A	支持 10bit/12bit/14bit
Hi3518EV200	支持 8bit/10bit/12bit/14bit
Hi3519V100	支持 8bit/10bit/12bit/14bit/16bit
Hi3519V101	支持 8bit/10bit/12bit/14bit/16bit
Hi3516CV300	支持 8bit/10bit/12bit/14bit/16bit

芯片类型	vsync_type	fid_type
Hi3516A	不支持	不支持



Hi3518EV200	不支持	不支持
Hi3519V100	支持	支持
Hi3519V101	支持	支持
Hi3516CV300	支持	支持

芯片类型	img_size
Hi3516A	支持
Hi3518EV200	支持
Hi3519V100	支持
Hi3519V101	img_size 移入 combo_dev_attr_t 中的 img_rect。
Hi3516CV300	支持

【注意事项】

无。

【相关数据类型及接口】

- [wdr_mode_e](#)
- [lvds_sync_mode_e](#)
- [raw_data_type_e](#)
- [lvds_bit_endian](#)
- [lvds_vsync_type_t](#)
- [lvds_fid_type_t](#)
- [HI_MIPI_SET_DEV_ATTR](#)

phy_cmvp_e

【说明】

PHY 共模电压模式

【定义】

```
typedef enum
{
    PHY_CMV_GE900MV    = 0x00,
    PHY_CMV_LT900MV    = 0x01,
    PHY_CMV_BUTT
} phy_cmvp_e;
```



【成员】

成员名称	描述
PHY_CMV_GE900MV	PHY 共模电压大于等于 900mv
PHY_CMV_LT900MV	PHY 共模电压小于 900 mv

【芯片差异】

芯片类型	是否支持
Hi3516A	不支持
Hi3518EV200	不支持
Hi3519V100	支持
Hi3519V101	支持
Hi3516CV300	支持

【注意事项】

无。

【相关数据类型及接口】

无。

phy_cmv_t

【说明】

PHY 共模电压配置信息

【定义】

```
typedef struct
{
    COMBO_DEV devno;
    phy_cmv_e cmv_mode;
} phy_cmv_t;
```

【成员】

成员名称	描述
devno	MIPI Rx 设备号
cmv_mode	PHY 功能电压模式



【芯片差异】

芯片类型	是否支持
Hi3516A	不支持
Hi3518EV200	不支持
Hi3519V100	支持
Hi3519V101	支持
Hi3516CV300	支持

【注意事项】

无。

【相关数据类型及接口】

- [phy_cmv_e](#)
- [HI_MIPI_SET_PHY_CMVMODE](#)

combo_dev_attr_t

【说明】

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

【定义】

Hi3516A/Hi3518EV200:

```
typedef struct
{
    input_mode_t        input_mode;
    union
    {
        mipi_dev_attr_t    mipi_attr;
        lvds_dev_attr_t    lvds_attr;
    };
} combo_dev_attr_t;
```

Hi3519V100:

```
typedef struct
{
    COMBO_DEV            devno;
```




```
        input_mode_t        input_mode;
        phy_clk_share_e      phy_clk_share;

    union
    {
        mipi_dev_attr_t      mipi_attr;
        lvds_dev_attr_t      lvds_attr;
    };
} combo_dev_attr_t;
```

Hi3519V101:

```
typedef struct
{
    COMBO_DEV                devno;
    input_mode_t             input_mode;
    phy_clk_share_e          phy_clk_share;
    img_rect_t               img_rect;

    union
    {
        mipi_dev_attr_t      mipi_attr;
        lvds_dev_attr_t      lvds_attr;
    };
} combo_dev_attr_t;
```

Hi3516CV300:

```
typedef struct
{
    COMBO_DEV                devno;
    input_mode_t             input_mode;

    union
    {
        mipi_dev_attr_t      mipi_attr;
        lvds_dev_attr_t      lvds_attr;
    };
} combo_dev_attr_t;
```

【成员】



成员名称	描述
input_mode	输入接口类型
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_LVDS/ INPUT_MODE_HISPI, 则必须配置 lvds_attr
devno	MIPI Rx 设备号
phy_clk_share	PHY 共享时钟信息
img_rect	图像 crop 区域

【芯片差异】

芯片类型	devno	phy_clk_share
Hi3516A	不支持	不支持
Hi3518EV200	不支持	不支持
Hi3519V100	支持	支持 PHY1 和 PHY2 共享 PHY 的时钟
Hi3519V101	支持	支持 PHY1 和 PHY2 共享 PHY 的时钟
Hi3516CV300	支持	不支持

芯片类型	img_rect
Hi3516A	不支持
Hi3518EV200	不支持
Hi3519V100	不支持, 但可以通过 HI_MIPI_SET_CROP 设置 crop
Hi3519V101	支持
Hi3516CV300	不支持

【注意事项】

无。

【相关数据类型及接口】



无。

img_rect_t

【说明】

Mipi crop 属性。

【定义】

```
typedef struct
{
    int x;
    int y;
    unsigned int width;
    unsigned int height;
} img_rect_t;
```

【成员】

成员名称	描述
x	Crop 起始位置 x
y	Crop 起始位置 y
width	Crop 宽度
height	Crop 高度

【芯片差异】

芯片类型	CROP 接口
Hi3516A	不支持
Hi3518EV200	不支持
Hi3519V100	支持
Hi3519V101	支持
Hi3516CV300	不支持

【注意事项】

无。

【相关数据类型及接口】

[HI_MIPI_SET_CROP](#)



img_size_t

【说明】

图像宽高属性。

【定义】

```
typedef struct
{
    unsigned int width;
    unsigned int height;
} img_size_t;
```

【成员】

成员名称	描述
width	图像宽度
height	图像高度

【芯片差异】

无。

【注意事项】

无。

【相关数据类型及接口】

[HI_MIPI_SET_DEV_ATTR](#)

1.6 PROC 信息

PROC 调试信息在各个芯片中存在差异，但大部分调试信息都一致，此调试信息以 Hi3519V100 芯片的 proc 信息作为示例。MIPI_RX 正常工作状态下 proc 信息中宽高应该是稳定不变且和 sensor 输出时序的宽高匹配，并且 MIPI_RX 各种错误中断计数为 0。如果错误中断计数不为 0，请检查 MIPI_RX 相关属性是否配置正确。

【调试信息】

Module: [MIPI], Build Time: [May 24 2016, 22:40:41]

```
-----Combo DEV ATTR-----
Devno  WorkMode  DataType  WDRMode      LinkId  bEnCrop  ImgX  ImgY
ImgW   ImgH   SyncMode  DataEndian  SyncCodeEndian
      0    LVDS    RAW12      None      0, 1, 2    N      -      -
4248   2182   SAV      Big        Big
```



-----LINK INFO-----

LinkId	Count	LaneId	PhyData	AlignedData	ValidLane
0	3	0, 1, 2,-1	0x838492	0x5252f2	0, 1, 2
1	3	3, 4,-1, 5	0xe00025e6	0xa0003697	0, 1, 3
2	2	6, 7,-1,-1	0x5a9	0x3e28	0, 1

-----mipi detect info-----

Devno	VC	width	height
0	0	4248	2182
0	1	0	0
0	2	0	0
0	3	0	0

-----lvds detect info-----

Devno	Lane	LaneWidth
0	0	531
0	1	531
0	2	531
0	3	531
0	4	531
0	5	531
0	6	531
0	7	531

Devno	WDR_Frame	width	height
0	LEF	4248	2182
0	SEF1	0	0
0	SEF2	0	0
0	SEF3	0	0

-----fsm timeout and escape info-----

link	clkTOutCnt	d0TOutCnt	d1TOutCnt	d2TOutCnt	d3TOutCnt	clkEscCnt
d0EscCnt	d1EscCnt	d2EscCnt	d3EscCnt			
0	0	0	0	0	0	0
0	0	0				
1	0	0	0	0	0	0
0	0	0				
2	0	0	0	0	0	0
0	0	0				

-----ALING Err info-----

Devno	FIFO_FullErr	Lane0Err	Lane1Err	Lane2Err	Lane3Err	Lane4Err	
	Lane5Err	Lane6Err	Lane7Err	Lane8Err	Lane9Err	Lane10Err	Lane11Err

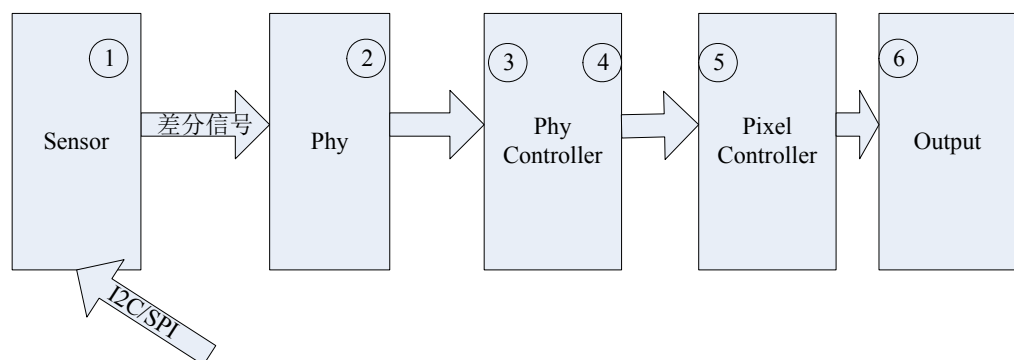


0 0 0 0 0 0 0 0 0
0 0 0 0 0 0

【调试信息分析】

- MIPI_Rx 通过 Phy 接收 sensor 的差分数据，Phy Controller 检测到同步头后，将每条 lane 上的数据对齐；
- Pixel Controller 解析同步信息并按照 raw data 的位宽将 lane 上面的数据合并为 Pixel 数据；Output 模式将 Pixel 数据发送给后级模块。
- Phy PhyController PixelController 由 sensor 的 pixel clk 提供时钟，output 模块的时钟为称为随路时钟，与后级模块的工作时钟相同。MIPI_Rx 的 crop 功能在 Pixel Controller 的末端实现，所以 Crop 后可以降低需要的随路时钟。

图1-3 MIPI 数据流



【参数说明】

参数		描述
Combo DEV ATTR	Devno	MIPI 设备号
	WorkMode	• MIPI 设备工作模式： LVDS/MIPI/CMOS 等模式
	DataType	• 数据类型： RAW8 / RAW10/ RAW12/ RAW14/ RAW16 bit 等类型



参数		描述
	WDRMode	WDR 模式： <ul style="list-style-type: none">• None: 非 WDR 模式• 2To1: 2 合 1 WDR• 3To1: 3 合 1 WDR• 4To1: 4 合 1 WDR• DOL2To1: DOL2 合 1WDR• DOL3To1: DOL3 合 1WDR• DOL4To1: DOL4 合 1WDR
	LinkId	此设备使用了哪几个 Link，对应 Link ID。 一个物理 Link 对应 4 个 lane。
	bEnCrop	Crop 是否使能。N 表示 Crop 被关闭，Y 表示 Crop 使能。
	ImgX	Crop 图像起始 X
	ImgY	Crop 图像起始 Y
	ImgW	Crop 图像宽度
	ImgH	Crop 图像高度
	SyncMode	同步头模式 SOF: 同步方式为 SOF, EOF, SOL, EOL SAV: 同步方式为 invalid SAV, invalid EAV, valid SAV, valid EAV
	DataEndian	数据大小端： <ul style="list-style-type: none">• Big: 大端模式• Little: 小端模式
	SyncCodeEndian	同步头大小端： <ul style="list-style-type: none">• Big: 大端模式• Little: 小端模式
LINK INFO	LinkIdx	Link ID 序号
	LaneCount	该 Link 中使用了几条 Lane
	LaneId	对应的 Lane Id
	PhyData	PHY 接收到的实时数据
	AlignedData	检测到帧同步信号后的实时数据
	ValidLane	Link 内部有效 Lane ID。



参数		描述
mipi detect info (仅 MIPI 模式下可见)	Devno	MIPI_Rx 设备号
	VC	Virtual Channel
	width	MIPI 控制器检测到的图像总宽度
	height	MIPI 控制器检测到的图像总高度
lvds detect info (仅 LVDS/Su bLVDS/Hi SPi 模式下可见)	Devno	MIPI_Rx 设备号
	Lane	Lane ID 序号
	LaneWidth	对应 lane 检测到的图像宽度
	Devno	MIPI_Rx 设备号
	WDR_Frame	表示 WDR 模式下的长短帧，线性模式不显示此列。 <ul style="list-style-type: none"> • LEF: 长曝光帧 • SEF1: 短曝光帧 1 • SEF2: 短曝光帧 2 • SEF3: 短曝光帧 3
	width	LVDS 控制器检测到的图像总宽度
	height	LVDS 控制器检测到的图像总高度
fsm timeout and escape info (仅 MIPI 模式下可见)	link	Link ID
	clkTOutCnt	时钟 lane 从 LP 切换到 HS 超时
	d0TOutCnt	数据 lane0 从 LP 切换到 HS 超时
	d1TOutCnt	数据 lane1 从 LP 切换到 HS 超时
	d2TOutCnt	数据 lane2 从 LP 切换到 HS 超时
	d3TOutCnt	数据 lane3 从 LP 切换到 HS 超时
	clkEscCnt	时钟 lane 切换到 escape 模式超时
	d0EscCnt	数据 lane0 切换到 escape 模式超时
	d1EscCnt	数据 lane0 切换到 escape 模式超时
	d2EscCnt	数据 lane0 切换到 escape 模式超时
	d3EscCnt	数据 lane0 切换到 escape 模式超时
ALING Err info	Devno	MIPI 设备号
	FIFO_FullErr	FIFO 溢出
	Lane0Err	Lane0 FIFO 溢出



参数		描述
	Lane1Err	Lane1 FIFO 溢出
	Lane2Err	Lane2 FIFO 溢出
	Lane3Err	Lane3 FIFO 溢出
	Lane4Err	Lane4 FIFO 溢出
	Lane5Err	Lane5 FIFO 溢出
	Lane6Err	Lane6 FIFO 溢出
	Lane7Err	Lane7 FIFO 溢出
	Lane8Err	Lane8 FIFO 溢出
	Lane9Err	Lane9 FIFO 溢出
	Lane10Err	Lane10 FIFO 溢出
	Lane11Err	Lane11 FIFO 溢出

1.7 FAQ

MIPI 具体规格请参考芯片手册《Hi35xx 专业型 HD IP Camera Soc 用户指南.pdf》和《Features of the Video Interfaces of HiSilicon IP Cameras.pdf》

Lane id 如何配置

Lane id 的配置对应 `mipi_dev_attr_t` 中的 `short lane_id[MIPI_LANE_NUM]` 或者 `lvds_dev_attr_t` 中的 `short lane_id[LVDS_LANE_NUM]`。

对接 sensor 时，未使用的 lane 将其对应的 lane_id 配置为-1。配置 lane_id 还可以调整数据通道顺序，根据硬件单板与实际 sensor 输出通道的对应关系调整 lane_id 的配置。

下面以 demo 板 mn34220 为例进行说明，例如 demo 板的 MIPI_Rx 引脚与 MN34220 的引脚硬件连接如表 1-3 所示。

表1-3 MN34220 与 MIPI_Rx 管脚关系

MN34220 管脚	Hi3516A MIPI 管脚
SENSOR_SDODA0M	MIPI0_D0M
SENSOR_SDODA0P	MIPI0_D0P
SENSOR_SDODA1M	MIPI0_D1M
SENSOR_SDODA1P	MIPI0_D1P

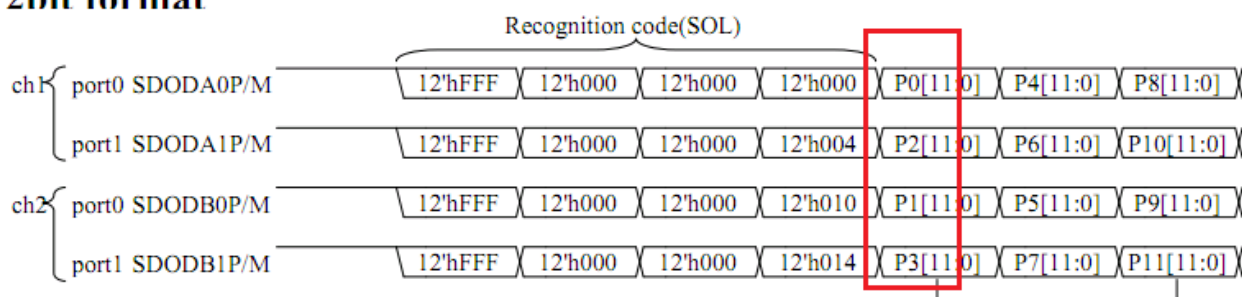


SENSOR_SDODB0M	MIPI1_D0M
SENSOR_SDODB0P	MIPI1_D0P
SENSOR_SDODB1M	MIPI1_D1M
SENSOR_SDODB1P	MIPI1_D1P

引脚上实际传输数据的顺序为 Pixel0, Pixel2, Pixel1, Pixel3, 见图 1-4。

图1-4 MN34220 Output 时序图 (2ch 2port 12bit format)

12bit format



由于 sensor 没有输出数据到 MIPI_Rx 的 MIPI0_D2M, MIPI0_D2P, MIPI0_D3M, MIPI0_D3P, MIPI1_D2M, MIPI1_D2P, MIPI1_D3M, MIPI1_D3P, 需要将对于的 lane_id 配置为-1, 所以所以 lane_id 配置如下:

lane_id = {0, 2, -1, -1, 1, 3, -1, -1}

Sync_code 配置是根据 lane_id 生效的, 如果 lane_id 为-1, 则对应 sync_code 不会生效。

1.7.2 LVDS mode sync code 如何配置

LVDS/SUB_LVDS 的同步方式有两种模式, 参考 [lvds_sync_mode_e](#):

- LVDS_SYNC_MODE_SOF/LVDS_SYNC_MODE_SOL
- LVDS_SYNC_MODE_SAV

不同 sensor 或者同一 sensor 不同的传输模式都需要配置不同的 sync code。

sync_code 的定义如下:

- unsigned short
sync_code[LVDS_LANE_NUM][WDR_VC_NUM][SYNC_CODE_NUM];
需要根据 sensor datasheet 进行配置, 如表 1-4 所示。

表1-4 sync_code 定义

sync_code 中的元素	每一项的含义
LVDS_LANE_NUM	对应 lvds 硬件物理通道。



WDR_VC_NUM	WDR 通道数，如 2 合 1WDR 对应 2 个 WDR 通道，最多为 4 个。
SYNC_CODE_NUM	每条 lane 的 Sync_code 由 4 个码字组成，在不同的同步模式下对应的含义不同，请参考表 1-2。

说明
每一 SOF/EOF/SOL/EOL 的 sync_code 前三个为固定码字：0xFFFF 0x0000 0x0000

Hi3516A/Hi3518EV200 同步码配置举例

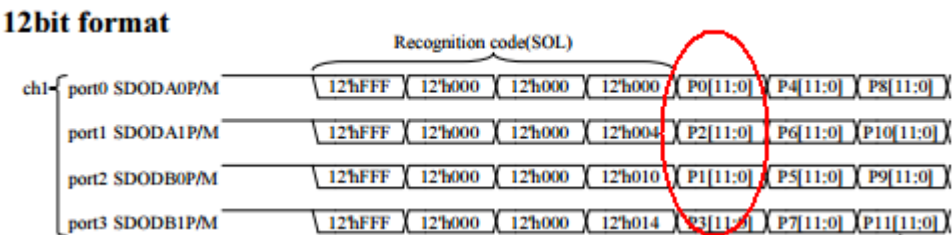
- Sync code 配置的原则：
- 同一个 Link 内乱序时，无论硬件以何种顺序与 Sensor 连接，Sync code 都按照正常顺序配置。
 - 两个 Link 之间乱序时，仍然遵循上述准则，即在同一 Link 内保持正常顺序配置。Panasonic 的 Sensor 普遍存在两个 Link 之间乱序，其他厂家暂不涉及。
- 举例：
- 举例 1：同一 Link 内乱序
MN34220 工作在 1 channel 4port(4 lane) 12bit 模式时，datasheet 中关于 sync code 的描述，如图 1-5 所示。

图1-5 同步码

ch	port	Name	Code (12bit×4)				ch	port	Name	Code (12bit×4)			
ch1	Port0	SOF	FFFh	000h	000h	002h	ch1	Port2	SOF	FFFh	000h	000h	012h
		SOL	FFFh	000h	000h	000h			SOL	FFFh	000h	000h	010h
		EOL	FFFh	000h	000h	001h			EOL	FFFh	000h	000h	011h
		EOF	FFFh	000h	000h	003h			EOF	FFFh	000h	000h	013h
	Port1	SOF	FFFh	000h	000h	006h		Port3	SOF	FFFh	000h	000h	016h
		SOL	FFFh	000h	000h	004h			SOL	FFFh	000h	000h	014h
		EOL	FFFh	000h	000h	005h			EOL	FFFh	000h	000h	015h
		EOF	FFFh	000h	000h	007h			EOF	FFFh	000h	000h	017h

各通道像素格式，如图 1-6 所示。

图1-6 像素格式图





所以相当于正常的顺序应该为 SDODA0、SDODB0、SDODA1、SDODB1，只有前面 4 个通道的 sync code 配置有效。应该按照这个顺序在 mipi 接口中配置 Sync code。则 sync_code 配置为：

```
.sync_code = {  
    {{0x002, 0x003, 0x000, 0x001}}, //PHY0_lane0  
    {0x202, 0x203, 0x200, 0x201},  
    {0x102, 0x103, 0x100, 0x101},  
    {0x302, 0x303, 0x300, 0x301}},  
  
    {{0x012, 0x013, 0x010, 0x011}}, //PHY0_lane1  
    {0x212, 0x213, 0x210, 0x211},  
    {0x112, 0x113, 0x110, 0x111},  
    {0x312, 0x313, 0x310, 0x311}},  
  
    {{0x006, 0x007, 0x004, 0x005}}, //PHY0_lane2  
    {0x206, 0x207, 0x204, 0x205},  
    {0x106, 0x107, 0x104, 0x105},  
    {0x306, 0x307, 0x304, 0x305}},  
  
    {{0x016, 0x017, 0x014, 0x015}}, //PHY0_lane3  
    {0x216, 0x217, 0x214, 0x215},  
    {0x116, 0x117, 0x114, 0x115},  
    {0x316, 0x317, 0x314, 0x315}},  
  
    {{0x00a, 0x00b, 0x008, 0x009}}, //PHY1_lane0  
    {0x20a, 0x20b, 0x208, 0x209},  
    {0x10a, 0x10b, 0x108, 0x109},  
    {0x30a, 0x30b, 0x308, 0x309}},  
  
    {{0x00a, 0x00b, 0x008, 0x009}}, //PHY1_lane1  
    {0x20a, 0x20b, 0x208, 0x209},  
    {0x10a, 0x10b, 0x108, 0x109},  
    {0x30a, 0x30b, 0x308, 0x309}},  
  
    {{0x01a, 0x01b, 0x018, 0x019}}, //PHY1_lane2  
    {0x21a, 0x21b, 0x218, 0x219},  
    {0x11a, 0x11b, 0x118, 0x119},  
    {0x31a, 0x31b, 0x318, 0x319}},  
  
    {{0x01a, 0x01b, 0x018, 0x019}}, //PHY1_lane3  
    {0x21a, 0x21b, 0x218, 0x219},  
    {0x11a, 0x11b, 0x118, 0x119},  
    {0x31a, 0x31b, 0x318, 0x319}}  
}
```



- 举例 2：两个 Link 之间乱序

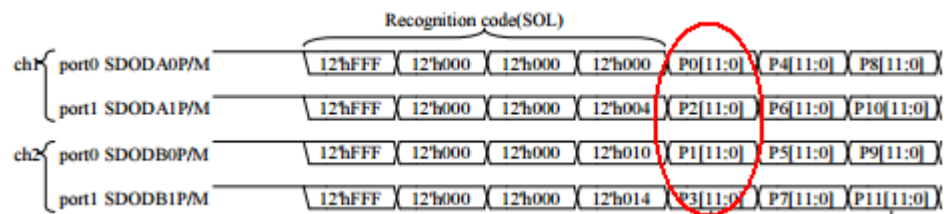
MN34220 工作在 2 channel 2port(4 lane) 12bit 模式时，datasheet 中关于 sync code 的描述，如图 1-7 同步码。

图1-7 同步码

ch	port	Name	Code (12bit×4)				ch	port	Name	Code (12bit×4)			
ch1	Port0	SOF	FFFh	000h	000h	002h	ch2	Port0	SOF	FFFh	000h	000h	012h
		SOL	FFFh	000h	000h	000h			SOL	FFFh	000h	000h	010h
		EOL	FFFh	000h	000h	001h			EOL	FFFh	000h	000h	011h
		EOF	FFFh	000h	000h	003h			EOF	FFFh	000h	000h	013h
	Port1	SOF	FFFh	000h	000h	006h		Port1	SOF	FFFh	000h	000h	016h
		SOL	FFFh	000h	000h	004h			SOL	FFFh	000h	000h	014h
		EOL	FFFh	000h	000h	005h			EOL	FFFh	000h	000h	015h
		EOF	FFFh	000h	000h	007h			EOF	FFFh	000h	000h	017h

各通道像素格式，如图 1-8 所示。

图1-8 像素格式图



由于在两个 Link 之间线序有交叉，但是根据上述原则，只在同一 Link 内部考虑 Sync code 的配置。

Link0 的通道 0，通道 1 和 Link1 的通道 0，通道 1 配置的 sync code 有效。所以最终的 sync_code 配置为：

```
.sync_code = {  
    {{0x002, 0x003, 0x000, 0x001}, //PHY0_lane0  
    {0x202, 0x203, 0x200, 0x201},  
    {0x102, 0x103, 0x100, 0x101},  
    {0x302, 0x303, 0x300, 0x301}},  
  
    {{0x006, 0x007, 0x004, 0x005}, //PHY0_lane1  
    {0x206, 0x207, 0x204, 0x205},  
    {0x106, 0x107, 0x104, 0x105},  
    {0x306, 0x307, 0x304, 0x305}},  
  
    {{0x00a, 0x00b, 0x008, 0x009}, //PHY0_lane2  
    {0x20a, 0x20b, 0x208, 0x209},  
    {0x10a, 0x10b, 0x108, 0x109},  
    {0x30a, 0x30b, 0x308, 0x309}},  
};
```



```

{{0x00a, 0x00b, 0x008, 0x009}, //PHY0_lane3
{0x20a, 0x20b, 0x208, 0x209},
{0x10a, 0x10b, 0x108, 0x109},
{0x30a, 0x30b, 0x308, 0x309}},

{{0x012, 0x013, 0x010, 0x011}, //PHY1_lane0
{0x212, 0x213, 0x210, 0x211},
{0x112, 0x113, 0x110, 0x111},
{0x312, 0x313, 0x310, 0x311}},

{{0x016, 0x017, 0x014, 0x015}, //PHY1_lane1
{0x216, 0x217, 0x214, 0x215},
{0x116, 0x117, 0x114, 0x115},
{0x316, 0x317, 0x314, 0x315}},

{{0x01a, 0x01b, 0x018, 0x019}, //PHY1_lane2
{0x21a, 0x21b, 0x218, 0x219},
{0x11a, 0x11b, 0x118, 0x119},
{0x31a, 0x31b, 0x318, 0x319}},

{{0x01a, 0x01b, 0x018, 0x019}, //PHY1_lane3
{0x21a, 0x21b, 0x218, 0x219},
{0x11a, 0x11b, 0x118, 0x119},
{0x31a, 0x31b, 0x318, 0x319}}
}

```

Hi3519V100/Hi3519V101/Hi3516CV300 同步码配置举例

- Sync code 配置的原则：
根据实际使用多少个 lane，配置 sync_code 中前几个 lane 的同步码。如果 lane 乱序，Sync code 按照正常顺序配置。
- 举例：
MN34220 工作在 2 channel 2port(4 lane) 12bit 模式时，datasheet 中关于 sync code 的描述，如图 1-9 所示。

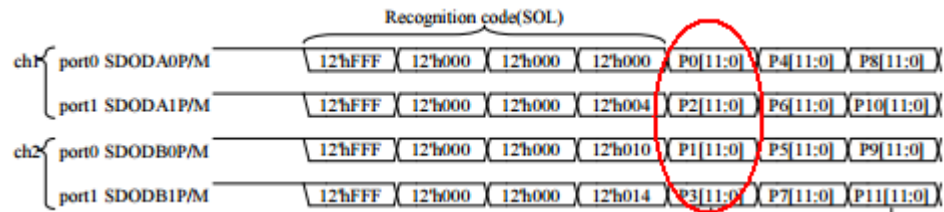
图1-9 同步码

ch	port	Name	Code (12bit×4)				ch	port	Name	Code (12bit×4)			
ch1	Port0	SOF	FFFh	000h	000h	002h	ch2	Port0	SOF	FFFh	000h	000h	012h
		SOL	FFFh	000h	000h	000h			SOL	FFFh	000h	000h	010h
		EOL	FFFh	000h	000h	001h			EOL	FFFh	000h	000h	011h
		EOF	FFFh	000h	000h	003h			EOF	FFFh	000h	000h	013h
	Port1	SOF	FFFh	000h	000h	006h		Port1	SOF	FFFh	000h	000h	016h
		SOL	FFFh	000h	000h	004h			SOL	FFFh	000h	000h	014h
		EOL	FFFh	000h	000h	005h			EOL	FFFh	000h	000h	015h
		EOF	FFFh	000h	000h	007h			EOF	FFFh	000h	000h	017h



各通道像素格式，如图 1-10 所示。

图1-10 像素格式图



同步码的配置为:

```
.sync_code = {
    // lane0 (mn34220 chn1 port0)
    { {0x002, 0x003, 0x000, 0x001}, //VC0
      {0x202, 0x203, 0x200, 0x201}, // VC1
      {0x102, 0x103, 0x100, 0x101}, // VC2
      {0x302, 0x303, 0x300, 0x301} // VC3
    },
    // lane1 (mn34220 chn2 port0)
    { {0x012, 0x013, 0x010, 0x011},
      {0x212, 0x213, 0x210, 0x211},
      {0x112, 0x113, 0x110, 0x111},
      {0x312, 0x313, 0x310, 0x311}
    },
    // lane2 (mn34220 chn1 port1)
    { {0x006, 0x007, 0x004, 0x005},
      {0x206, 0x207, 0x204, 0x205},
      {0x106, 0x107, 0x104, 0x105},
      {0x306, 0x307, 0x304, 0x305}
    },
    // lane3 (mn34220 chn2 port1)
    { {0x016, 0x017, 0x014, 0x015},
      {0x216, 0x217, 0x214, 0x215},
      {0x116, 0x117, 0x114, 0x115},
      {0x316, 0x317, 0x314, 0x315}
    }
}
```



1.7.3 MIPI 频率说明

MIPI lane 频率与 VI 频率关系

使用 MIPI 多个 lanes 进行数据传输，mipi lane 的传输频率与 VI 处理频率如何对应，每一 lane 可传输的最高速率如何计算？

- MIPI_Rx 使用多 lane 接收数据，会转成内部时序，送给 VI 模块进行处理，多 lane 传输的数据总量不变，有这样的计算公式：

$$VI_Freq * Pix_Width = Lane_Num * MIPI_Freq$$

- 其中，VI_Freq 为 VI 的工作时钟，Pix_Width 为像素位宽，Lane_Num 为传输 lane 个数，MIPI_Freq 为一个 lane 能接收的最大频率。
- 下面以 VI 工作频率为 250M，MIPI 数据为 RAW 12, 4Lane 传输为例进行说明：

$$MIPI_Freq = (250 * 12) / 4 = 750$$

即每个 lane 最高频率为 750MHz

1.7.4 Sensor 复位

- sensor reset/unreset 操作是在 mipi 驱动中实现的，mipi 驱动中的复位顺序如下：
 - sensor reset
 - mipi core reset
 - config mipi attr
 - mipi core unreset
 - sensor unreset
- 芯片有一个 pin 脚专门用于 sensor 复位(SENSOR_RSTN)，建议客户默认使用。
- 如果需要修改 sensor reset pin，需要注意 mipi 驱动做适配修改，根据实际使用的管脚，在 mipi 驱动。
- mipi_reset_sensor/mipi_unreset_sensor 或者 mipi_drv_reset_sensor/mipi_drv_unreset_sensor 函数中进行适配。
- 如果 sensor 支持 standby 模式，不希望进行 reset sensor 操作，可以将 mipi_reset_sensor/mipi_unreset_sensor 或者 mipi_drv_reset_sensor/mipi_drv_unreset_sensor 操作去掉，sensor reset 改成 enable sensor standby 模式，sensor unreset 改成 disable standby 模式。