HiMPP MIPI

User Guide

Issue	00B03
Date	2016-09-22

Trademarks and Permissions

Notice

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

HiSilicon Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://www.hisilicon.com

Email: support@hisilicon.com

About This Document

Related Versions

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

Product Name	Version
Hi3516A	V100
Hi3516D	V100
Hi3518E	V200
Hi3518E	V201
Hi3516C	V200
Hi3519	V100
Hi3519	V101
Hi3516C	V300
Hi3559	V100

NOTE

- Unless otherwise specified, descriptions about the Hi3516A also apply to the Hi3516D.
- Unless otherwise specified, descriptions about Hi3518E V200 also apply to Hi3518E V201 and Hi3516C V200.
- Unless otherwise specified, descriptions about the Hi3519 V101 also apply to the Hi3559 V100.

Intended Audience

This document is intended for:

- Technical support engineers
- Software development engineers

Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
⚠ DANGER	Alerts you to a high risk hazard that could, if not avoided, result in serious injury or death.
⚠ WARNING	Alerts you to a medium or low risk hazard that could, if not avoided, result in moderate or minor injury.
⚠ CAUTION	Alerts you to a potentially hazardous situation that could, if not avoided, result in equipment damage, data loss, performance deterioration, or unanticipated results.
TIP	Provides a tip that may help you solve a problem or save time.
NOTE	Provides additional information to emphasize or supplement important points in the main text.

Change History

Changes between document issues are cumulative. Therefore, the latest document issue contains all changes made in previous issues.

Issue 00B03 (2016-09-22)

This issue is the third draft release, which incorporates the following change:

The contents related to Hi3559 V100 are added.

Issue 00B02 (2016-07-12)

This issue is the second draft release, which incorporates the following change:

The contents related to Hi3516C V300 are added.

Chapter 1 HiMPP MIPI User Guide

In section 1.4, the description of HI_MIPI_SET_DEV_ATTR is modified. The APIs HI_MIPI_RESET_SENSOR, HI_MIPI_UNRESET_SENSOR, HI_MIPI_RESET_MIPI, and HI_MIPI_UNRESET_MIPI are added.

In section 1.5, the data structures COMBO_DEV, HI_MIPI_IOC_MAGIC, phy_cmv_e, and phy_cmv_t are added.

Issue 00B01 (2016-06-20)

This issue is the first draft release.

Contents

1 HiMPP MIPI User Guide

1.1 Overview

The mobile industry processor interface (MIPI) RX receives raw video data by using low-voltage differential signals, converts the received serial differential signals into digital camera (DC) timings, and then transmit the timings to the downstream Video Capture (VICAP) module.

The MIPI RX supports the MIPI D-PHY, LVDS, and high-speed serial pixel interface (HiSPi) serial video signal inputs and is compatible with the DC video interface.

1.2 Important Concepts

- MIPI

  The MIPI described in this document refers to the communications interface which uses the D-PHY transmission specifications at the physical layer and CSI-2 at the protocol layer.

- LVDS

  The low-voltage differential signaling (LVDS) technology differentiates blanking regions and valid data by using the sync code.

- Lane

  A lane is a high-speed differential pair for connecting the TX end and RX end. It can be a clock lane or a data lane.

- Link

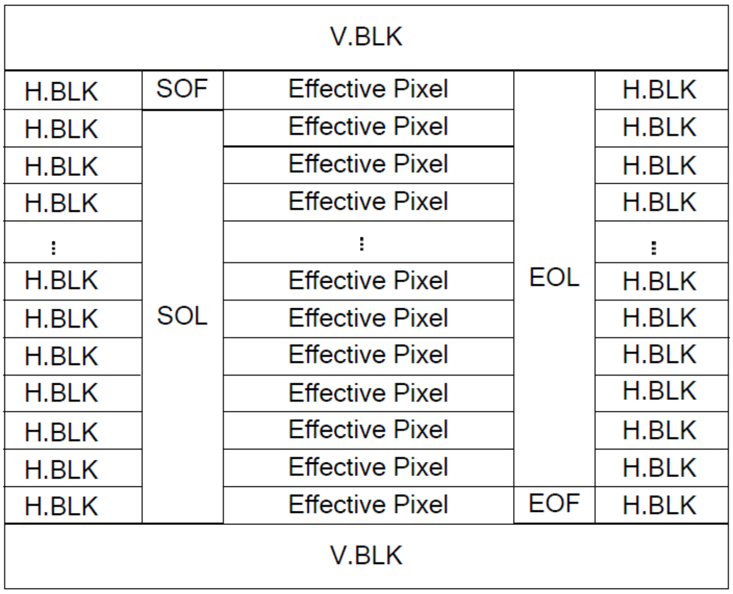
  A link consists of a clock lane and at least one data lane between the TX end and RX end.

- Sync code

  The MIPI interface uses the short packet in CSI-2 for synchronization, and the LVDS uses the sync code to differentiate valid data and blanking regions. There are two sync modes for the LVDS:

  - SOF and EOF indicate the start and end of a frame respectively, and SOL and EOL indicate the start and end of a line respectively. Figure 1-1 shows the sync mode.

Figure 1-1 SOF/EOF/SOL/EOL sync mode

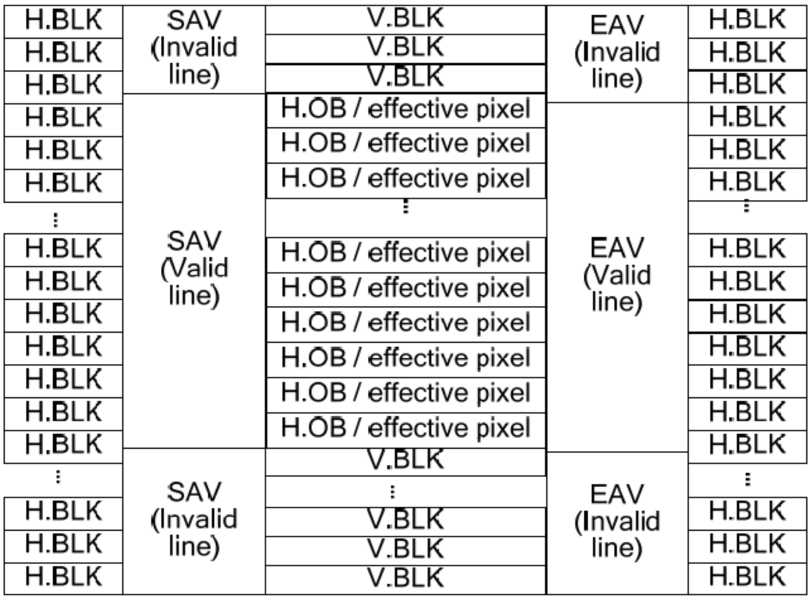


- SAV(invalid) and EAV(invalid) indicate the start and end of invalid data in the blanking region respectively, and SAV(valid) and EAV(valid) indicate the start and end of valid pixel data respectively.

  Each sync code consists of four fields. The bit width of each field is consistent with that of pixel data. The first three fields are fixed reference codewords, and the fourth field is defined by the sensor vendor.

  Because the sync code differs according to the sensor, it needs to be configured based on the sensor. Figure 1-2 shows the sync mode.

Figure 1-2 SAV/EAV sync mode



- DOL

  Digital overlap (DOL) indicates the WDR function of Sony.

1.3 Function Description

The MIPI RX is a collection unit that supports multiple differential video input interfaces. It uses the combo-PHY to receive data through the MIPI, LVDS, sub-LVDS, HiSPi, or DC interface. Depending on the functional mode configuration, the MIPI RX allows data transmission at different speeds and resolutions and supports multiple external input devices.

Table 1-1 Maximum number of lanes

Chip	Maximum Number of Lanes Supported by the MIPI RX
Hi3516A	1-link/4-lane MIPI inputs or 2-link/8-lane LVDS inputs
Hi3518E V200/Hi3516C V300	1-link/4-lane MIPI inputs or 1-link/4-lane LVDS inputs
Hi3519 V100/Hi3519 V101	2-link/8-lane MIPI inputs or 3-link/12-lane LVDS inputs

The input pins can be multiplexed to support single-ended DC or BT.1120 channel inputs, thereby providing better compatibility with fewer chip pins.

1.4 API Reference

The MIPI RX provides the function of interworking with sensor timings. It provides the ioctl interface and the following APIs:

- HI_MIPI_SET_DEV_ATTR: Configures the MIPI device attributes.
- HI_MIPI_SET_PHY_CMVMODE: Configures the common-mode voltage mode.
- HI_MIPI_SET_CROP: Configures the MIPI crop attribute.
- HI_MIPI_RESET_SENSOR: Resets the sensor.
- HI_MIPI_UNRESET_SENSOR: Deasserts the reset on the sensor.
- HI_MIPI_RESET_MIPI: Resets the MIPI RX.
- HI_MIPI_UNRESET_MIPI: Deasserts the reset on the MIPI RX.

HI_MIPI_SET_DEV_ATTR

[Description]

Configures the attributes of the MIPI RX device.

[Definition]

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

[Parameter]

combo_dev_attr_t pointer

[Return Values]

Return Value	Description
0	Success
–1	Failure. errno is configured.

[Chip Difference]

Chip	Internal Process
Hi3516A	The interface contains the following procedures: 1. Reset the sensor. 2. Reset the MIPI RX. 3. Deassert the reset on the MIPI RX. 4. Configure MIPI RX parameters. 5. Deassert the reset on the sensor.
Hi3518E V200	
Hi3519 V100	Configure MIPI RX parameters.
Hi3519 V101	
Hi3516C V300	

[Requirement]

Header file: hi_mipi.h

[Note]

For Hi3519 V100, Hi3519 V101, and Hi3516C V300:

- The following APIs also need to be configured besides HI_MIPI_SET_DEV_ATTR.
- The operations of resetting the sensor, deasserting the reset on the sensor, resetting the MIPI RX, and deasserting the reset on the MIPI RX are performed by calling the following independent APIs:
  - Resetting the sensor: HI_MIPI_RESET_SENSOR
  - Deasserting the reset on the sensor: HI_MIPI_UNRESET_SENSOR
  - Resetting the MIPI RX: HI_MIPI_RESET_MIPI
  - Deasserting the reset on the MIPI RX: HI_MIPI_UNRESET_MIPI
- The recommended configuration process is as follows:
  1. Reset the MIPI RX.
  2. Reset the sensor.
  3. Configure the attributes of the MIPI RX device.
  4. Deassert the reset on the MIPI RX.

5. Deassert the reset on the sensor.

[See Also]

- HI_MIPI_RESET_SENSOR
- HI_MIPI_UNRESET_SENSOR
- HI_MIPI_RESET_MIPI
- HI_MIPI_UNRESET_MIPI

HI_MIPI_SET_PHY_CMVMODE

[Description]

Configures the common-mode voltage mode.

[Definition]

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

[Parameter]

phy_cmv_t pointer

[Return Values]

Return Value	Description
0	Success
–1	Failure. errno is configured.

[Chip Difference]

Chip	Supported or Not
Hi3516A	Not supported
Hi3518E V200	Not supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Supported

[Requirement]

Header file: hi_mipi.h

[Note]

None

HI_MIPI_SET_CROP

[Description]

Configures the MIPI crop attribute.

[Definition]

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

[Parameter]

img_rect_t pointer

[Return Values]

Return Value	Description
0	Success
–1	Failure. errno is configured.

[Chip Difference]

Chip	Supported or Not
Hi3516A	Not supported
Hi3518E V200	Not supported
Hi3519 V100	Supported
Hi3519 V101	Not supported
Hi3516C V300	Not supported

[Requirement]

Header file: hi_mipi.h

[Note]

- The crop operation takes effect only if this API is called after HI_MIPI_SET_DEV_ATTR. (The crop function is disabled by default when HI_MIPI_SET_DEV_ATTR is called.)
- The crop attribute needs to be configured based on the actual output width and height of the sensor. (x+width, y+height) cannot exceed the actual image resolution.
- The width of the cropped image must be an integral multiple of the number of valid lanes.

HI_MIPI_RESET_SENSOR

[Description]

Resets the sensor.

[Definition]

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

[Parameter]

COMBO_DEV, indicating the device ID

[Return Values]

Return Value	Description
0	Success
–1	Failure. errno is configured.

[Chip Difference]

Chip	Supported or Not
Hi3516A	Not supported
Hi3518E V200	Not supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Supported

[Requirement]

Header file: hi_mipi.h

[Note]

None

HI_MIPI_UNRESET_SENSOR

[Description]

Deasserts the reset on the sensor.

[Definition]

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

[Parameter]

COMBO_DEV, indicating the device ID

[Return Values]

Return Value	Description
0	Success
–1	Failure. errno is configured.

[Chip Difference]

Chip	Supported or Not
Hi3516A	Not supported
Hi3518EV200	Not supported
Hi3519V100	Supported
Hi3519V101	Supported
Hi3516CV300	Supported

[Requirement]

Header file: hi_mipi.h

[Note]

None

HI_MIPI_RESET_MIPI

[Description]

Resets the MIPI RX.

[Definition]

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

[Parameter]

COMBO_DEV, indicating the device ID

[Return Values]

Return Value	Description
0	Success
–1	Failure. errno is configured.

[Chip Difference]

Chip	Supported or Not
Hi3516A	Not supported
Hi3518EV200	Not supported
Hi3519V100	Supported
Hi3519V101	Supported
Hi3516CV300	Supported

[Requirement]

Header file: hi_mipi.h

[Note]

None

HI_MIPI_UNRESET_MIPI

[Description]

Deasserts the reset on the MIPI RX.

[Definition]

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

[Parameter]

COMBO_DEV, indicating the device ID

[Return Values]

Return Value	Description
0	Success
–1	Failure. errno is configured.

[Chip Difference]

Chip	Supported or Not
Hi3516A	Not supported
Hi3518EV200	Not supported
Hi3519V100	Supported
Hi3519V101	Supported
Hi3516CV300	Supported

[Requirement]

Header file: hi_mipi.h

[Note]

None

1.5 Data Structures

The MIPI RX provides the following data structures:

- HI_MIPI_IOC_MAGIC: Defines the magic number of the MIPI RX ioctl command.
- COMBO_DEV: Defines the type of the MIPI RX device.
- COMBO_MAX_LINK_NUM: Specifies the maximum number of links supported by the device.
- LANE_NUM_PER_LINK: Specifies the maximum number of lanes supported by each link.
- COMBO_MAX_LANE_NUM: Specifies the maximum number of lanes supported by the device.
- MIPI_LANE_NUM: Specifies the number of lanes supported by the MIPI interface.
- LVDS_LANE_NUM: Specifies the number of lanes supported by the LVDS/HiSPi interface.
- COMBO_MAX_DEV_NUM: Specifies the number of the MIPI RX devices.
- WDR_VC_NUM: Specifies the maximum number of supported virtual channels.
- SYNC_CODE_NUM: Specifies the number of sync codes supported by each virtual channel.
- input_mode_t: Specifies the type of the MIPI RX input interface.
- phy_clk_share_e: Specifies whether PHY1 and PHY2 share the clock with PHY0.
- raw_data_type_e: Specifies the number of bits of transmitted raw data.
- mipi_wdr_mode_e: Defines the MIPI WDR mode.
- mipi_dev_attr_t: Defines the attributes of the MIPI device.
- wdr_mode_e: Defines the LVDS WDR mode.
- lvds_sync_mode_e: Defines the LVDS sync mode.
- lvds_bit_endian: Defines the bit big/little endian mode.
- lvds_vsync_type_e: Defines the LVDS VSYNC types.
- lvds_vsync_type_t: Defines the LVDS VSYNC parameters.
- lvds_fid_type_e: Defines the frame ID type.
- lvds_fid_type_t: Defines the frame ID configuration information.
- lvds_dev_attr_t: Defines the LVDS/SubLVDS/HiSPi device attributes.

- phy_cmv_e: Defines the PHY common-mode voltage mode.
- phy_cmv_t: Defines the configuration information of the PHY common-mode voltage.
- combo_dev_attr_t: Defines the combo device attributes.
- img_rect_t: Defines the crop attributes.
- img_size_t: Defines the width and height of an image.

HI_MIPI_IOC_MAGIC

[Description]

Defines the magic number of the MIPI RX ioctl command.

[Definition]

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

[Member]

None

[Chip Difference]

None

[Note]

None

[See Also]

None

COMBO_DEV

[Description]

Defines the type of the MIPI RX device.

[Definition]

```
typedef unsigned int COMBO_DEV;
```

[Chip Difference]

None

[Note]

None

[See Also]

- 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

[Description]

Specifies the maximum number of links supported by the device.

[Chip Difference]

Chip	Definition
Hi3516A	#define COMBO_MAX_LINK_NUM 2
Hi3518E V200	#define COMBO_MAX_LINK_NUM 2 (There is only one link. COMBO_MAX_LINK_NUM is defined to 2 to ensure compatibility with the Hi3516A.)
Hi3519 V100	#define COMBO_MAX_LINK_NUM 3
Hi3519 V101	#define COMBO_MAX_LINK_NUM 3
Hi3516C V300	#define COMBO_MAX_LINK_NUM 1

[Note]

None

[See Also]

None

LANE_NUM_PER_LINK

[Description]

Specifies the maximum number of lanes supported by each link.

[Definition]

```
#define LANE_NUM_PER_LINK   4
```

[Chip Difference]

None

[Note]

None

[See Also]

None

COMBO_MAX_LANE_NUM

[Description]

Specifies the maximum number of lanes supported by the device.

[Chip Difference]

Chip	Definition
Hi3516A	#define COMBO_MAX_LANE_NUM 8
Hi3518E V200	#define COMBO_MAX_LANE_NUM 8 (There are only four lanes. COMBO_MAX_LANE_NUM is defined to 8 to ensure compatibility with the Hi3516A.)
Hi3519 V100	#define COMBO_MAX_LANE_NUM 12
Hi3519 V101	#define COMBO_MAX_LANE_NUM 12
Hi3516C V300	#define COMBO_MAX_LANE_NUM 4

[Note]

None

[See Also]

None

MIPI_LANE_NUM

[Description]

Specifies the number of lanes supported by the MIPI interface.

[Chip Difference]

Chip	Definition
Hi3516A	#define MIPI_LANE_NUM COMBO_MAX_LANE_NUM
Hi3518E V200	#define MIPI_LANE_NUM COMBO_MAX_LANE_NUM
Hi3519 V100	#define MIPI_LANE_NUM (LANE_NUM_PER_LINK * 2)
Hi3519 V101	#define MIPI_LANE_NUM (LANE_NUM_PER_LINK * 2)
Hi3516C V300	#define MIPI_LANE_NUM (LANE_NUM_PER_LINK * 1)

[Note]

None

[See Also]

None

LVDS_LANE_NUM

[Description]

Specifies the number of lanes supported by the LVDS/HiSPi interface.

[Definition]

```
#define LVDS_LANE_NUM       COMBO_MAX_LANE_NUM
```

[Chip Difference]

None

[Note]

None

[See Also]

None

COMBO_MAX_DEV_NUM

[Description]

Specifies the number of the MIPI RX devices.

[Chip Difference]

Chip	Definition
Hi3516A	None
Hi3518E V200	None
Hi3519 V100	#define COMBO_MAX_DEV_NUM 1
Hi3519 V101	#define COMBO_MAX_DEV_NUM 2
Hi3516C V300	#define COMBO_MAX_DEV_NUM 1

[Note]

None

[See Also]

None

WDR_VC_NUM

[Description]

Specifies the maximum number of supported virtual channels.

[Definition]

```
#define WDR_VC_NUM          4
```

[Chip Difference]

None

[Note]

None

[See Also]

None

SYNC_CODE_NUM

[Description]

Specifies the number of sync codes supported by each virtual channel.

[Definition]

```
#define SYNC_CODE_NUM        4
```

[Chip Difference]

None

[Note]

None

[See Also]

None

input_mode_t

[Description]

Specifies the type of the MIPI RX input interface.

[Definition]

Hi3516A/Hi3518E V200

```
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 1.8 V */
    INPUT_MODE_CMOS_33V      = 0x5,            /* CMOS 3.3 V */
    INPUT_MODE_BT1120        = 0x6,            /* CMOS 3.3 V */
    INPUT_MODE_BYPASS        = 0x7,            /* MIPI Bypass */
    INPUT_MODE_BUTT
}input_mode_t;
```

Hi3519 V100/Hi3519 V101

```
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_BYPASS      = 0x6,          /* MIPI Bypass */
    INPUT_MODE_BUTT
} input_mode_t;
```

Hi3516C V300

```
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;
```

[Chip Difference]

Chip	input_mode_t
Hi3516A/Hi3518E V200	The CMOS mode is defined as INPUT_MODE_CMOS_18V and INPUT_MODE_CMOS_33V.
Hi3519 V100/Hi3519 V101/Hi3516C V300	The MIPI RX can identify the 1.8 V and 3.3 V CMOS. Therefore, the CMOS mode is defined as INPUT_MODE_CMOS.

[Note]

INPUT_MODE_BYPASS is not supported.

[See Also]

None

phy_clk_share_e

[Description]

Specifies whether PHY1 and PHY2 share the clock with PHY0.

[Definition]

```
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;
```

[Member]

Member	Description
PHY_CLK_SHARE_NONE	PHY1 and PHY2 use separate clocks.
PHY_CLK_SHARE_PHY0	PHY1 and PHY2 share the clock with PHY0.

[Chip Difference]

Chip	Whether to Support Shared PHY Clock
Hi3516A	Not supported
Hi3518E V200	Not supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Not supported

[Note]

None

[See Also]

None

raw_data_type_e

[Description]

Specifies the number of bits of transmitted raw data.

[Definition]

Hi3516A/Hi3518E V200

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

Hi3519 V100/Hi3519 V101/Hi3516C V300

```
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;
```

[Chip Difference]

None

[Note]

None

[See Also]

None

mipi_wdr_mode_e

[Description]

Defines the MIPI WDR mode.

[Definition]

```
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;
```

[Member]

Member	Description
HI_MIPI_WDR_MODE_NONE	Linear mode
HI_MIPI_WDR_MODE_VC	The virtual channel in the packet header is used to differentiate long and short exposure frames.
HI_MIPI_WDR_MODE_DT	The self-defined data type in the packet header is used to differentiate long and short exposure frames.

Member	Description
HI_MIPI_WDR_MODE_DOL	DOL-mode WDR. A pixel after the packet header is used to differentiate long and short exposure frames.

[Chip Difference]

Chip	Whether mipi_wdr_mode_e Is Supported
Hi3516A	Not supported
Hi3518E V200	Not supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Supported

[Note]

None

[See Also]

None

mipi_dev_attr_t

[Description]

Defines the attributes of the MIPI device.

[Definition]

Hi3516A/Hi3518E V200

```
typedef struct
{
    raw_data_type_e         raw_data_type;
    short                 lane_id[MIPI_LANE_NUM];
}mipi_dev_attr_t;
```

Hi3519 V100/Hi3519 V101/Hi3516C V300

```
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;
```

[Member]

Member	Description
raw_data_type	Number of bits of transmitted raw data
lane_id	Mapping between the TX end (sensor) and RX end (MIPI RX) lanes This member is set to **–1** for unused lanes.
wdr_mode	MIPI WDR mode
data_type	Data type corresponding to different exposure length data. When **wdr_mode** is **HI_MIPI_WDR_MODE_DT**, **data_type** needs to be configured.

[Chip Difference]

Chip	Supported raw_data_type
Hi3516A	10-bit/12-bit/14-bit
Hi3518E V200	8-bit/10-bit/12-bit/14-bit
Hi3519 V100/Hi3519 V101/Hi3516C V300	8-bit/10-bit/12-bit/14-bit/16-bit

Chip	wdr_mode
Hi3516A/Hi3518E V200	The configuration of **wdr_mode** is not supported. The virtual channel is used to differentiate data with different exposure lengths by default.
Hi3519 V100/Hi3519 V101/Hi3516C V300	The configuration of **wdr_mode** is supported. Besides the virtual channel mode, the date type WDR and Sony DOL mode WDR are supported.

[Note]

None

[See Also]

- raw_data_type_e
- mipi_wdr_mode_e

- HI_MIPI_SET_DEV_ATTR

wdr_mode_e

[Description]

Defines the LVDS WDR mode.

[Definition]

```
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;
```

[Member]

Member	Description
HI_WDR_MODE_NONE	Linear mode
HI_WDR_MODE_2F	Two-in-one WDR
HI_WDR_MODE_3F	Three-in-one WDR
HI_WDR_MODE_4F	Four-in-one WDR
HI_WDR_MODE_DOL_2F	DOL mode two-in-one WDR
HI_WDR_MODE_DOL_3F	DOL mode three-in-one WDR
HI_WDR_MODE_DOL_4F	DOL mode four-in-one WDR

[Chip Difference]

None

[Note]

- The DOL WDR mode needs to be configured to **HI_WDR_MODE_DOL_2F**, **HI_WDR_MODE_DOL_3F**, or **HI_WDR_MODE_DOL_4F**.
- The built-in WDR mode and frame-merging WDR mode need to be set to **HI_WDR_MODE_NONE**.

[See Also]

None

lvds_sync_mode_e

[Description]

Defines the LVDS sync mode.

Table 1-2 LVDS sync mode

sync_mode	Sync Mode
LVDS_SYNC_MODE_SOL/ LVDS_SYNC_MODE_SOF	SOF, EOF, SOL, EOL See Figure 1-1.
LVDS_SYNC_MODE_SAV	invalid SAV, invalid EAV, valid SAV, valid EAV See Figure 1-2.

[Chip Difference]

Chip	Definition

Hi3516A/Hi3518E V200	`typedef enum`
	`{`
	`    LVDS_SYNC_MODE_SOL = 0,`
	`    LVDS_SYNC_MODE_SAV,`
	`    LVDS_SYNC_MODE_BUTT`
	`} lvds_sync_mode_e;`

Hi3519 V100/Hi3519 V101/Hi3516C V300	`typedef enum`
	`{`
	`    LVDS_SYNC_MODE_SOF = 0,`
	`    LVDS_SYNC_MODE_SAV,`
	`    LVDS_SYNC_MODE_BUTT`
	`} lvds_sync_mode_e;`

[Note]

LVDS_SYNC_MODE_SOF and LVDS_SYNC_MODE_SOL have the same meaning. For Hi3519 V100, LVDS_SYNC_MODE_SOL is changed to LVDS_SYNC_MODE_SOF.

[See Also]

None

lvds_bit_endian

[Description]

Defines the bit big/little endian mode.

[Definition]

```
typedef enum
```

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

[Chip Difference]

None

[Note]

None

[See Also]

None

lvds_vsync_type_e

[Description]

Defines the LVDS VSYNC types.

[Definition]

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

[Member]

Member	Description
LVDS_VSYNC_NORMAL	The long and short exposure frames have independent SOF-EOF and SOL-EOL or invalid SAV-invalid EAV and valid SAV-valid EAV.
LVDS_VSYNC_SHARE	The long and short exposure frames share a pair of SOF-EOF flags, and the first few lines of the short exposure frame are filled with fixed values.
LVDS_VSYNC_HCONNECT	The long and short exposure frames share a pair of SAV-EAV flags, and blanking with a fixed period is between the long and short exposure frames.

LVDS_VSYNC_SHARE sync mode

<table>
<tr><td>SOF</td><td rowspan="2">Long exposure</td><td rowspan="3">EOL</td><td rowspan="3">Horizontal blanking</td><td rowspan="3">SOL</td><td>Padding</td><td rowspan="2">EOL</td><td rowspan="3">Horizontal blanking</td></tr>
<tr><td rowspan="2">SOL</td><td rowspan="2">Short exposure</td></tr>
<tr><td>Padding</td><td>EOF</td></tr>
<tr><td>SOV</td><td>V.BLK</td><td>EOV</td><td>-</td><td>SOV</td><td>V.BLK</td><td>EOV</td><td>-</td></tr>
</table>

LVDS_VSYNC_HCONNECT sync mode

<table>
<tr><td rowspan="6"><b>SAV</b></td><td rowspan="3">Long exposure frame</td><td rowspan="5">Horizontal blanking (fixed period)</td><td>V.BLK</td><td rowspan="5">Horizontal blanking (fixed period)</td><td rowspan="2">V.BLK</td><td rowspan="6"><b>EAV</b></td><td rowspan="5">Horizontal blanking</td></tr>
<tr><td rowspan="3">Short exposure frame 1</td></tr>
<tr><td rowspan="3">Short exposure frame 2</td></tr>
<tr><td rowspan="2">V.BLK</td></tr>
<tr><td>V.BLK</td></tr>
<tr><td colspan="5">V.BLK</td><td></td></tr>
</table>

[Chip Difference]

Chip	Whether lvds_vsync_type_e Is Supported
Hi3516A/Hi3518E V200	Not supported
Hi3519 V100/Hi3519 V101	Supported
Hi3516C V300	Supported

[Note]

None

[See Also]

lvds_vsync_type_t

lvds_vsync_type_t

[Description]

Defines the LVDS VSYNC parameters.

[Definition]

```
typedef struct
{
    lvds_vsync_type_e sync_type;

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

[Chip Difference]

Chip	Whether lvds_vsync_type_t Is Supported
Hi3516A/Hi3518E V200	Not supported
Hi3519 V100/Hi3519 V101	Supported
Hi3516C V300	Supported

[Note]

When **sync_type** is **LVDS_VSYNC_HCONNECT**, **hblank1** and **hblank2** need to be configured, indicating the blanking region length of **Hconnect**.

[See Also]

lvds_vsync_type_e

lvds_fid_type_e

[Description]

Defines the frame ID type.

[Definition]

```
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;
```

- **LVDS_FID_NONE** indicates that the frame ID is not used.
- **LVDS_FID_IN_SAV** indicates that the FID is inserted into the fourth field of SAV. **fid_type** of the sync code of the four fields of the DOL needs to be set to **LVDS_FID_IN_SAV**.

- **LVDS_FID_IN_DATA** indicates that the FID is inserted before the first pixel after the sync code as the frame information column. **fid_type** of the sync code of the five fields of the DOL needs to be set to **LVDS_FID_IN_DATA**.

[Chip Difference]

Chip	Whether lvds_fid_type_e Is Supported
Hi3516A/Hi3518E V200	Not supported
Hi3519 V100/Hi3519 V101	Supported
Hi3516C V300	Supported

[Note]

None

[See Also]

None

lvds_fid_type_t

[Description]

Defines the frame ID configuration information.

[Definition]

```
typedef struct
{
    lvds_vsync_type_e fid;

    HI_BOOL output_fil;
} lvds_fid_type_t;
```

[Member]

Member	Description
fid	Frame ID type in LVDS DOL mode
output_fil	The frame information line in DOL mode is output right after the V-Blanking. The frame ID is the first pixel value in the frame information line. The frame information line does not contain valid video data. If **output_fil** is set to **HI_TRUE**, the frame information line is output to the back-end device. If **output_fil** is set to **HI_FALSE**, the MIPI RX will discard data in this line.

[Chip Difference]

Chip	Whether lvds_fid_type_t Is Supported
Hi3516A/Hi3518E V200	Not supported
Hi3519 V100/Hi3519 V101	Supported
Hi3516C V300	Supported

[Note]

None

[See Also]

lvds_fid_type_e

lvds_dev_attr_t

[Description]

Defines the LVDS/SubLVDS/HiSPi device attributes.

[Definition]

Hi3516A/Hi3518E V200

```
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;
```

Hi3519 V100/Hi3516C V300

```
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;
```

Hi3519 V101

```
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;
```

[Member]

Member	Description
img_size	Width and height of the sensor input image. The image width must be an integral multiple of the number of valid lanes.
wdr_mode	WDR mode
sync_mode	LVDS sync mode
raw_data_type	Number of bits of transmitted raw data
data_endian	Data endian mode
sync_code_endian	Sync code endian mode
lane_id	Mapping between the TX end (sensor) and RX end (MIPI RX) lanes This member is set to **–1** for unused lanes. For details about how to configure the lane ID, see section 1.7.1 "How Do I Configure the Lane ID?"

Member	Description
sync_code	Each virtual channel has four sync codes, indicating the SOF/EOF/SOL/EOL sync code or invalid SAV/invalid EAV/valid SAV/valid EAV sync code respectively according to the sync mode.
vsync_type	VSYNC type. It needs to be configured when **wdr_mod** is DOL mode and **sync_mode** is **LVDS_SYNC_MODE_SAV**.
fid_type	Frame ID type. It needs to be configured when **wdr_mod** is DOL mode and **sync_mode** is **LVDS_SYNC_MODE_SAV**.

[Chip Difference]

Chip	raw_data_type
Hi3516A	10-bit/12-bit/14-bit
Hi3518E V200	8-bit/10-bit/12-bit/14-bit
Hi3519 V100	8-bit/10-bit/12-bit/14-bit/16-bit
Hi3519 V101	8-bit/10-bit/12-bit/14-bit/16-bit
Hi3516C V300	8-bit/10-bit/12-bit/14-bit/16-bit

Chip	vsync_type	fid_type
Hi3516A	Not supported	Not supported
Hi3518E V200	Not supported	Not supported
Hi3519 V100	Supported	Supported
Hi3519 V101	Supported	Supported
Hi3516C V300	Supported	Supported

Chip	img_size
Hi3516A	Supported
Hi3518E V200	Supported
Hi3519 V100	Supported
Hi3519 V101	**img_size** is moved to **img_rect** in combo_dev_attr_t.
Hi3516C V300	Supported

[Note]

None

[See Also]

- 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_cmv_e

[Description]

Defines the PHY common-mode voltage mode.

[Definition]

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

[Member]

Member	Description
PHY_CMV_GE900MV	The PHY common-mode voltage is greater than or equal to 900 mV.
PHY_CMV_LT900MV	The PHY common-mode voltage is less than 900 mV.

[Chip Difference]

Chip	Supported or Not
Hi3516A	Not supported
Hi3518E V200	Not supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Supported

[Note]

None

[See Also]

None

phy_cmv_t

[Description]

Defines the configuration information of the PHY common-mode voltage.

[Definition]

```
typedef struct

{

    COMBO_DEV   devno;
    phy_cmv_e   cmv_mode;

} phy_cmv_t;
```

[Member]

Member	Description
devno	ID of the MIPI RX device
cmv_mode	Voltage mode of the PHY function

[Chip Difference]

Chip	Supported or Not
Hi3516A	Not supported
Hi3518EV200	Not supported
Hi3519V100	Supported
Hi3519V101	Supported
Hi3516CV300	Supported

[Note]

None

[See Also]

- , phy_cmv_e
- HI_MIPI_SET_PHY_CMVMODE

combo_dev_attr_t

[Description]

Defines the combo device attributes. The MIPI RX device is called the combo device because the MIPI RX can interwork with the CSI-2, LVDS, and HiSPi timings.

[Definition]

Hi3516A/Hi3518E V200

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

Hi3519 V100

```
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;
```

Hi3519 V101

```
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;
```

Hi3516C V300

```
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;
```

[Member]

Member	Description
input_mode	Input interface type
mipi_attr	**mipi_attr** must be configured if **input_mode** is set to **INPUT_MODE_MIPI**.
lvds_attr	**lvds_attr** must be configured if **input_mode** is set to **INPUT_MODE_SUBLVDS**, **INPUT_MODE_LVDS**, or **INPUT_MODE_HISPI**.
devno	MIPI RX device ID
phy_clk_share	Information about the shared PHY clock
img_rect	Picture crop region

[Chip Difference]

Chip	devno	phy_clk_share
Hi3516A	Not supported	Not supported
Hi3518E V200	Not supported	Not supported
Hi3519 V100	Supported	Supported. PHY1 and PHY2 can share the PHY clock.
Hi3519 V101	Supported	Supported. PHY1 and PHY2 can share the PHY clock.

Chip	devno	phy_clk_share
Hi3516C V300	Supported	Not supported

Chip	img_rect
Hi3516A	Not supported
Hi3518E V200	Not supported
Hi3519 V100	Not supported. But the crop region can be configured by calling HI_MIPI_SET_CROP.
Hi3519 V101	Supported
Hi3516C V300	Not supported

[Note]

None

[See Also]

None

img_rect_t

[Description]

Defines the MIPI crop attributes.

[Definition]

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

[Member]

Member	Description
x	X coordinate of the crop start position
y	Y coordinate of the crop start position
width	Crop width
height	Crop height

[Chip Difference]

Chip	Crop Interface
Hi3516A	Not supported
Hi3518E V200	Not supported
Hi3519 V100	Supported
Hi3519 V101	Supported
Hi3516C V300	Not supported

[Note]

None

[See Also]

HI_MIPI_SET_CROP

img_size_t

[Description]

Defines the width and height of an image.

[Definition]

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

[Member]

Member	Description
width	Image width
height	Image height

[Chip Difference]

None

[Note]

None

[See Also]

HI_MIPI_SET_DEV_ATTR

1.6 Proc Information

[Debugging Information]

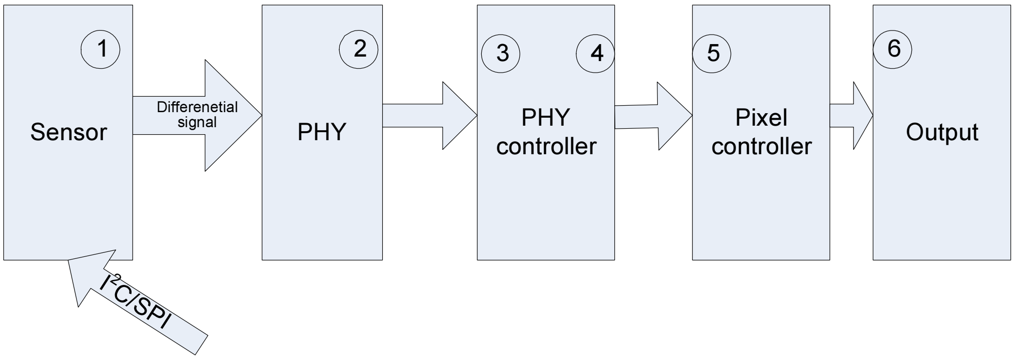
```
Module: [MIPI], Build Time: [May 24 2016, 22:40:41]

-----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--------------------------------------------------
 LinkIdx LaneCount       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        0
   1     0         0         0         0         0        0        0
0        0        0
   2     0         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
```

[Analysis]

- The MIPI RX receives differential data of the sensor by using the PHY. After detecting the sync header, the PHY controller aligns data in each lane.
- The pixel controller parses sync information and merges data in the lane into pixel data based on the bit width of raw data. It transmits pixel data to the downstream module in output mode.
- The clocks of the PHY, PHY controller, and pixel controller are provided by the pixel clock of the sensor. The clock of the output module is the associated clock, which is the same as the working clock of the downstream module. The crop function of the MIPI RX is implemented at the end of the pixel controller. Therefore, the required associated clock can be reduced after cropping.

Figure 1-3 MIPI data stream



[Parameter Description]

Parameter		Description
Combo DEV	Devno	MIPI device ID

Parameter		Description
ATTR	WorkMode	• MIPI device working mode LVDS/MIPI/CMOS mode
	DataType	• Data type RAW8/RAW10/RAW12/RAW14/RAW16 -bit
	WDRMode	WDR mode • None: non-WDR mode • 2To1: two-in-one WDR • 3To1: three-in-one WDR • 4To1: four-in-one WDR • DOL2To1: DOL two-in-one WDR • DOL3To1: DOL three-in-one WDR • DOL4To1: DOL four-in-one WDR
	LinkId	IDs of links used by the device A physical link corresponds to four lanes.
	bEnCrop	Whether the crop function is enabled. **N** indicates disabled, and **Y** indicates enabled.
	ImgX	X coordinate of the cropped image
	ImgY	Y coordinate of the cropped image
	ImgW	Width of the cropped image
	ImgH	Height of the cropped image
	SyncMode	Sync header mode SOF: The sync mode is SOF, EOF, SOL, or EOL. SAV: The sync mode is invalid SAV, invalid EAV, valid SAV, or valid EAV.
	DataEndian	Data endian mode • Big: big endian mode • Little: little endian mode
	SyncCodeEndian	Sync header endian mode • Big: big endian mode • Little: little endian mode
LINK INFO	LinkIdx	Link ID
	LaneCount	Number of lanes in the link
	LaneId	Lane ID
	PhyData	Real-time data received by the PHY

Parameter		Description
	AlignedData	Real-time data after the frame sync signal is detected
	ValidLane	Valid lane ID in the link
mipi detect info (visible only in MIPI mode)	Devno	MIPI RX device ID
	VC	Virtual channel
	width	Total width of images detected by the MIPI controller
	height	Total height of images detected by the MIPI controller
lvds detect info (visible only in LVDS/SubLVDS/ HiSPi mode)	Devno	MIPI RX device ID
	Lane	Lane ID
	LaneWidth	Image width detected by the lane
	Devno	MIPI RX device ID
	WDR_Frame	Long/Short frame in WDR mode. This column is not displayed in linear mode. • LEF: long exposure frame • SEF1: short exposure frame 1 • SEF2: short exposure frame 2 • SEF3: short exposure frame 3
	width	Total width of images detected by the LVDS controller
	height	Total height of images detected by the LVDS controller
fsm timeout and escape info (visible only in MIPI mode)	link	Link ID
	clkTOutCnt	Timeout when the clock lane is switched from the LP to HS
	d0TOutCnt	Timeout when data lane 0 is switched from the LP to HS
	d1TOutCnt	Timeout when data lane 1 is switched from the LP to HS
	d2TOutCnt	Timeout when data lane 2 is switched from the LP to HS
	d3TOutCnt	Timeout when data lane 3 is switched from the LP to HS
	clkEscCnt	Timeout when the clock lane is switched to escape mode

Parameter		Description
	d0EscCnt	Timeout when data lane 0 is switched to escape mode
	d1EscCnt	Timeout when data lane 1 is switched to escape mode
	d2EscCnt	Timeout when data lane 2 is switched to escape mode
	d3EscCnt	Timeout when data lane 3 is switched to escape mode
ALING Err info	Devno	MIPI device ID
	FIFO_FullErr	FIFO overflow
	Lane0Err	Lane 0 FIFO overflow
	Lane1Err	Lane 1 FIFO overflow
	Lane2Err	Lane 2 FIFO overflow
	Lane3Err	Lane 3 FIFO overflow
	Lane4Err	Lane 4 FIFO overflow
	Lane5Err	Lane 5 FIFO overflow
	Lane6Err	Lane 6 FIFO overflow
	Lane7Err	Lane 7 FIFO overflow
	Lane8Err	Lane 8 FIFO overflow
	Lane9Err	Lane 9 FIFO overflow
	Lane10Err	Lane 10 FIFO overflow
	Lane11Err	Lane 11 FIFO overflow

1.7 FAQs

For details about the Hi3516A MIPI specifications, see the *Hi3516A/Hi3516D HD IP Camera SoC Data Sheet* and *Features of the Video Interfaces of HiSilicon IP Cameras*.

1.7.1 How Do I Configure the Lane ID?

The lane ID corresponds to short lane_id[MIPI_LANE_NUM] in mipi_dev_attr_t or short lane_id[LVDS_LANE_NUM] in lvds_dev_attr_t.

Set **lane_id** of unused lanes to **–1** when the MIPI connects to the sensor. You can also adjust the data channel sequence by configuring **lane-id** based on the hardware board and actual sensor output channels.

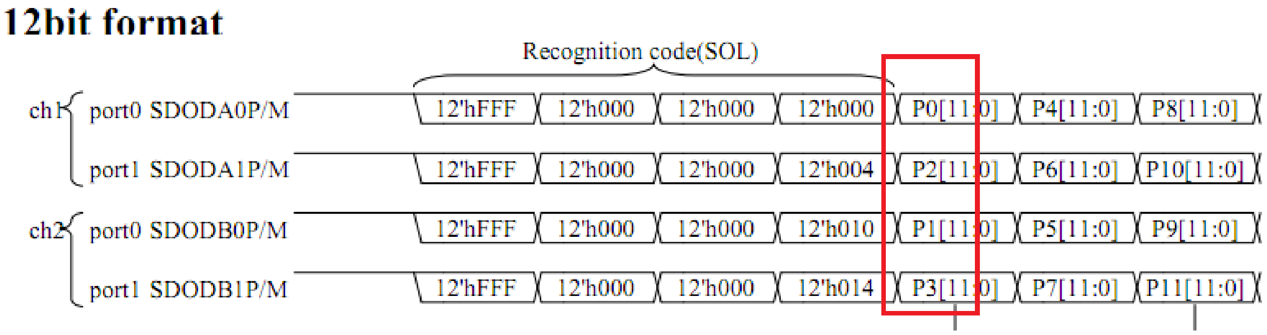
The following uses the MN34220 of the demo board as an example. Table 1-3 describes the mapping between the MIPI_Rx pins of the demo board and pins of the MN34220.

Table 1-3 Mapping between the MN34220 and MIPI_Rx pins

MN34220 Pins	Hi3516A MIPI Pins
SENSOR_SDODA0M SENSOR_SDODA0P	MIPI0_D0M MIPI0_D0P
SENSOR_SDODA1M SENSOR_SDODA1P	MIPI0_D1M MIPI0_D1P
SENSOR_SDODB0M SENSOR_SDODB0P	MIPI1_D0M MIPI1_D0P
SENSOR_SDODB1M SENSOR_SDODB1P	MIPI1_D1M MIPI1_D1P

The actual transferred data sequence of pins is pixel 0, pixel 2, pixel 1, and pixel 3. For details, see Figure 1-4.

Figure 1-4 MN34220 output timings (2-channel 2-port 12-bit format)



Because the sensor does not output data to MIPI0_D2M, MIPI0_D2P, MIPI0_D3M, MIPI0_D3P, MIPI1_D2M, MIPI1_D2P, MIPI1_D3M, and MIPI1_D3P of MIPI RX, the corresponding **lane_id** needs to be set to **–1**. **lane_id** is configured as follows:

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

sync_code configured takes effect based on **lane_id**. If **lane_id** is **–1**, the corresponding **sync_code** does not take effect.

1.7.2 How Do I Configure the LVDS Mode Sync Code?

There are two LVDS/SUB_LVDS sync modes. For details, see lvds_sync_mode_e.

- LVDS_SYNC_MODE_SOF/LVDS_SYNC_MODE_SOL
- LVDS_SYNC_MODE_SAV

The sync code needs to be configured for different transmission modes of the same sensor or different sensors.

The sync code is defined as follows:

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

The sync code needs to be configured based on the sensor data sheet. See Table 1-4.

Table 1-4 sync_code definition

sync_code Element	Definition
LVDS_LANE_NUM	It corresponds to the LVDS hardware physical channel.
WDR_VC_NUM	It indicates the number of WDR channels. For example, the two-in-one WDR corresponds to two WDR channels. There are at most four WDR channels.
SYNC_CODE_NUM	The sync code of each lane consists of four codewords, which have different meanings in different sync mode. For details, see Table 1-2.

NOTE

The first three codewords of the sync_code for each SOF/EOF/SOL/EOL are fixed at 0xFFFF, 0x0000, 0x0000.

Hi3516A/Hi3518E V200 Sync Code Configuration Sample

The principles for configuring the sync code are as follows:

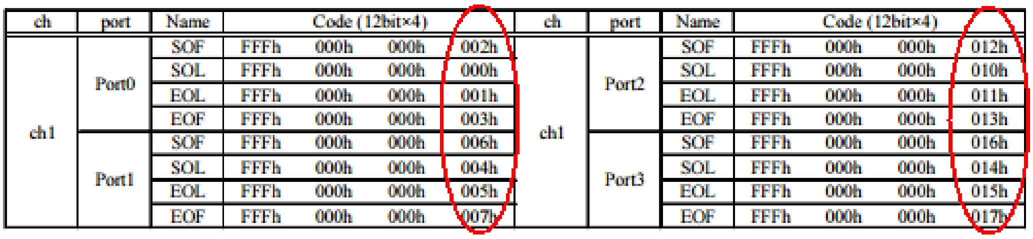
- For disorder within one link, the sync code is configured in normal sequence regardless of the sequence for the hardware to connect to the sensor.
- For disorder between two links, the preceding principle also applies. That is, the sync code is configured in the normal sequence in one link. Disorder between two links commonly exists only for Panasonic sensors.

The following are some examples.

- Disorder within one link

  Figure 1-5 shows the description about the sync code in the data sheet when the MN34220 works in 1-channel 4-port (4-lane) 12-bit mode.

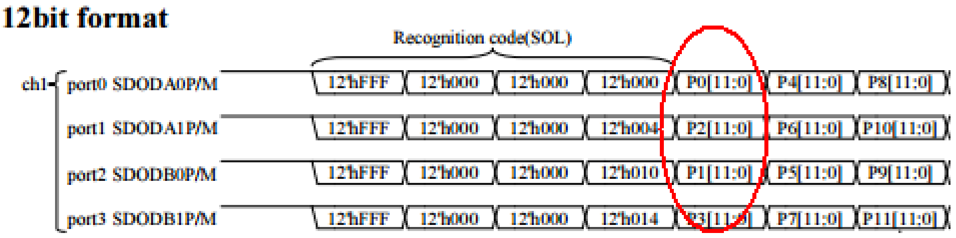
Figure 1-5 Sync code



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

Figure 1-6 shows the pixel format of each channel.

Figure 1-6 Pixel format



Therefore, the normal sequence should be SDODA0, SDODB0, SDODA1, and SDODB1. Only the sync code configurations of the first four channels take effect. You can configure the sync code in the MIPI interface based on this sequence. The sync_code configuration is as follows:

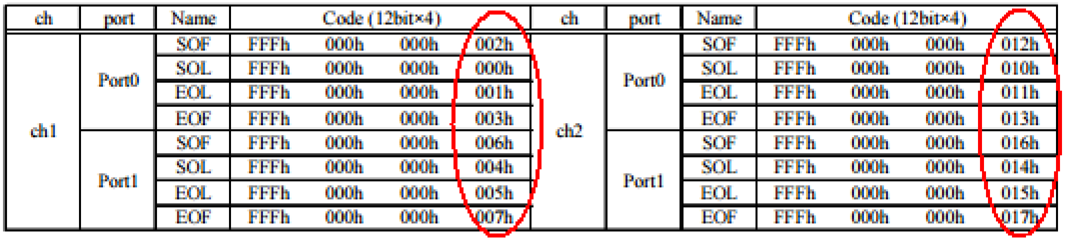
```
.sync_code =  {
               {{0x002, 0x003, 0x000, 0x001}, //PHY0_lane0
               {0x202, 0x203, 0x200, 0x201},
               {0x102, 0x103, 0x100, 0x101},
               {0x302, 0x303, 0x300, 0x301}},

               {{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}}
             }
```

- Disorder between two links

  Figure 1-7 shows the description about the sync code in the data sheet when the MN34220 works in 2-channel 2-port (4-lane) 12-bit mode.

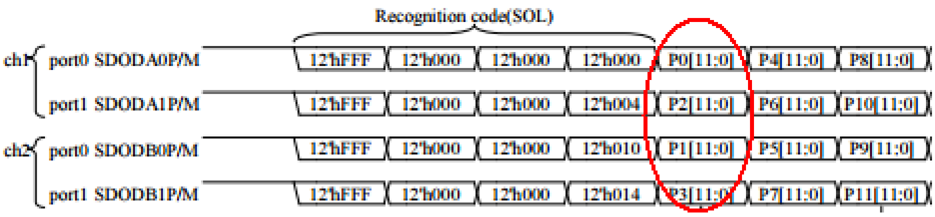
Figure 1-7 Sync code



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

Figure 1-8 shows the pixel format of each channel.

Figure 1-8 Pixel format



The line sequences of the two links may overlap. However, according to the preceding principles, the sync code configuration is considered only within the same link.

The configurations of sync code of channel 0 and channel 1 of link 0 as well as channel 0 and channel 1 of link 1 are valid. Therefore, the final sync code is configured as follows:

```
.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}}
   }
```

Hi3519 V100 /Hi3519 V101/Hi3516C V300 Sync Code Configuration Sample

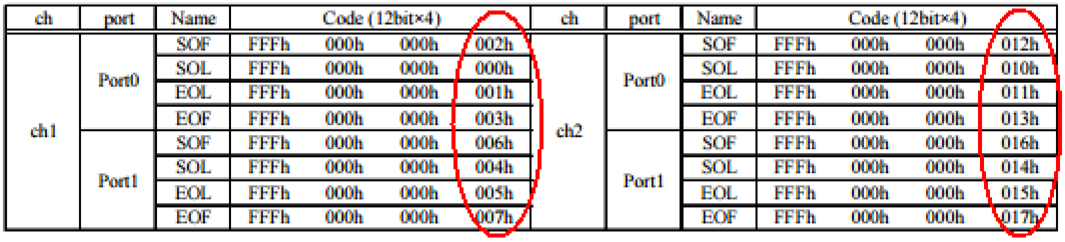
- The principles for configuring the sync code are as follows:

  The sync code of the first few lanes in sync_code is configured based on the number of used lanes. If the lanes are disordered, the sync code is configured based on the normal sequence.

- The following is an example:

  Figure 1-9 shows the description about the sync code in the data sheet when the MN34220 works in 2-channel 2-port (4-lane) 12-bit mode.

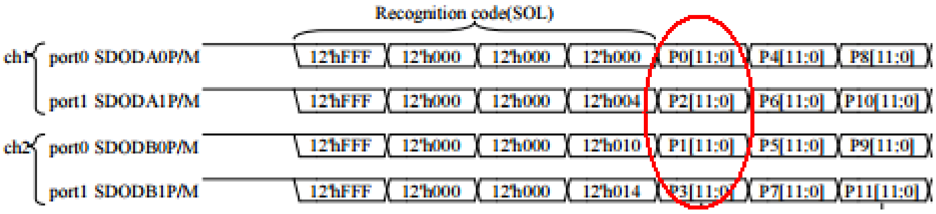
Figure 1-9 Sync code



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

Figure 1-10 shows the pixel format of each channel.

Figure 1-10 Pixel format



The sync code is configured as follows:

```
.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 What Is the Relationship Between the MIPI Lane Frequency and VI Frequency?

When multiple MIPI lanes are used for data transfer, what is the relationship between the transfer frequency of MIPI lanes and VI processing frequency, and how do I calculate the maximum transfer frequency of each lane?

- The MIPI RX receives data from multiple lanes, converts data into the internal timing, and transmits the timing to the VIU for processing. The total amount of data transferred by multiple lanes remains unchanged, as described in the following equation:

  VI_Freq x Pix_Width = Lane_Num x MIPI_Freq

- Where **VI_Freq** indicates the frequency of the VI working clock, **Pix_Width** indicates the pixel bit width, **Lane_Num** indicates the number of lanes used for data transfer, and **MIPI_Freq** indicates the maximum RX frequency of each lane.

- **MIPI_Freq** is calculated as follows: MIPI_Freq = (VI_Freq x Pix_Width)/Lane_Num. For example, if the frequency of the VI working clock is 250 MHz, the MIPI data is in RAW12 format, and four lanes are used for data transfer, **MIPI_Freq** is calculated as follows:

  MIPI_Freq = (250 x 12) / 4 = 750

  That is, the maximum transfer frequency of each lane is 750 MHz.

1.7.4 How Do I Configure the Sensor Reset Function?

- The sensor reset and unreset operations are implemented in the MIPI driver. The reset sequence in the MIPI driver is as follows:
  - sensor reset
  - mipi core reset
  - config mipi attr
  - mipi core unreset
  - sensor unreset
- The chip has a pin (SENSOR_RSTN) dedicated for sensor reset. You are advised to use it by default.
- If the sensor reset pin needs to be changed, the MIPI driver needs to be adapted. You can modify the sensor reset function and sensor reset canceling function (mipi_reset_sensor/mipi_unreset_sensor or mipi_drv_reset_sensor/mipi_drv_unreset_sensor) in the MIPI RX driver based on the used pin.
- If the sensor supports the standby mode and you do not want to reset the sensor, you can enable the sensor standby mode instead of resetting the sensor, and disable the sensor standby mode instead of canceling reset on the sensor.