



CV186AH MIPI 使用手册

Version: 1.0.0

Release date: 2023/12

©2022 北京晶视智能科技有限公司
本文件所含信息归北京晶视智能科技有限公司所有。
未经授权，严禁全部或部分复制或披露该等信息。

目录

1	声明	2
2	CV186AH VI Overview	3
2.1	CV186AH VI 概述	3
2.1.1	多 Sensor 组合情况	4
2.2	SDK 介绍	6
2.2.1	sensor_cfg.ini 文件	6
2.2.2	stWDRAttr 的 WDR 模式	8
3	MIPI 使用指南	9
3.1	重要概念	9
3.2	API 参考	11
3.2.1	CVI_MIPI_SET_DEV_ATTR	12
3.2.2	CVI_MIPI_SET_OUTPUT_CLK_EDGE	14
3.2.3	CVI_MIPI_RESET_SENSOR	14
3.2.4	CVI_MIPI_UNRESET_SENSOR	15
3.2.5	CVI_MIPI_RESET_MIPI	16
3.2.6	CVI_MIPI_ENABLE_SENSOR_CLOCK	17
3.2.7	CVI_MIPI_DISABLE_SENSOR_CLOCK	17
3.2.8	CVI_MIPI_SET_CROP_TOP	18
3.2.9	CVI_MIPI_SET_WDR_MANUAL	19
3.2.10	CVI_MIPI_SET_LVDS_FP_VS	20
3.2.11	CVI_MIPI_SET_BT_FMT_OUT	20
3.2.12	CVI_MIPI_GET_CIF_ATTR	21
3.2.13	CVI_MIPI_SET_SENSOR_CLOCK	22
3.2.14	CVI_MIPI_SET_MAX_MAC_CLOCK	23
3.2.15	CVI_MIPI_SET_CROP_WINDOW	23
3.2.16	CVI_MIPI_SET_YUV_SWAP	24
3.2.17	CVI_MIPI_RESET_LVDS	25
3.2.18	mipi_tx_cfg	25
3.2.19	mipi_tx_send_cmd	26
3.2.20	mipi_tx_recv_cmd	27
3.2.21	mipi_tx_enable	28
3.2.22	mipi_tx_disable	29
3.2.23	mipi_tx_set_hs_settle	29
3.2.24	mipi_tx_get_hs_settle	30
3.2.25	mipi_tx_suspend	31
3.2.26	mipi_tx_resume	32
3.3	数据类型	32
3.3.1	combo_dev_attr_s	34
3.3.2	input_mode_e	36

3.3.3	rx_mac_clk_e	37
3.3.4	mclk_pll_s	38
3.3.5	cam_pll_freq_e	38
3.3.6	mipi_dev_attr_s	39
3.3.7	raw_data_type_e	40
3.3.8	mipi_wdr_mode_e	41
3.3.9	dphy_s	42
3.3.10	lvds_fid_type_s	43
3.3.11	lvds_fid_type_e	43
3.3.12	mipi_demux_info_s	44
3.3.13	lvds_dev_attr_s	44
3.3.14	lvds_sync_mode_e	46
3.3.15	lvds_bit_endian	46
3.3.16	lvds_vsync_type_s	47
3.3.17	lvds_vsync_type_e	48
3.3.18	ttl_dev_attr_s	48
3.3.19	ttl_src_e	49
3.3.20	ttl_fmt_e	50
3.3.21	ttl_VIOclk_e	51
3.3.22	bt_demux_attr_s	52
3.3.23	bt_demux_mode_e	53
3.3.24	bt_demux_sync_s	54
3.3.25	img_size_s	54
3.3.26	manual_wdr_attr_s	55
3.3.27	clk_edge_s	56
3.3.28	clk_edge_e	56
3.3.29	sns_rst_config	57
3.3.30	sns_rst_active_e	58
3.3.31	crop_top_s	58
3.3.32	manual_wdr_s	59
3.3.33	vsync_gen_s	60
3.3.34	bt_fmt_out_s	60
3.3.35	bt_fmt_out_e	61
3.3.36	cif_attr_s	61
3.3.37	cif_crop_win_s	62
3.3.38	cif_yuv_swap_s	63
3.3.39	SNS_ATTR_S	64
3.3.40	SNS_ATTR_LARGE_S	64
3.3.41	ISP_SNS_STATE_S	65
3.3.42	ISP_SNS_MIRRORFLIP_TYPE_E	66
3.3.43	ISP_SNS_INTTIME_MODE_E	67
3.3.44	MCLK_ATTR_S	67
3.3.45	RX_INIT_ATTR_S	68
3.3.46	SNS_BDG_MUX_MODE_E	69
3.3.47	ISP_INIT_ATTR_S	69
3.3.48	ISP_SNS_OBJ_S	71
3.3.49	MIPI_TX0_NAME	73
3.3.50	MIPI_TX1_NAME	73
3.3.51	CMD_MAX_NUM	73
3.3.52	RX_MAX_NUM	74

3.3.53	LANE_MAX_NUM	74
3.3.54	output_mode_e	74
3.3.55	video_mode_e	75
3.3.56	output_format_e	76
3.3.57	mipi_tx_lane_id	76
3.3.58	sync_info_s	77
3.3.59	combo_dev_cfg_s	78
3.3.60	cmd_info_s	79
3.3.61	get_cmd_info_s	80
3.3.62	hs_settle_s	80
3.3.63	dsc_instr	81
3.4	用户函数	81
3.4.1	SAMPLE_COMM_VI_ParseIni	82
3.4.2	SAMPLE_COMM_VI_IniToVICfg	83
3.4.3	SAMPLE_COMM_VI_GetSensorInfo	84
3.4.4	SAMPLE_COMM_VI_GetSizeBySensor	84
3.4.5	SAMPLE_COMM_VI_StartMIPI	85
3.4.6	SAMPLE_COMM_VI_SensorProbe	86
3.4.7	SAMPLE_COMM_VI_ResetSensor	86
3.4.8	SAMPLE_COMM_VI_ResetMipi	87
3.4.9	SAMPLE_COMM_VI_UnresetSensor	88
3.4.10	SAMPLE_COMM_VI_UnresetMipi	88
3.4.11	SAMPLE_COMM_VI_SetMipiAttr	89
3.4.12	SAMPLE_COMM_VI_EnableSensorClock	90
3.4.13	SAMPLE_COMM_VI_SuspendSensor	90
3.4.14	SAMPLE_COMM_VI_ResumeSensor	91
3.4.15	SAMPLE_COMM_VI_SuspendMipi	92
3.4.16	SAMPLE_COMM_VI_ResumeMipi	92
3.4.17	SAMPLE_COMM_VI_StartSensor	93
3.4.18	SAMPLE_COMM_SYS_GetPicSize	94
3.4.19	SAMPLE_PLAT_SYS_INIT	94
3.4.20	SAMPLE_PLAT_VI_INIT	95
3.4.21	SAMPLE_COMM_ISP_SetSnsObj	96
3.4.22	SAMPLE_COMM_ISP_GetSnsObj	97
3.4.23	SAMPLE_COMM_ISP_PatchSnsObj	97
3.4.24	SAMPLE_COMM_ISP_Sensor_Register_callback	98
3.4.25	SAMPLE_COMM_ISP_SetSensorMode	99
3.5	MIPI 函数:	100
3.5.1	mipi_open_dev	100
3.5.2	CVI_MIPI_SetMipiReset	101
3.5.3	CVI_MIPI_SetSensorClock	101
3.5.4	CVI_MIPI_SetSensorReset	102
3.5.5	CVI_MIPI_SetMipiAttr	103
3.5.6	CVI_MIPI_SetClkEdge	104
3.5.7	CVI_MIPI_SetSnsMclk	104
3.6	Proc 信息	105
3.6.1	MIPI_RX Proc 信息	105
3.7	FAQ	108
3.7.1	Land id 如何配置	108
3.7.2	MIPI 频率说明	109

3.7.3	Manual WDR 模式使用说明	110
-------	-----------------------------	-----

修订记录

Revision	Date	Description
1.0.0	2023/03/07	初稿
1.0.1	2023/03/14	部分章节添加一些使用参考
1.0.2	2023/03/16	添加表头以及实际例子采用灰色字体表示
1.0.3	2023/03/27	添加表头以及实际例子采用灰色字体表示

1 声明



法律声明

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

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

联系我们

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

深圳市宝安区福海街道展城社区会展湾云岸广场 T10 栋

电话 +86-10-57590723 +86-10-57590724

邮编 100094（北京）518100（深圳）

官方网站 <https://www.sophgo.com/>

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

2 CV186AH VI Overview

2.1 CV186AH VI 概述

CV186AH VI (VIdeo Input) 是处理摄像头数据接收模块, 支持通过 MIPI 接口 (包含 MIPI、Sub-LVDS) 或 BT.656、BT.601、BT.1120 接口和 DC (Digital Camera) 接收视频数据, 并经过一系列的处理后, 将数据送入下一级模块 (ISP) 中. 下面给出了 VI 模块的结构框图。

VI 分为两个物理子模块, 由 MIPI 信号接收模块 Sensor PHY 和视频输入处理模块 Sensor MAC 组成。Sensor PHY 模块接收处理不同的视频数据, 而 Sensor MAC 模块则会将不同格式的视频信号处理为 ISP 模块所需的单一规范视频信号送出。其中后面 2 组 MAC 只支持 BT.656 和 BT.DEMUX 协议

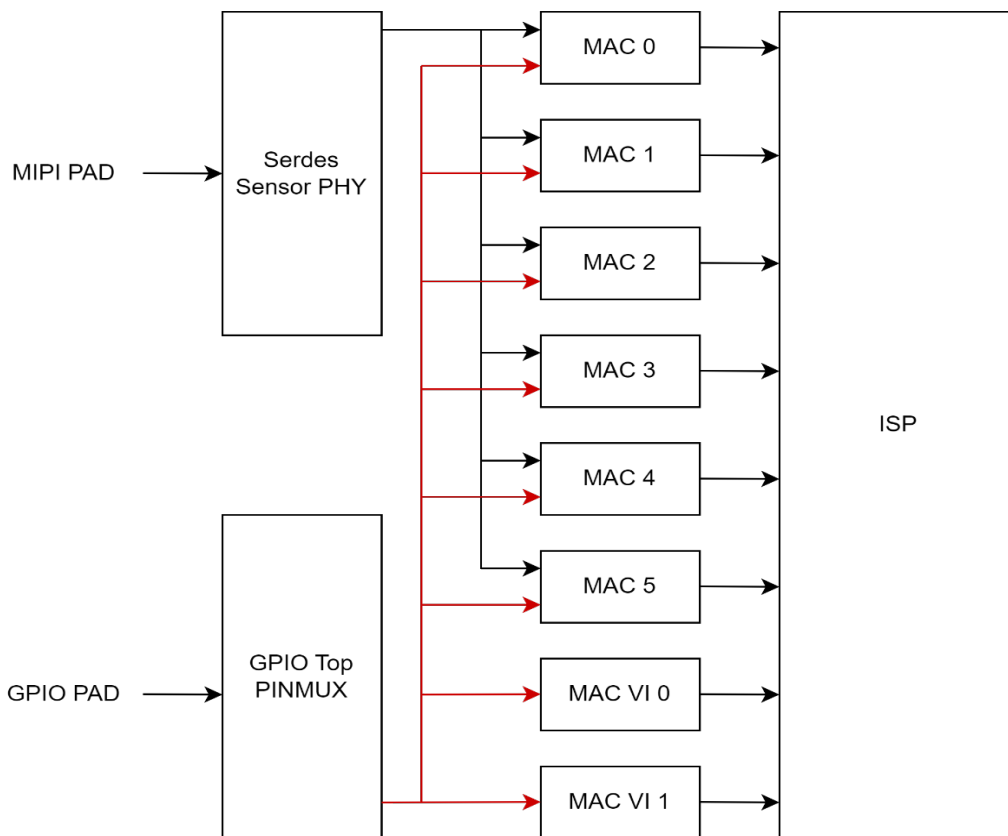


图 2.1: CIF 与 ISP 对接概览图

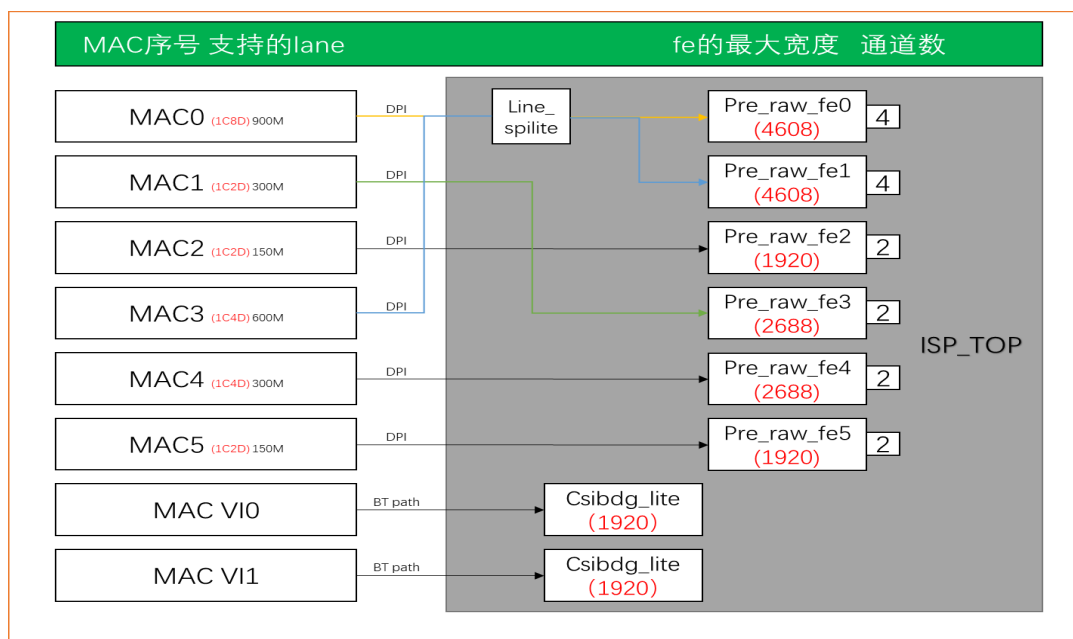


图 2.2: CIF 与 ISP 对接详细图

上图体现了 CIF 与 ISP 中多个 FE 模块通过 sensor bridge 连接，这将对数据进行后续处理，包括了 YUV 和 RAW 图像。而 Csibdg Lite 只支持 yuv，不经过 FE 处理，传递到 BE 阶段或是直接展示。

2.1.1 多 Sensor 组合情况

同时支持多个 MIPI sensor 时，需按照以下表格组合，表中 1C8D, 1C4D, 1C2D, 等均为最大可支持，向下兼容：

MODE_SEL	MODE0	MODE1	MODE2	MODE3	MODE4	MODE5	MODE6
MAC_SEL							
MAC0	1C8D	1C8D	1C8D	1C4D	1C4D	1C4D	1C2D
MAC1	X	X	X	X	X	X	1C2D
MAC2	X	X	X	X	X	1C2D	1C2D
MAC3	X	1C4D	1C2D	1C4D	1C4D	1C2D	1C2D
MAC4	X	X	1C2D	1C4D	1C2D	1C2D	1C2D
MAC5	X	1C2D	1C2D	X	1C2D	1C2D	1C2D

当 MIPI sensor 与 BT/DC sensor 组合，需要遵循以下图示要求，两种 sensor 的接线不可重叠：

PAD_MIPI_RX0P	8L	8L	4L	4L	4L	4L	2L	8L	2L	4L	2L	2L	
PAD_MIPI_RX0N	-	-	-	-	-	-	-	-	-	-	-	-	
PAD_MIPI_RX1P	-	-	-	-	-	-	-	-	-	-	-	-	
PAD_MIPI_RX1N	-	-	-	-	-	-	-	-	-	-	-	-	
PAD_MIPI_RX2P	-	-	-	-	-	-	-	-	-	-	-	-	
PAD_MIPI_RX2N	-	-	-	-	-	-	-	-	-	-	-	-	
PAD_MIPI_RX3P	-	-	-	-	-	-	2L	-	2L	-	2L	2L	
PAD_MIPI_RX3N	-	-	-	-	-	-	-	-	-	-	-	-	
PAD_MIPI_RX4P	-	-	-	-	-	-	-	-	-	-	-	-	
PAD_MIPI_RX4N	-	-	-	-	-	-	-	-	-	-	-	-	
PAD_MIPI_RX5P	-	-	-	-	-	-	-	-	-	-	-	-	
PAD_MIPI_RX5N	-	-	-	-	-	-	-	-	-	-	-	-	
PAD_MIPI_RX6P	-	-	4L	4L	4L	2L	2L	-	2L	4L	-	-	VI2_CLK
PAD_MIPI_RX6N	-	-	-	-	-	-	-	-	-	-	-	-	VI2_HS
PAD_MIPI_RX7P	-	-	-	-	-	-	-	-	-	-	-	-	VI2_VS
PAD_MIPI_RX7N	-	-	-	-	-	-	-	-	-	-	-	-	VI2_DE
PAD_MIPI_RX8P	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D15
PAD_MIPI_RX8N	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D14
PAD_MIPI_RX9P	4L	2L	-	-	-	2L	2L	-	2L	-	-	-	VI0_D13
PAD_MIPI_RX9N	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D12
PAD_MIPI_RX10P	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D11
PAD_MIPI_RX10N	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D10
PAD_MIPI_RX11P	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D9
PAD_MIPI_RX11N	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D8
PAD_MIPI_RX12P	-	-	-	4L	2L	2L	2L	-	-	-	-	-	VI0_D7
PAD_MIPI_RX12N	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D6
PAD_MIPI_RX13P	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D5
PAD_MIPI_RX13N	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D4
PAD_MIPI_RX14P	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D3
PAD_MIPI_RX14N	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D2
PAD_MIPI_RX15P	-	-	-	-	2L	2L	2L	-	-	-	-	-	VI0_D1
PAD_MIPI_RX15N	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D0
PAD_MIPI_RX16P	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D0
PAD_MIPI_RX16N	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D0
PAD_MIPI_RX17P	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D0
PAD_MIPI_RX17N	-	-	-	-	-	-	-	-	-	-	-	-	VI0_D0

图 2.3: sensor 组合图

各个 sensor 的 size 及 use case 见下图，注意，MACVI 作为 MAC6，MAC7 供 btdemux 或者 bt656 使用：

6VI use Case	resolution	MAC0	MAC3	MAC4	MAC1	MAC2	MAC5	MAC VI	before fusion BW(MPix/s)	FE0	FE1	FE4	FE3	FE2	FE5	bdg_lite
1 sensor(8L)-max 8Kp15		1C8D	1C4D	1C4D	1C2D	1C2D	1C2D	AHD/TTL	600	4608	4608	2688	2688	1920	1920	1920
8K@15P SDR	8192x3834	8L	4K SDR	4M HDR	4M SDR	2M	2M		471.9	4608 tile	4608 tile					
20K@20P SDR	4432x4446	8L	4K HDR						394.1	4432						
16K@30P SDR	5430x3054	8L							497.5	4608 tile						
12K@40P SDR	4512x2512	8L							453.4	4512						
12K@30P SDR(2I-HDR)	4512x2512	8L							340	4512						
8K@30P SDR(2I-HDR)	3856x2220	8L							256.8	3856						
2 sensor(8L+4L 2x4L)-->8Kp15+4Mp60																
2*8M(8 56M)@30P SDR	3856x2220	4L	4L						513.6	3856	3856					
2*4K(8 29M)@30P SDR	3840x2160	4L	4L						497.7	3840	3840					
3 sensor(3x4L)-->3x4Mp60																
3*4M@30P (2I-HDR)	2688x1520	4L	4L	4L					367.7	2688	2688	2688				
2*4M@30P (2I-HDR)+TTL 2M	2688x1520	4L	4L				TTL		308.4	2688	2688				1920	1920
4 sensor(4x2L)-->4x4Mp30 or 4x2Mp60																
4*4M@30P SDR	2688x1520	2L	2L	2L	2L				490.3	2688	2688	2688	2688			
4*2M@30P (2I-HDR)	1920x1080	2L	2L	2L	2L				253	1920	1920	1920	1920			
2*4M@30P (2I-HDR)+2*2M@30P (2I-HDR)	1920x1080	4L	4L	2L	2L				371.6	2688	2688	1920	1920			
5 or 6 sensor(6x2L)-->6x2Mp60																
4M@30P (2I-HDR)+4*2M@30P(2I-HDR)	2688x1520	4L	2L	2L	2L	2L			375.4	2688	1920	1920	1920	1920		
4*4M@30P SDR+2M@30P(2I-HDR)	2688x1520	2L	2L	2L	2L	2L	TTL		553.6	2688	2688	2688	2688		1920	1920
6*2M@30P(2I-HDR)	1920x1080	2L	2L	2L	2L	2L	2L		379.2	1920	1920	1920	1920	1920	1920	

图 2.4: sensor 详细参数图

2.2 SDK 介绍

2.2.1 sensor_cfg.ini 文件

MIPI 接口的 ini 范例如下：

```
; section for source
[source]
;type = SOURCE_USER_FE
dev_num = 1
en_mode = 0
; section for sensor
[sensor]
; sensor name
name = PR2100_MIPI_2M_12FPS_8BIT

bus_id = 2
sns_i2c_addr = 0x3c

mipi_dev = 0

lane_id = 0, 1, 2, 3, 4, -1, -1, -1, -1
pn_swap = 1, 1, 1, 1, 1, 0, 0, 0, 0

mclk_en = 1
mclk = 0

rst_pin = 69
rst_active= 0
```

图 2.5: MIPI 接口的 ini 范例

【定义】

成员名称	描述
dev_mun	表示当前组合 sensor 的个数
en_mode	表示当前 sensor 组合使用的 mode，参考第一章 < 多 sensor 组合 >
[sensor]	表示当前是哪一个 sensor 使用的参数
name	表示当前 sensor 的 ID，详见 SDK 的 sample_common.h
busid	表示当前 sensor 使用的 I2C 总线号
sns_i2c_addr	表示当前 sensor 使用的 I2C 总线号上的 I2C 地址
mipi_dev	表示当前 sensor 使用的 MAC 编号，参考第一章 < 多 sensor 组合 >
lane_id	表示当前 sensor 对应 mipi_rx 的数据编号
pn_swap	表示当前 sensor 对应 mipi_rx 的数据是否需要翻转
mclk_en	表示使能哪一组 mclk 输出
mclk	表示当前 sensor 使用哪一组 mclk 作为参考时钟
rst_pin	表示当前 sensor 的 ret 脚是哪个管脚
rst_active	表示当前 sensor 的 ret 引脚是高电平还是低电平有效

BT/DC 接口的 ini 范例如下：

```
; section for source
[source]
;type = SOURCE_USER_FE
dev_num = 1
;en_mode = 0
; section for sensor
[sensor]
; sensor name
name = NEXTCHIP_N5_2M_25FPS_8BIT

;bus/i2c dev number
bus_id = 0
sns_i2c_addr = 50

mipi_dev = 6
func_id = -1,-1,-1,-1,
          1, 2, 3, 4,
          5, 6, 7, 8,
          -1,-1,-1,-1,
          -1,-1,-1,-1
```

图 2.6: BT/DC 接口的 ini 范例

【定义】

成员名称	描述
dev_mun	表示当前组合 sensor 的个数
en_mode	表示当前 sensor 组合使用的 mode，参考第一章 < 多 sensor 组合 >
[sensor]	表示当前是哪一个 sensor 使用的参数
name	表示当前 sensor 的 ID，详见 SDK 的 sample_common.h
busid	表示当前 sensor 使用的 I2C 总线号
sns_i2c_addr	表示当前 sensor 使用的 I2C 总线号上的 I2C 地址
mipi_dev	表示当前 sensor 使用的 MAC 编号，参考第一章 < 多 sensor 组合 >
func_id	表示当前 sensor 对应 GPIO_data 的数据编号

2.2.2 stWDRAttr 的 WDR 模式

WDR_MODE_X To1_LINE:X 条 Line 合成一条 line。(现在只支持 2to1line)

WDR_MODE_X To1_FRAME:X 个 frame 合成一个 frame (暂时不支持)

名称	描述
enBayerFormat	BAYER 阵列的排布格式
chn_num	sensor 的 channel 数
sbrFps	sensor 的 fps
_SAMPLE_INI_CFG_S	初始化结构体
enSource	设置图像数据源
enSnsMode	sensor 工作模式
devNum	设备数量
s32BusId	表示当前 sensor 使用的 I2C 总线号
enSnsType	使用的 sensor 类型
enWDRMode	WDR 模式
s32SnsI2cAddr	设备的 I2C 地址
MipiDev	mipi 设备号
as16LaneId	Mipi snesor 使用的 laneid
as16FuncId	DVPsensor 使用的 laneid
as8PNSwap	Lane 的 P 线和 N 线是否置换
stMclkAttr	MAC Clock 属性
u8Orien	是否设置旋转, 包括折叠镜像

3 MIPI 使用指南

3.1 重要概念

- MIPI

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

- Lane

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

- LVDS

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

- 同步码

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

- 使用 SOF/EOF 表示一帧的开始与结束。使用 SOL/EOL 表示行的开始与结束。
- 使用 SAV invalid 与 EAV invalid 表示 VBLANK 的开始与结束。使用 SAV valid 与 EAV valid 表示有效数据 (information line, H.OB 与 pixel data) 的开始与结束。

VBLANK				
HBLANK	SOF	Active Line 1	EOL	HBLANK
HBLANK	SOL	Active Line 2		HBLANK
HBLANK		Active Line 3		HBLANK
⋮		⋮		⋮
HBLANK		Active Line P-1		HBLANK
HBLANK		Active Line P	EOF	HBLANK
VBLANK				

图 3.1: SOF/EOF/SOL/EOL 同步方式

HBLANK	SAV Invalid	VBLANK	EAV Invalid	HBLANK
HBLANK		VBLANK		HBLANK
HBLANK		⋮		HBLANK
⋮		VBLANK		⋮
HBLANK	SAV Valid	Frame Information Line	EAV Valid	HBLANK
HBLANK		OB/ effective pixel		HBLANK
HBLANK		OB/ effective pixel		HBLANK
HBLANK		⋮		HBLANK
HBLANK		OB/ effective pixel		HBLANK
HBLANK	SAV Invalid	VBLANK	EAV Invalid	HBLANK
HBLANK		VBLANK		HBLANK
HBLANK		⋮		HBLANK
HBLANK		VBLANK		HBLANK

图 3.2: SAV/EAV 同步方式

· DOI

SONY 的交错式 WDR 模式，全称为 Digital Overlap。

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

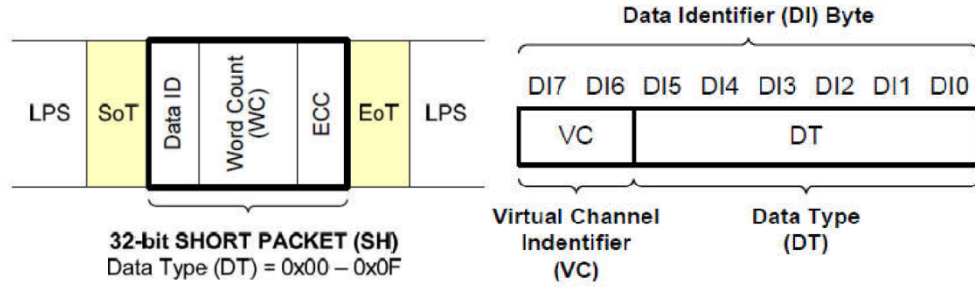


图 3.3: MIPI-CSI 的 channel ID

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

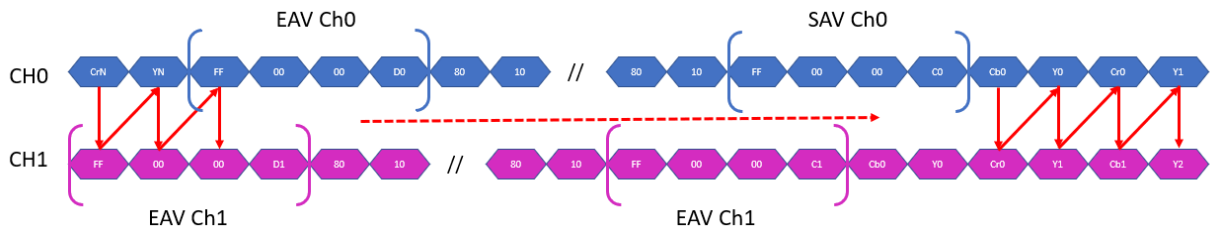


图 3.4: demux 模式支持的输入

3.2 API 参考

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

- CVI_MIPI_SET_DEV_ATTR : 设置 MIPI、并口、输入模式、图小大小等设备属性。
- CVI_MIPI_SET_OUTPUT_CLK_EDGE : 设置输出时钟电平变化触发状态。
- CVI_MIPI_RESET_SENSOR : 复位 SENSOR。
- CVI_MIPI_UNRESET_SENSOR : 撤销复位 SENSOR。
- CVI_MIPI_RESET_MIPI : 复位 MIPI Rx。
- CVI_MIPI_ENABLE_SENSOR_CLOCK : 打开 SENSOR 的时钟。
- CVI_MIPI_DISABLE_SENSOR_CLOCK : 关闭 SENSOR 的时钟。
- CVI_MIPI_SET_CROP_TOP : 舍弃每帧中的首 N 条资料。
- CVI_MIPI_SET_WDR_MANUAL : 打开 WDR 手动模式。
- CVI_MIPI_SET_LVDS_FP_VS : 设定 Sub-LVDS 中 VSYNC 生成的时间点。
- CVI_MIPI_SET_BT_FMT_OUT : 设置输出并口的格式。

- `CVI_MIPI_GET_CIF_ATTR` : 获取 CIF 的属性。
- `CVI_MIPI_SET_SENSOR_CLOCK` : 设置 SENSOR 的时钟。
- `CVI_MIPI_SET_MAX_MAC_CLOCK` : 设置 MAC 高速时钟。
- `CVI_MIPI_SET_CROP_WINDOW` : 设置裁剪窗口的长宽, 位置。
- `CVI_MIPI_SET_YUV_SWAP` : 设置 yuv 中不同分量的互换。
- `CVI_MIPI_RESET_LVDS` : 重置 LVDS 接口。

MIPI Tx 提供对接显示屏、级联的功能。提供函数如下:

- `mipi_tx_cfg` : 配置 MIPI TX 设备。
- `mipi_tx_send_cmd` : 向 MIPI TX 发送命令。
- `mipi_tx_recv_cmd` : 从 MIPI TX 接收命令。
- `mipi_tx_enable` : 使能 MIPI TX。
- `mipi_tx_disable` : 禁用 MIPI TX。
- `mipi_tx_set_hs_settle` : 设置 HS settle 配置。
- `mipi_tx_get_hs_settle` : 获取 HS settle 配置。
- `mipi_tx_suspend` : 挂起 MIPI TX。
- `mipi_tx_resume` : 恢复 MIPI TX。

3.2.1 CVI_MIPI_SET_DEV_ATTR

【描述】

设置 MIPI、并口、输入模式、图小大小等设备属性。

【定义】

```
#define CVI_MIPI_SET_DEV_ATTR_IOW(CVI_MIPI_IOC_MAGIC, 0x01, struct combo_dev_attr_s)
↪attr_s)
```

【参数】

参数名称	描述	输入/输出
<code>CVI_MIPI_IOC_MAGIC</code>	宏定义, 用于标识 MIPI 设备的 IO 控制命令。	输入
<code>combo_dev_attr_s</code>	代表设备属性的结构体, 包含了 mipi, lvds, ttl 等设备结构体, 规定了输入模式以及图片大小等。	输入

【返回值】

返回值	描述
0	成功
非 0	失败, 请参考错误码

【需求】

- 头文件: cif_uapi.h

【注意】

配置 CVI_MIPI_SET_DEV_ATTR 之前, 需要使用 ISP 接口打开 MIPI_RX 时钟.
详情请见 ISP 相关文件。

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

【举例】

无

【相关主题】

combo_dev_attr_s

3.2.2 CVI_MIPI_SET_OUTPUT_CLK_EDGE

【描述】

设置输出时钟电平变化触发状态。

【定义】

```
#define CVI_MIPI_SET_OUTPUT_CLK_EDGE_IOW(CVI_MIPI_IOC_MAGIC, 0x02, struct _  
↪clk_edge_s)
```

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义，用于标识 MIPI 设备的 IO 控制命令。	输入
clk_edge_s	代表时钟电平的结构体，包含了上升沿触发、下降沿触发、不变三种状态。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：cif_uapi.h

【注意】

需要先使能时钟。

【举例】

无

【相关主题】

clk_edge_s

3.2.3 CVI_MIPI_RESET_SENSOR

【描述】

复位 SENSOR。

【定义】

```
#define CVI_MIPI_RESET_SENSOR_IOW(CVI_MIPI_IOC_MAGIC, 0x05, struct sns_rst_  
↪config)
```

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义，用于标识 MIPI 设备的 IO 控制命令。	输入
sns_rst_config	代表传感器复位配置的结构体，包含了设备号、GPIO 引脚号、GPIO 电平枚举。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：cif_uapi.h

【注意】

无

【举例】

无

【相关主题】

无

3.2.4 CVI_MIPI_UNRESET_SENSOR

【描述】

撤销复位 SENSOR。

【定义】

```
#define CVI_MIPI_UNRESET_SENSOR_IOW(CVI_MIPI_IOC_MAGIC, 0x06, struct sns_rst_  
↪ config)
```

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义，用于标识 MIPI 设备的 IO 控制命令。	输入
sns_rst_config	代表传感器复位配置的结构体，包含了设备号、GPIO 引脚号、GPIO 电平枚举。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件: cif_uapi.h

【注意】

无

【举例】

无

【相关主题】

无

3.2.5 CVI_MIPI_RESET_MIPI

【描述】

复位 MIPI Rx。

【定义】

```
#define CVI_MIPI_RESET_MIPI_IOW(CVI_MIPI_IOC_MAGIC, 0x07, unsigned int)
```

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义，用于标识 MIPI 设备的 IO 控制命令。	输入
无符号整型	MIPI Rx 设备号。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件: cif_uapi.h

【注意】

无

【举例】

无

【相关主题】

无

3.2.6 CVI_MIPI_ENABLE_SENSOR_CLOCK

【描述】

使能 SENSOR 的时钟。

【定义】

```
#define CVI_MIPI_ENABLE_SENSOR_CLOCK_IOW(CVI_MIPI_IOC_MAGIC, 0x10, unsigned_↵  
↵int)
```

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义，用于标识 MIPI 设备的 IO 控制命令。	输入
无符号整型	MIPI Rx 设备号。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：cif_uapi.h

【注意】

无

【举例】

无

【相关主题】

无

3.2.7 CVI_MIPI_DISABLE_SENSOR_CLOCK

【描述】

关闭 SENSOR 的时钟。

【定义】

```
#define CVI_MIPI_DISABLE_SENSOR_CLOCK_IOW(CVI_MIPI_IOC_MAGIC, 0x11, unsigned_↵  
↵int)
```

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义，用于标识 MIPI 设备的 IO 控制命令。	输入
无符号整型	MIPI Rx 设备号。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：cif_uapi.h

【注意】

无

【举例】

无

【相关主题】

无

3.2.8 CVI_MIPI_SET_CROP_TOP

【描述】

舍弃每帧中的首 N 条资料。

【定义】

```
#define CVI_MIPI_SET_CROP_TOP_IOW(CVI_MIPI_IOC_MAGIC, 0x20, struct crop_top_s)
```

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义，用于标识 MIPI 设备的 IO 控制命令。	输入
crop_top_s	代表 crop 控制寄存器的结构体，包含了设备号，寄存器号等信息，用来决定裁剪的信息行数。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：cif_uapi.h

【注意】

无

【举例】

无

【相关主题】

crop_top_s

3.2.9 CVI_MIPI_SET_WDR_MANUAL

【描述】

打开 WDR 手动模式。

【定义】

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

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义，用于标识 MIPI 设备的 IO 控制命令。	输入
manual_wdr_s	代表 wdr 配置相关的结构体，包含了使能，长短曝间隔，填充行数等信息。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

· 头文件：cif_uapi.h

【注意】

无

【举例】

无

【相关主题】

manual_wdr_s

3.2.10 CVI_MIPI_SET_LVDS_FP_VS

【描述】

设定 Sub-LVDS 中 VSYNC 生成的时间点。

【定义】

```
#define CVI_MIPI_SET_LVDS_FP_VS _IOW(CVI_MIPI_IOC_MAGIC, 0x22, struct vsync_gen_
↪s)
```

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义，用于标识 MIPI 设备的 IO 控制命令。	输入
vsync_gen_s	代表 LVDS 信号同步属性的结构体，包含设备号和脉冲前消隐区行数的信息。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：cif_uapi.h

【注意】

无

【举例】

无

【相关主题】

vsync_gen_s

3.2.11 CVI_MIPI_SET_BT_FMT_OUT

【描述】

设置并口输出格式。

【定义】

```
#define CVI_MIPI_SET_BT_FMT_OUT _IOW(CVI_MIPI_IOC_MAGIC, 0x24, struct bt_fmt_
↪out_s)
```

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义，用于标识 MIPI 设备的 IO 控制命令。	输入
bt_fmt_out_s	代表并口格式的结构体，包含了 CBYCRY、CRYCBY、YCBYCR 和 YCRYCB 的格式。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：cif_uapi.h

【注意】

无

【举例】

无

【相关主题】

bt_fmt_out_s

3.2.12 CVI_MIPI_GET_CIF_ATTR

【描述】

获取 CIF 的属性。

【定义】

```
#define CVI_MIPI_GET_CIF_ATTR \_IOWR(CVI_MIPI_IOC_MAGIC, 0x25, struct cif_attr_s)
```

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义，用于标识 MIPI 设备的 IO 控制命令。	输入
cif_attr_s	代表 cif 属性的结构体，包含了设备号和信号同步信息。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

· 头文件: cif_uapi.h

【注意】

无

【举例】

无

【相关主题】

cif_attr_s

3.2.13 CVI_MIPI_SET_SENSOR_CLOCK

【描述】

设置 SENSOR 的时钟。

【定义】

```
#define CVI_MIPI_SET_SENSOR_CLOCK_IOW(CVI_MIPI_IOC_MAGIC, 0x26, struct mclk_  
→pll_s)
```

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义，用于标识 MIPI 设备的 IO 控制命令。	输入
mclk_pll_s	代表相机时钟的结构体，包含了相机设备号和不同种类的频率信息。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

· 头文件: cif_uapi.h

【注意】

无

【举例】

无

【相关主题】

mclk_pll_s

3.2.14 CVI_MIPI_SET_MAX_MAC_CLOCK

【描述】

设置 MAC 高速时钟。

【定义】

```
#define CVI_MIPI_SET_MAX_MAC_CLOCK \_IOW(CVI_MIPI_IOC_MAGIC, 0x27, unsigned_  
↪int)
```

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义，用于标识 MIPI 设备的 IO 控制命令。	输入
无符号整型	根据整型数据设置最大 mac 时钟频率。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：cif_uapi.h

【注意】

无

【举例】

无

【相关主题】

无

3.2.15 CVI_MIPI_SET_CROP_WINDOW

【描述】

设置裁剪窗口的长宽，位置。

【定义】

```
#define CVI_MIPI_SET_CROP_WINDOW \_IOW(CVI_MIPI_IOC_MAGIC, 0x28, struct cif_  
↪crop_win_s)
```

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义，用于标识 MIPI 设备的 IO 控制命令。	输入
cif_crop_win_s	代表裁剪窗口的结构体，包含了设备号、使能情况、长宽大小和位置节点的信息。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：cif_uapi.h

【注意】

无

【举例】

无

【相关主题】

cif_crop_win_s

3.2.16 CVI_MIPI_SET_YUV_SWAP

【描述】

设置 yuv 不同分量的交换。

【定义】

```
#define CVI_MIPI_SET_YUV_SWAP \_IOW(CVI_MIPI_IOC_MAGIC, 0x29, struct cif_yuv_
↪swap_s)
```

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义，用于标识 MIPI 设备的 IO 控制命令。	输入
cif_yuv_swap_s	代表 cif 中 yuv 分量互换的结构体，包含了设备号，亮度和色度分量交换等信息。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

· 头文件: cif_uapi.h

【注意】

暂时无法使用 uv swap, 支持 yc swap。

【举例】

无

【相关主题】

cif_yuv_swap_s

3.2.17 CVI_MIPI_RESET_LVDS

【描述】

重置 LVDS 接口。

【定义】

```
#define CVI_MIPI_RESET_LVDS_IOW(CVI_MIPI_IOC_MAGIC, 0x23, unsigned int)
```

【参数】

参数名称	描述	输入/输出
CVI_MIPI_IOC_MAGIC	宏定义, 用于标识 MIPI 设备的 IO 控制命令。	输入
无符号整型	LVDS 设备号。	输入

【返回值】

返回值	描述
0	成功
-1	失败, 并设置 errno

【需求】

头文件: cif_uapi.h

【注意】

无

3.2.18 mipi_tx_cfg

【描述】

配置 MIPI TX 设备。

【定义】

```
int mipi_tx_cfg(int fd, struct combo_dev_cfg_s *dev_cfg)
```

【参数】

参数名称	描述	输入/输出
fd	设备文件描述符。	输入
dev_cfg	设备配置信息。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件: cvi_mipi_tx.h, cvi_comm_mipi_tx.h
- 库文件: libmipi_tx.a

【注意】

无

【举例】

无

【相关主题】

无

3.2.19 mipi_tx_send_cmd

【描述】

向 MIPI TX 发送命令。

【定义】

```
int mipi_tx_send_cmd(int fd, struct cmd_info_s *cmd_info)
```

【参数】

参数名称	描述	输入/输出
fd	设备文件描述符。	输入
cmd_info	命令信息。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件: `cvi_mipi_tx.h`, `cvi_comm_mipi_tx.h`
- 库文件: `libmipi_tx.a`

【注意】

无

【举例】

无

【相关主题】

`mipi_tx_recv_cmd`

3.2.20 `mipi_tx_recv_cmd`

【描述】

从 MIPI TX 接收命令。

【定义】

```
int mipi_tx_recv_cmd(int fd, struct get_cmd_info_s *cmd_info)
```

【参数】

参数名称	描述	输入/输出
<code>fd</code>	设备文件描述符。	输入
<code>cmd_info</code>	接收的命令信息。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件: `cvi_mipi_tx.h`, `cvi_comm_mipi_tx.h`
- 库文件: `libmipi_tx.a`

【注意】

无

【举例】

无

【相关主题】

[mipi_tx_send_cmd](#)

3.2.21 mipi_tx_enable

【描述】

使能 MIPI TX。

【定义】

```
int mipi_tx_enable(int fd)
```

【参数】

参数名称	描述	输入/输出
fd	设备文件描述符。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：cvi_mipi_tx.h, cvi_comm_mipi_tx.h
- 库文件：libmipi_tx.a

【注意】

无

【举例】

无

【相关主题】

[mipi_tx_disable](#)

3.2.22 mipi_tx_disable

【描述】

禁用 MIPI TX。

【定义】

```
int mipi_tx_disable(int fd)
```

【参数】

参数名称	描述	输入/输出
fd	设备文件描述符。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：cvi_mipi_tx.h, cvi_comm_mipi_tx.h
- 库文件：libmipi_tx.a

【注意】

无

【举例】

无

【相关主题】

[mipi_tx_enable](#)

3.2.23 mipi_tx_set_hs_settle

【描述】

设置 HS settle 配置。

【定义】

```
int mipi_tx_set_hs_settle(int fd, const struct hs_settle_s *hs_cfg)
```

【参数】

参数名称	描述	输入/输出
fd	设备文件描述符。	输入
hs_cfg	HS settle 配置。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件: `cvi_mipi_tx.h`, `cvi_comm_mipi_tx.h`
- 库文件: `libmipi_tx.a`

【注意】

无

【举例】

无

【相关主题】

`mipi_tx_get_hs_settle`

3.2.24 `mipi_tx_get_hs_settle`

【描述】

获取 HS settle 配置。

【定义】

```
int mipi_tx_get_hs_settle(int fd, struct hs_settle_s *hs_cfg)
```

【参数】

参数名称	描述	输入/输出
<code>fd</code>	设备文件描述符。	输入
<code>hs_cfg</code>	HS settle 配置。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件: `cvi_mipi_tx.h`, `cvi_comm_mipi_tx.h`
- 库文件: `libmipi_tx.a`

【注意】

无

【举例】

无

【相关主题】

[mipi_tx_set_hs_settle](#)

3.2.25 mipi_tx_suspend

【描述】

挂起 MIPI TX。

【定义】

```
int mipi_tx_suspend(int fd)
```

【参数】

参数名称	描述	输入/输出
fd	设备文件描述符。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：cvi_mipi_tx.h, cvi_comm_mipi_tx.h
- 库文件：libmipi_tx.a

【注意】

无

【举例】

无

【相关主题】

[mipi_tx_resume](#)

3.2.26 mipi_tx_resume

【描述】

恢复 MIPI TX。

【定义】

```
int mipi_tx_resume(int fd)
```

【参数】

参数名称	描述	输入/输出
fd	设备文件描述符。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件: `cvi_mipi_tx.h`, `cvi_comm_mipi_tx.h`
- 库文件: `libmipi_tx.a`

【注意】

无

【举例】

无

【相关主题】

[mipi_tx_suspend](#)

3.3 数据类型

MIPI RX 相关数据类型定义如下:

- `combo_dev_attr_s`: combo 设备各类属性组合的结构体。
- `input_mode_e`: MIPI Rx 输入接口类型枚举。
- `rx_mac_clk_e`: MAC 支持的工作时钟频率枚举。
- `mclk_pll_s`: MIPI Rx 输出 SENSOR 参考时钟设定结构体。
- `cam_pll_freq_e`: MIPI Rx 输出的 SENSOR 参考时钟枚举。
- `mipi_dev_attr_s`: MIPI-CSI 设备属性设置结构体。

- `raw_data_type_e`:MIPI Rx 输入图像格式类型枚举。
- `mipi_wdr_mode_e`: 宽动态范围模式枚举。
- `dphy_s`:MIPI Rx DPHY 属性结构体。
- `lvds_fid_type_s`:LVDS 场 ID 类型属性。
- `lvds_fid_type_e`:LVDS 场 ID 类型枚举。
- `mipi_demux_info_s`:MIPI CSI 使用 Virtual Channel 解多任务的属性结构体。
- `lvds_dev_attr_s`:LVDS/SubLVDS 设备属性结构体。
- `lvds_sync_mode_e`:LVDS 同步模式枚举。
- `lvds_bit_endian`:lvds 协议同步码大小端。
- `lvds_vsync_type_s`:LVDS WDR 同步参数。
- `lvds_vsync_type_e`:LVDS 在 WDR 模式的同步方式。
- `ttl_dev_attr_s`:TTL/BT 接口的属性设定结构体。
- `ttl_src_e`:TTL/BT 数据源枚举。
- `ttl_fmt_e`:TTL/BT 同步码方式枚举。
- `ttl_VIOclk_e`:TTL/BT 的时钟枚举。
- `bt_demux_attr_s`:BT 解多任务模式的属性设定结构体。
- `bt_demux_mode_e`:BT 解多任务模式的信道数量枚举。
- `bt_demux_sync_s`:BT 解多任务模式的同步码结构体。
- `img_size_s`:MIPI Rx 输入数据每帧的大小结构体。
- `manual_wdr_attr_s`: 手动宽动态范围属性设置结构体。
- `clk_edge_s`: 设备时钟电平信息结构体。
- `clk_edge_e`: 电平采样枚举。
- `sns_rst_config`: 传感器复位配置结构体。
- `sns_rst_active_e`: 传感器复位电平枚举。
- `crop_top_s`: 数据开头裁剪信息结构体。
- `manual_wdr_s`: 手动 WDR 模式设定结构体。
- `vsync_gen_s`:LVDS 同步时间点设置结构体。
- `bt_fmt_out_s`: 设备并口输出格式信息结构体。
- `bt_fmt_out_e`: 输出格式枚举。
- `cif_attr_s`:cif 同步信息属性结构体。
- `cif_crop_win_s`: 裁剪窗口属性结构体。
- `cif_yuv_swap_s`:yuv 图像不同分量交换结构体。
- `SNS_ATTR_S`:16 位参数的基本传感器属性结构体。
- `SNS_ATTR_LARGE_S`:32 位参数的扩展传感器属性结构体。

- `ISP_SNS_STATE_S`:ISP 传感器状态结构体 - 跟踪当前传感器状态。
- `ISP_SNS_MIRRORFLIP_TYPE_E`: 传感器镜像/翻转模式枚举。
- `ISP_SNS_INTTIME_MODE_E`: 长曝光到短曝光模式枚举。
- `MCLK_ATTR_S`:MCLK (Master Clock) 时钟属性结构体。
- `RX_INIT_ATTR_S`:MIPI/TTL 接口的接收器初始化属性结构体。
- `SNS_BDG_MUX_MODE_E`: 传感器桥接多路复用器模式枚举。
- `ISP_INIT_ATTR_S`:ISP 初始化属性结构体。
- `ISP_SNS_OBJ_S`:ISP 传感器对象结构体 - 包含传感器操作的函数指针。

MIPI TX 相关数据类型定义如下:

- `MIPI_TX0_NAME`:MIPI TX 设备 0 的路径定义。
- `MIPI_TX1_NAME`:MIPI TX 设备 1 的路径定义。
- `CMD_MAX_NUM`: 最大命令数量定义。
- `RX_MAX_NUM`: 最大接收通道数定义。
- `LANE_MAX_NUM`: 最大通道数定义。
- `output_mode_e`: 输出模式枚举类型。
- `video_mode_e`: 视频模式枚举类型。
- `output_format_e`: 输出格式枚举类型。
- `mipi_tx_lane_id`:MIPI TX 通道 ID 枚举类型。
- `sync_info_s`: 视频同步时序参数结构体。
- `combo_dev_cfg_s`:MIPI TX 设备配置参数结构体。
- `cmd_info_s`: 命令信息结构体。
- `get_cmd_info_s`: 获取命令信息结构体。
- `hs_settle_s`: 高速时序配置结构体。
- `dsc_instr`:DSC 指令结构体。

3.3.1 combo_dev_attr_s

【说明】

combo 设备各类属性组合的结构体。

【定义】

```
struct combo_dev_attr_s {
    enum input_mode_e input_mode;

    enum rx_mac_clk_e mac_clk;
```

(下页继续)

(续上页)

```
struct mclk_pll_s mclk;

union {

    struct mipi_dev_attr_s mipi_attr;

    struct lvds_dev_attr_s lvds_attr;

    struct ttl_dev_attr_s ttl_attr;

    struct bt_demux_attr_s bt_demux_attr;

};

unsigned int devno;

unsigned int cif_mode;

struct img_size_s img_size;

struct manual_wdr_attr_s wdr_manu;

};
```

【成员】

成员名称	描述
input_mode	输入接口类型
mac_clk	MIPI RX MAC 时钟设定
mclk	MIPI RX 输出的 Sensor 参考时钟设定
mipi_attr	如果 input_mode 配置为 INPUT_MODE_MIPI, 则必须配置 mipi_attr
lvds_attr	如果 input_mode 配置为 INPUT_MODE_SUBLVDS, 则必须配置 lvds_attr
ttl_attr	TTL/BT 接口的属性
bt_demux_attr	BT 解多任务模式的属性
devno	MIPI-Rx 设备号
cif_mode	cif 模式
img_size	输入帧大小
wdr_manu	手动 WDR 属性

【注意事项】

无

【相关数据类型及接口】

combo_dev_attr_s; CVI_MIPI_SET_DEV_ATTR

3.3.2 input_mode_e

【说明】

MIPI Rx 输入接口类型枚举。

【定义】

```
enum input_mode_e {  
  
    INPUT_MODE_MIPI = 0,  
  
    INPUT_MODE_SUBLVDS,  
  
    INPUT_MODE_HISPI,  
  
    INPUT_MODE_CMOS,  
  
    INPUT_MODE_BT1120,  
  
    INPUT_MODE_BT601,  
  
    INPUT_MODE_BT656_9B,  
  
    INPUT_MODE_BT656_9B_DDR,  
  
    INPUT_MODE_CUSTOM_0,  
  
    INPUT_MODE_BT_DEMUX,  
  
    INPUT_MODE_BUTT  
};
```

【成员】

成员名称	描述
INPUT_MODE_MIPI	MIPI 模式
INPUT_MODE_SUBLVDS	SUBLVDS 模式
INPUT_MODE_CMOS	CMOS 模式
INPUT_MODE_BT1120	BT1120 模式
INPUT_MODE_BT656_9B	BT656_9B 模式
INPUT_MODE_BT656_9B_DDR	BT656_9B_DDR 模式
INPUT_MODE_CUSTOM_0	CUSTOM 模式
INPUT_MODE_BT_DEMUX	BT_DEMUX 模式
INPUT_MODE_BUTT	空，代表结尾

【注意事项】

无

【相关数据类型及接口】

combo_dev_attr_s; CVI_MIPI_SET_DEV_ATTR

3.3.3 rx_mac_clk_e

【说明】

MAC 支持的工作时钟频率枚举。

【定义】

```
enum rx_mac_clk_e {  
  
    RX_MAC_CLK_150M = 0,  
  
    RX_MAC_CLK_200M,  
  
    RX_MAC_CLK_300M,  
  
    RX_MAC_CLK_400M,  
  
    RX_MAC_CLK_500M,  
  
    RX_MAC_CLK_600M,  
  
    RX_MAC_CLK_900M,  
  
    RX_MAC_CLK_BUTT,  
  
};
```

【成员】

成员名称	描述
RX_MAC_CLK_150M	150M 频率
RX_MAC_CLK_200M	200M 频率
RX_MAC_CLK_300M	300M 频率
RX_MAC_CLK_400M	400M 频率
RX_MAC_CLK_500M	500M 频率
RX_MAC_CLK_600M	600M 频率
RX_MAC_CLK_900M	900M 频率
INPUT_MODE_BUTT	空，代表结尾

【注意事项】

MAC 时钟与支持的 MIPI 时钟关系请参考手册。

【相关数据类型及接口】

combo_dev_attr_s; CVI_MIPI_SET_DEV_ATTR

3.3.4 mclk_pll_s

【说明】

MIPI Rx 输出 SENSOR 参考时钟设定结构体。

【定义】

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

【成员】

成员名称	描述
cam	0: 输出为 CAM_MCLK0 1: 输出为 CAM_MCLK1
freq	输出的 SENSOR 参考时钟

【注意事项】

无

【相关数据类型及接口】

combo_dev_attr_s, cam_pll_freq_e; CVI_MIPI_SET_DEV_ATTR

3.3.5 cam_pll_freq_e

【说明】

MIPI Rx 输出的 SENSOR 参考时钟枚举。

【定义】

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

(下页继续)

(续上页)

};

【成员】

成员名称	描述
CAMPLL_FREQ_NONE	不需要 clock
CAMPLL_FREQ_37P125M	37.125M 频率
CAMPLL_FREQ_25M	25M 频率
CAMPLL_FREQ_27M	27M 频率
CAMPLL_FREQ_24M	24M 频率
CAMPLL_FREQ_26M	26M 频率
CAMPLL_FREQ_NUM	时钟频率数量

【注意事项】

无

【相关数据类型及接口】

mclk_pll_s

3.3.6 mipi_dev_attr_s

【说明】

MIPI-CSI 设备属性设置结构体。

【定义】

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

【成员】

成员名称	描述
raw_data_type	传输的图像数据类型枚举
lane_id	发送端 (sensor) 和接收端 (MIPI Rx) lane 的对应关系
wdr_mode	WDR 模式
data_type	当 WDR 模式为 CVI_MIPI_WDR_MODE_DT 时, 每个 WDR frames 对应的数据类型
data_type pn_swap	P/N 差分通道是否交换
dphy	使能 DPHY 模式
demux	解多任务复用信息

【注意事项】

支持的最大 MIPI-CSI 通道数 MIPI_LANE_NUM 为 8。用于宽动态范围 (WDR) 模式时, 虚拟通道数量 WDR_VC_NUM 为 2。

【相关数据类型及接口】

combo_dev_attr_s, raw_data_type_e, mipi_wdr_mode_e, dphy_s, mipi_demux_info_s

3.3.7 raw_data_type_e

【说明】

MIPI Rx 输入图像格式类型枚举。

【定义】

```
enum raw_data_type_e {  
    RAW_DATA_8BIT = 0,  
    RAW_DATA_10BIT,  
    RAW_DATA_12BIT,  
    RAW_DATA_16BIT,  
    YUV422_8BIT,  
    YUV422_10BIT,  
    RAW_DATA_BUTT  
};
```

【成员】

成员名称	描述
RAW_DATA_8BIT	raw 格式图片 8 比特
RAW_DATA_10BIT	raw 格式图片 10 比特
RAW_DATA_12BIT	raw 格式图片 12 比特
RAW_DATA_16BIT	raw 格式图片 16 比特
YUV422_8BIT	yuv422 格式图片 8 比特
YUV422_10BIT	yuv422 格式图片 10 比特
RAW_DATA_BUTT	空，代表结尾

【注意事项】

YUV422_8BIT 与 YUV422_10BIT 只支持 MIPI-CSI 格式。

【相关数据类型及接口】

mipi_dev_attr_s

3.3.8 mipi_wdr_mode_e

【说明】

宽动态范围模式枚举。

【定义】

```
enum mipi_wdr_mode_e {
    CVI_MIPI_WDR_MODE_NONE = 0,
    CVI_MIPI_WDR_MODE_VC,
    CVI_MIPI_WDR_MODE_DT,
    CVI_MIPI_WDR_MODE_DOL,
    CVI_MIPI_WDR_MODE_MANUAL,
    CVI_MIPI_WDR_MODE_BUTT
};
```

【成员】

成员名称	描述
CVI_MIPI_WDR_MODE_NONE	线性模式
CVI_MIPI_WDR_MODE_VC	VC WDR 模式
CVI_MIPI_WDR_MODE_DT	DT WDR 模式
CVI_MIPI_WDR_MODE_DOL	DOL WDR 模式
CVI_MIPI_WDR_MODE_MANUAL	手动模式
CVI_MIPI_WDR_MODE_BUTT	空，代表结尾

【注意事项】

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

【相关数据类型及接口】

mipi_dev_attr_s, lvds_dev_attr_s

3.3.9 dphy_s

【说明】

MIPI Rx DPHY 属性结构体。

【定义】

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

【成员】

成员名称	描述
enable	开启 DPHY 属性设定
hs_settle	设置 hs_settle 数值

【注意事项】

无

【相关数据类型及接口】

mipi_dev_attr_s

3.3.10 lvds_fid_type_s

【说明】

LVDS 字段 ID 类型属性。

【定义】

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

【成员】

成员名称	描述
fid	帧 ID 类型

【注意事项】

· 无

【相关数据类型及接口】

lvds_dev_attr_s

3.3.11 lvds_fid_type_e

【说明】

LVDS 字段 ID 类型枚举。

【定义】

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

【成员】

成员名称	描述
LVDS_FID_NONE	无帧 ID
LVDS_FID_IN_SAV	帧 ID 在 SAV 中
LVDS_FID_BUTT	无效类型

【注意事项】

· 无

【相关数据类型及接口】

lvds_fid_type_s

3.3.12 mipi_demux_info_s

【说明】

MIPI CSI 使用 Virtual Channel 解多任务的属性结构体。

【定义】

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

【成员】

成员名称	描述
demux_en	开启 MIPI Virtual channel 解多任务
vc_mapping	Virtual channel 状态映射信息，设定 ISP channel 与 mipi Virtual channel 对应关系。 例如 vc_mapping = {0, 2, 3, 1}, ISP ch0 代表 vc=0; ch1 代表 vc=2; ch2 代表 vc=3; ch3 代表 vc=1

【注意事项】

MIPI-CSI 使用 Virtual Channel 解多任务时，支持的解多任务数量 MIPI_DEMUX_NUM 为 4。

【相关数据类型及接口】

mipi_dev_attr_s

3.3.13 lvds_dev_attr_s

【说明】

LVDS/SubLVDS 设备属性结构体。

【定义】

```
struct lvds_dev_attr_s {  
  
    enum wdr_mode_e wdr_mode;  
  
    enum lvds_sync_mode_e sync_mode;
```

(下页继续)

(续上页)

```

enum raw_data_type_e raw_data_type;

enum lvds_bit_endian data_endian;

enum lvds_bit_endian sync_code_endian;

short lane_id[MIPI_LANE_NUM+1];

short sync_code[MIPI_LANE_NUM][WDR_VC_NUM+1][SYNC_CODE_NUM];

struct lvds_vsync_type_s vsync_type;

struct lvds_fid_type_s fid_type;

char pn_swap[MIPI_LANE_NUM+1];
};

```

【成员】

成员名称	描述
wdr_mode	WDR 模式枚举
sync_mode	LVDS 同步模式
raw_data_type	传输图像数据类型
data_endian	数据大小端模式
sync_code_endian	同步信号大小端
lane_id	发送端 (sensor) 和接收端 (MIPI Rx) lane 的对应关系
sync_code	每个 Virtual Channel 有 4 个同步码, 根据同步模式不同, 分别表示 SOF/EOF/SOL/EOL 的同步码或者无效 AV/无效 EAV/有效 SAV/有效 EAV 的同步码
vsync_type	evsync 类型, 当 wdr_mode 为 DOL 模式并且 sync_mode 为 LVDS_SYNC_MODE_SAV 时, 需要配置 vsync 的类型
fid_type	frame identification 类型
pn_swap	P/N 差分通道是否交换

【注意事项】

支持的最大通道数 MIPI_LANE_NUM 为 8。用于宽动态范围 (WDR) 模式时, 虚拟通道数量 WDR_VC_NUM 为 2。支持的同步码数量 SYNC_CODE_NUM 为 4。

【相关数据类型及接口】

combo_dev_attr_s, wdr_mode_e, lvds_sync_mode_e, raw_data_type_e, lvds_bit_endian, lvds_vsync_type_s, lvds_fid_type_s

3.3.14 lvds_sync_mode_e

【说明】

LVDS 同步模式枚举。

【定义】

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

【成员】

成员名称	描述
LVDS_SYNC_MODE_SOF	Start of Frame 帧开头
LVDS_SYNC_MODE_SAV	有效数据起始同步码
LVDS_SYNC_MODE_BUTT	空，代表结尾

【注意事项】

无

【相关数据类型及接口】

lvds_dev_attr_s

3.3.15 lvds_bit_endian

【说明】

lvds 协议同步码大小端。

【定义】

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

【成员】

成员名称	描述
LVDS_ENDIAN_LITTLE	小端模式
LVDS_ENDIAN_BIG	大端模式
LVDS_ENDIAN_BUTT	空，代表结尾

【注意事项】

无

【相关数据类型及接口】

lvds_dev_attr_s

3.3.16 lvds_vsync_type_s

【说明】

LVDS WDR 同步参数。

【定义】

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

【成员】

成员名称	描述
sync_type	同步类型
hblank1	填充
hblank2	填充

【注意事项】

· 当 sync_type 为 LVDS_VSYNC_HCONNECT 时，需要配置 hblank1 和 hblank2，表示长短曝的填充长度。

【相关数据类型及接口】

lvds_dev_attr_s

3.3.17 lvds_vsync_type_e

【说明】

LVDS 在 WDR 模式的同步方式。

【定义】

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

【成员】

成员名称	描述
LVDS_VSYNC_NORMAL	长短曝光帧有独立的 SOF-EOF, SOL-EOL 或 invalid-SAVinvalid-EAV, valid SAV-valid EAV
LVDS_VSYNC_SHARE	长短曝光帧共享一对 SOF-EOF 标识, 短曝光的起始几行用固定值填充
LVDS_VSYNC_HCONNECT	长短曝光帧共用一对 SAV-EAV 标识, 长短曝光帧之间是固定周期的消隐
LVDS_VSYNC_BUTT	空, 代表结尾

【注意事项】

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

【相关数据类型及接口】

lvds_dev_attr_s

3.3.18 ttl_dev_attr_s

【说明】

TTL/BT 接口的属性设定结构体。

【定义】

```
struct ttl_dev_attr_s {
```

(下页继续)

(续上页)

```
enum ttl_src_e VI;

enum ttl_fmt_e ttl_fmt;

enum raw_data_type_e raw_data_type;

signed char func[TTL_PIN_FUNC_NUM];

enum ttl_VIOclk_e VIO_clk;

unsigned short v_bp;

unsigned short h_bp;

};
```

【成员】

成员名称	描述
VI	VI 通道序号
ttl_fmt	ttl 图像格式
raw_data_type	图像数据类型
func	BT 接口对应输入源的 lane number. Index 为 BT 逻辑功能
VIO_clk	VI 时钟
v_bp	垂直后肩
h_bp	水平后肩

【注意事项】

TTL_PIN_FUNC_NUM 为 TTL 引脚功能总数。

【相关数据类型及接口】

combo_dev_attr_s

3.3.19 ttl_src_e

【说明】

TTL/BT 数据源枚举。

【定义】

```
enum ttl_src_e {

    TTL_VI_SRC_VIO = 0,

    TTL_VI_SRC_VI1,

    TTL_VI_SRC_VI2,
```

(下页继续)

(续上页)

```
TTL_VI_SRC_NUM  
};
```

【成员】

成员名称	描述
TTL_VI_SRC_VI0	序号 0 通道
TTL_VI_SRC_VI1	序号 1 通道
TTL_VI_SRC_VI2	序号 2 通道
TTL_VI_SRC_NUM	ttl 数据源数量

【注意事项】

无

【相关数据类型及接口】

ttl_dev_attr_s

3.3.20 ttl_fmt_e

【说明】

TTL/BT 同步码方式枚举。

【定义】

```
enum ttl_fmt_e {  
    TTL_SYNC_PAT = 0,  
    TTL_VHS_11B,  
    TTL_VHS_19B,  
    TTL_VDE_11B,  
    TTL_VDE_19B,  
    TTL_VSDE_11B,  
    TTL_VSDE_19B,  
};
```

【成员】

成员名称	描述
TTL_SYNC_PAT	同步码方式同步
TTL_VHS_11B	vsync 和 hsync 同步 8lane 模式
TTL_VHS_19B	vsync 和 hsync 同步 16lane 模式
TTL_VDE_11B	Vertical data enable 和 horizon dataenable 同步方式。8lane 模式
TTL_VDE_19B	Vertical data enable 和 horizon dataenable 同步方式。16lane 模式
TTL_VSDE_11B	Vertical data enable 和 horizon dataenable 以及 data enable 同步方式。8lane 模式
TTL_VSDE_19B	Vertical data enable 和 horizon dataenable 以及 data enable 同步方式。16lane 模式

【注意事项】

无

【相关数据类型及接口】

ttl_dev_attr_s

3.3.21 ttl_VI0clk_e

【说明】

TTL/BT 的时钟枚举。

【定义】

```
enum ttl_VI0clk_e {
    TTL_VI0_CLK0 = 0,
    TTL_VI0_CLK1,
    TTL_VI0_CLK_MAX
};
```

【成员】

成员名称	描述
TTL_VI0_CLK0	时钟序号 0
TTL_VI0_CLK1	时钟序号 1
TTL_VI0_CLK_MAX	结尾

【注意事项】

无

【相关数据类型及接口】 ttl_dev_attr_s

3.3.22 bt_demux_attr_s

【说明】

BT 解多任务模式的属性设定结构体。

【定义】

```
struct bt_demux_attr_s {
    signed char func[TTL_PIN_FUNC_NUM];

    unsigned short v_fp;

    unsigned short h_fp;

    unsigned short v_bp;

    unsigned short h_bp;

    enum bt_demux_mode_e mode;

    unsigned char sync_code_part_A[3];

    struct bt_demux_sync_s sync_code_part_B[BT_DEMUX_NUM];

    char yc_exchg;
};
```

【成员】

成员名称	描述
func	BT 接口对应输入源 TTL_VI_SRC_VI2 的 lane number. Index 为 BT 逻辑功能
v_fp	垂直 fornt proch 每帧同步码后面。vsync 以后的无效行数，
h_fp	水平 front proch 每行后面。hsync 以后的无效 clock 数
v_bp	垂直 back proch 每帧同步码前面。vsync 以后的无效行数
h_bp	垂直 back proch 每行前面。Hsync 以后，同步码前的无效 clock 数
mode	并口模式枚举
sync_code_part_A	解复用模式下 BT 0 ~ 2 之间的同步码
sync_code_part_B	BT 解复用模式的同步码 3
yc_exchg	Bt 输入图像的 y c 顺序交换

【注意事项】

TTL_PIN_FUNC_NUM 为 TLL 引脚功能总数。

BT 解多任务模式下，解复用器的数量 BT_DEMUX_NUM 为 4。图为具体的 timing，在 vhs 模式下系统必须设定，v_bp 和 h_bp 以及图像的宽高

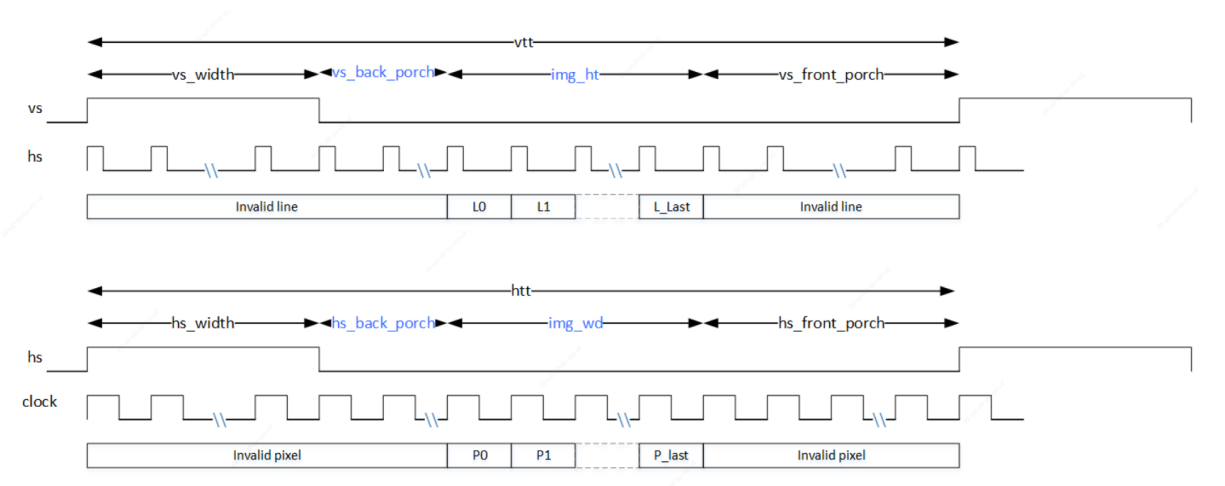


图 3.5: vhs mode timing

【相关数据类型及接口】

combo_dev_attr_s

3.3.23 bt_demux_mode_e

【说明】

BT 解多任务模式的信道数量枚举。

【定义】

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

【成员】

成员名称	描述
BT_DEMUX_DISABLE	默认关闭
BT_DEMUX_2	复用 2 通道
BT_DEMUX_3	复用 3 通道
BT_DEMUX_4	复用 4 通道

【注意事项】

无

【相关数据类型及接口】

bt_demux_attr_s

3.3.24 bt_demux_sync_s

【说明】

BT 解多任务模式的同步码结构体。

【定义】

```
struct bt_demux_sync_s {  
  
    unsigned char sav_vld;  
  
    unsigned char sav_blk;  
  
    unsigned char eav_vld;  
  
    unsigned char eav_blk;  
  
};
```

【成员】

成员名称	描述
sav_vld	有效行区，行同步信号结束，有效图像数据开始
sav_blk	消隐行区，行同步信号结束。(本行无有效图像的数据)
eav_vld	有效行区，行同步信号开始，有效图像数据结束
eav_blk	消隐行区，行同步信号开始。(本行无有效图像的数据)

【注意事项】

无

【相关数据类型及接口】

bt_demux_attr_s

3.3.25 img_size_s

【说明】

MIPI Rx 输入数据每帧的大小结构体。

【定义】

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

(下页继续)

(续上页)

```
unsigned int height;

};
```

【成员】

成员名称	描述
width	宽度
height	高度

【注意事项】

无

【相关数据类型及接口】 combo_dev_attr_s

3.3.26 manual_wdr_attr_s

【说明】

手动宽动态范围属性设置结构体。

【定义】

```
struct manual_wdr_attr_s {

    unsigned int manual_en;

    unsigned short l2s_distance;

    unsigned short lsef_length;

    unsigned int discard_padding_lines;

    unsigned int update;

};
```

【成员】

成员名称	描述
manual_en	使能
l2s_distance	长短曝时间间隔
lsef_length	长短帧曝光时间
discard_padding_lines	丢弃的填充行数量
update	更新标识符

【注意事项】

无

【相关数据类型及接口】

combo_dev_attr_s

3.3.27 clk_edge_s

【说明】

设备时钟电平信息结构体。

【定义】

```
struct clk_edge_s {  
  
    unsigned int devno;  
  
    enum clk_edge_e edge;  
  
};
```

【成员】

成员名称	描述
devno	设备号
edge	时钟电平枚举

【注意事项】

无

【相关数据类型及接口】

clk_edge_e, CVI_MIPI_SET_OUTPUT_CLK_EDGE

3.3.28 clk_edge_e

【说明】

电平采样枚举。

【定义】

```
enum clk_edge_e {  
  
    CLK_UP_EDGE = 0,  
  
    CLK_DOWN_EDGE,  
  
    CLK_EDGE_BUTT  
  
};
```

【成员】

成员名称	描述
CLK_UP_EDGE	上升沿电平采样
CLK_DOWN_EDGE	下降沿电平采样
CLK_EDGE_BUTT	空，代表结尾

【注意事项】

无

【相关数据类型及接口】

clk_edge_s

3.3.29 sns_rst_config

【说明】

传感器复位配置结构体。

【定义】

```
typedef struct sns_rst_config {  
  
    unsigned int devno;  
  
    unsigned int gpio_pin;  
  
    enum sns_rst_active_e gpio_active;  
  
} SNS_RST_CONFIG;
```

【成员】

成员名称	描述
devno	设备号
gpio_pin	GPIO 引脚号
gpio_active	GPIO 电平枚举

【注意事项】

无

【相关数据类型及接口】

sns_rst_active_e, CVI_MIPI_RESET_SENSOR, CVI_MIPI_UNRESET_SENSOR

3.3.30 sns_rst_active_e

【说明】

传感器复位电平枚举。

【定义】

```
enum sns_rst_active_e {  
  
    RST_ACTIVE_LOW,  
  
    RST_ACTIVE_HIGH,  
  
    RST_ACTIVE_BUFF,  
  
};
```

【成员】

成员名称	描述
RST_ACTIVE_LOW	低电平复位
RST_ACTIVE_HIGH	高电平复位
RST_ACTIVE_BUFF	缓冲复位

【注意事项】

无

【相关数据类型及接口】

sns_rst_config

3.3.31 crop_top_s

【说明】

数据开头裁剪信息结构体。

【定义】

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

【成员】

成员名称	描述
devno	MIPI-Rx 设备号
crop_top	裁剪个数
update	是否强制更新设定。若否, 设定会等下一张的同步讯号来才更新

【注意事项】

无

【相关数据类型及接口】

CVI_MIPI_SET_CROP_TOP

3.3.32 manual_wdr_s

【说明】

手动 WDR 模式设定结构体。

【定义】

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

【成员】

成员名称	描述
devno	MIPI-Rx 设备号
attr	手动 WDR 属性结构体

【注意事项】

无

【相关数据类型及接口】

manual_wdr_attr_s, CVI_MIPI_SET_WDR_MANUAL

3.3.33 vsync_gen_s

【说明】

LVDS 同步时间点设置结构体。

【定义】

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

【成员】

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

【注意事项】

无

【相关数据类型及接口】

CVI_MIPI_SET_LVDS_FP_VS

3.3.34 bt_fmt_out_s

【说明】

设备并口输出格式信息结构体。

【定义】

```
struct vsync_gen_s {  
  
    unsigned int devno;  
  
    enum bt_fmt_out_e fmt_out;  
  
};
```

【成员】

成员名称	描述
devno	MIPI-Rx 设备号
fmt_out	输出格式枚举

【注意事项】

无

【相关数据类型及接口】

bt_fmt_out_e, CVI_MIPI_SET_BT_FMT_OUT

3.3.35 bt_fmt_out_e

【说明】

输出格式枚举。

【定义】

```
enum bt_fmt_out_e {  
  
    BT_FMT_OUT_CBYCRY,  
  
    BT_FMT_OUT_CRYCBY,  
  
    BT_FMT_OUT_YCBYCR,  
  
    BT_FMT_OUT_YCRYCB,  
  
};
```

【成员】

成员名称	描述
BT_FMT_OUT_CBYCRY	输出 yuv 格式为 VYUY
BT_FMT_OUT_CRYCBY	输出 yuv 格式为 UYVY
BT_FMT_OUT_YCBYCR	输出 yuv 格式为 UYVY
BT_FMT_OUT_YCRYCB	输出 yuv 格式为 UYVY

【注意事项】

无

【相关数据类型及接口】

bt_fmt_out_s

3.3.36 cif_attr_s

【说明】

cif 同步信息属性结构体。

【定义】

```
struct cif_attr_s {  
  
    unsigned int devno;  
  
    unsigned int stagger_vsync;  
  
};
```

【成员】

成员名称	描述
devno	MIPI-Rx 设备号
stagger_vsync	交织 vsync

【注意事项】

无

【相关数据类型及接口】

CVI_MIPI_GET_CIF_ATTR

3.3.37 cif_crop_win_s

【说明】

裁剪窗口属性结构体。

【定义】

```
struct cif_crop_win_s {  
  
    unsigned int devno;  
  
    unsigned int enable;  
  
    unsigned int x;  
  
    unsigned int y;  
  
    unsigned int w;  
  
    unsigned int h;  
  
};
```

【成员】

成员名称	描述
devno	MIPI-Rx 设备号
enable	使能
x	X 坐标
y	Y 坐标
w	宽度
h	高度

【注意事项】

无

【相关数据类型及接口】

CVI_MIPI_SET_CROP_WINDOW

3.3.38 cif_yuv_swap_s

【说明】

yuv 图像不同分量交换结构体。

【定义】

```
struct cif_yuv_swap_s {  
    unsigned int devno;  
    unsigned int uv_swap;  
    unsigned int yc_swap;  
};
```

【成员】

成员名称	描述
devno	MIPI-Rx 设备号
uv_swap	色度分量交换
yc_swap	亮度与色度分量交换

【注意事项】

暂时无法使用 uv swap 。支持 yc swap.

【相关数据类型及接口】

CVI_MIPI_SET_YUV_SWAP

3.3.39 SNS_ATTR_S

【说明】

16 位传感器属性结构体。

【定义】

```
typedef struct _SNS_ATTR_S {  
    CVI_U16  u16Min;  
    CVI_U16  u16Max;  
    CVI_U16  u16Def;  
    CVI_U16  u16Step;  
} SNS_ATTR_S;
```

【成员】

成员名称	描述
u16Min	最小值
u16Max	最大值
u16Def	默认值
u16Step	步长

【注意事项】

无

【相关数据类型及接口】

无

3.3.40 SNS_ATTR_LARGE_S

【说明】

32 位传感器属性结构体。

【定义】

```
typedef struct _SNS_ATTR_LARGE_S {  
    CVI_U32  u32Min;  
    CVI_U32  u32Max;  
    CVI_U32  u32Def;  
    CVI_U32  u32Step;
```

(下页继续)

(续上页)

```
} SNS_ATTR_LARGE_S;
```

【成员】

成员名称	描述
u32Min	最小值
u32Max	最大值
u32Def	默认值
u32Step	步长

【注意事项】

无

【相关数据类型及接口】

无

3.3.41 ISP_SNS_STATE_S

【说明】

ISP 传感器状态结构-跟踪当前传感器状态

【定义】

```
typedef struct _ISP_SNS_STATE_S {
    CVI_BOOL    bInit;

    CVI_BOOL    bSyncInit;

    CVI_U8      u8ImgMode;

    CVI_U8      u8Hdr;

    WDR_MODE_E  enWDRMode;

    ISP_SNS_SYNC_INFO_S astSyncInfo[2];

    CVI_U32     au32FL[2];

    CVI_U32     u32FLStd;

    CVI_U32     au32WDRIntTime[4];
} ISP_SNS_STATE_S;
```

【成员】

成员名称	描述
bInit	传感器初始化状态
bSyncInit	同步寄存器初始化状态
u8ImgMode	图像模式
u8Hdr	宽动态范围模式
enWDRMode	宽动态范围模式
astSyncInfo	同步信息
au32FL	当前帧和前一帧的全线数
u32FLStd	标准帧线数
au32WDRIntTime	宽动态范围积分时间

【注意事项】

无

【相关数据类型及接口】

无

3.3.42 ISP_SNS_MIRRORFLIP_TYPE_E

【说明】

传感器镜像翻转类型枚举。

【定义】

```
typedef enum _ISP_SNS_MIRRORFLIP_TYPE_E {  
    ISP_SNS_NORMAL      = 0,  
    ISP_SNS_MIRROR      = 1,  
    ISP_SNS_FLIP        = 2,  
    ISP_SNS_MIRROR_FLIP = 3,  
    ISP_SNS_BUTT  
} ISP_SNS_MIRRORFLIP_TYPE_E;
```

【成员】

成员名称	描述
ISP_SNS_NORMAL	正常模式
ISP_SNS_MIRROR	镜像模式
ISP_SNS_FLIP	翻转模式
ISP_SNS_MIRROR_FLIP	镜像翻转模式
ISP_SNS_BUTT	无效模式

【注意事项】

无

【相关数据类型及接口】

无

3.3.43 ISP_SNS_INTTIME_MODE_E

【说明】

传感器积分时间模式枚举。

【定义】

```
typedef enum _ISP_SNS_L2S_MODE_E {  
    SNS_L2S_MODE_AUTO = 0,  
    SNS_L2S_MODE_FIX,  
} ISP_SNS_INTTIME_MODE_E;
```

【成员】

成员名称	描述
SNS_L2S_MODE_AUTO	自动模式
SNS_L2S_MODE_FIX	固定模式

【注意事项】

无

【相关数据类型及接口】

无

3.3.44 MCLK_ATTR_S

【说明】

时钟属性结构体。

【定义】

```
typedef struct _MCLK_ATTR_S {  
    CVI_U8 u8Mclk;  
    CVI_BOOL bMclkEn;  
} MCLK_ATTR_S;
```


【成员】

成员名称	描述
u8Mclk	时钟频率
bMclkEn	时钟使能

【注意事项】

无

【相关数据类型及接口】

无

3.3.45 RX_INIT_ATTR_S

【说明】

MIPI Rx 初始化属性结构体。

【定义】

```
typedef struct _RX_INIT_ATTR_S {  
    CVI_U32 MipiDev;  
    CVI_U32 MipiMode;  
    CVI_S16 as16LaneId[MIPI_LANE_NUM + 1];  
    CVI_S16 as16FuncId[TTL_PIN_FUNC_NUM];  
    CVI_S8 as8PNSwap[MIPI_LANE_NUM + 1];  
    MCLK_ATTR_S stMclkAttr;  
    CVI_BOOL hsettlen;  
    CVI_U8 hsettle;  
} RX_INIT_ATTR_S;
```

【成员】

成员名称	描述
MipiDev	MIPI 设备 ID
MipiMode	MIPI 模式
as16LaneId	MIPI 通道 ID
as16FuncId	TTL 引脚功能 ID
as8PNSwap	P/N Swap 配置
stMclkAttr	MCLK 属性
hsettlen	MIPI HS 稳定使能
hsettle	MIPI HS 稳定时间

【注意事项】

TTL_PIN_FUNC_NUM 为 TTL 引脚功能总数。

【相关数据类型及接口】

无

3.3.46 SNS_BDG_MUX_MODE_E

【说明】

传感器桥接模式枚举。

【定义】

```
typedef enum _SNS_BDG_MUX_MODE_E {  
  
    SNS_BDG_MUX_NONE = 0,  
  
    SNS_BDG_MUX_2,  
  
    SNS_BDG_MUX_3,  
  
    SNS_BDG_MUX_4,  
  
} SNS_BDG_MUX_MODE_E;
```

【成员】

成员名称	描述
SNS_BDG_MUX_NONE	传感器桥接 mux 禁用
SNS_BDG_MUX_2	传感器桥接 mux 2 输入
SNS_BDG_MUX_3	传感器桥接 mux 3 输入
SNS_BDG_MUX_4	传感器桥接 mux 4 输入

【注意事项】

无

【相关数据类型及接口】

无

3.3.47 ISP_INIT_ATTR_S

【说明】

ISP 初始化属性结构体。

【定义】

```
typedef struct _ISP_INIT_ATTR_S {  
  
    CVI_U32 u32ExpTime;  
  
    CVI_U32 u32AGain;  
  
    CVI_U32 u32DGain;  
  
    CVI_U32 u32ISPDGain;  
  
    CVI_U32 u32Exposure;  
  
    CVI_U32 u32LinesPer500ms;  
  
    CVI_U32 u32PirisFNO;  
  
    CVI_U16 u16WBRgain;  
  
    CVI_U16 u16WBGgain;  
  
    CVI_U16 u16WBBgain;  
  
    CVI_U16 u16SampleRgain;  
  
    CVI_U16 u16SampleBgain;  
  
    CVI_U16 u16UseHwSync;  
  
    ISP_SNS_GAIN_MODE_E enGainMode;  
  
    ISP_SNS_INTTIME_MODE_E enL2SMode;  
  
    SNS_BDG_MUX_MODE_E enSnsBdgMuxMode;  
  
} ISP_INIT_ATTR_S;
```

【成员】

成员名称	描述
u32ExpTime	曝光时间
u32AGain	模拟增益
u32DGain	数字增益
u32ISPDGain	ISP 数字增益
u32Exposure	整体曝光值
u32LinesPer500ms	每 500ms 的行数
u32PirisFNO	P-iris F-number
u16WBRgain	白平衡红色增益
u16WBGgain	白平衡绿色增益
u16WBBgain	白平衡蓝色增益
u16SampleRgain	采样红色增益
u16SampleBgain	采样蓝色增益
u16UseHwSync	硬件同步使用标志
enGainMode	增益模式
enL2SMode	积分时间模式
enSnsBdgMuxMode	传感器桥接 mux 模式

【注意事项】

无

【相关数据类型及接口】

无

3.3.48 ISP_SNS_OBJ_S

【说明】

ISP 传感器对象结构体。

【定义】

```
typedef struct _ISP_SNS_OBJ_S {
    CVI_S32 (*pfnRegisterCallback)(VI_PIPE ViPipe, ALG_LIB_S *, ALG_LIB_S *);
    CVI_S32 (*pfnUnRegisterCallback)(VI_PIPE ViPipe, ALG_LIB_S *, ALG_LIB_S *);
    CVI_S32 (*pfnSetBusInfo)(VI_PIPE ViPipe, ISP_SNS_COMMBUS_U unSNSBusInfo);
    CVI_VOID (*pfnStandby)(VI_PIPE ViPipe);
    CVI_VOID (*pfnRestart)(VI_PIPE ViPipe);
    CVI_VOID (*pfnMirrorFlip)(VI_PIPE ViPipe, ISP_SNS_MIRRORFLIP_TYPE_E_
↪eSnsMirrorFlip);
    CVI_S32 (*pfnWriteReg)(VI_PIPE ViPipe, CVI_S32 s32Addr, CVI_S32 s32Data);
}
```

(下页继续)

(续上页)

```

CVI_S32 (*pfnReadReg)(VI_PIPE ViPipe, CVI_S32 s32Addr);

CVI_S32 (*pfnSetInit)(VI_PIPE ViPipe, ISP_INIT_ATTR_S *);

CVI_S32 (*pfnPatchRxAttr)(VI_PIPE ViPipe, RX_INIT_ATTR_S *);

CVI_VOID (*pfnPatchI2cAddr)(VI_PIPE ViPipe, CVI_S32 s32I2cAddr);

CVI_S32 (*pfnGetRxAttr)(VI_PIPE ViPipe, SNS_COMBO_DEV_ATTR_S *);

CVI_S32 (*pfnExpSensorCb)(ISP_SENSOR_EXP_FUNC_S *);

CVI_S32 (*pfnExpAeCb)(AE_SENSOR_EXP_FUNC_S *);

CVI_S32 (*pfnSnsProbe)(VI_PIPE ViPipe);

} ISP_SNS_OBJ_S;

```

【成员】

成员名称	描述
pfnRegisterCallback	注册回调函数
pfnUnRegisterCallback	注销回调函数
pfnSetBusInfo	设置传感器总线信息
pfnStandby	设置传感器待机模式
pfnRestart	重启传感器
pfnMirrorFlip	设置镜像翻转模式
pfnWriteReg	写入传感器寄存器
pfnReadReg	读取传感器寄存器
pfnSetInit	设置传感器初始化参数
pfnPatchRxAttr	修补接收器属性
pfnPatchI2cAddr	更新 I2C 地址
pfnGetRxAttr	获取接收器属性
pfnExpSensorCb	曝光传感器回调
pfnExpAeCb	曝光 AE 回调
pfnSnsProbe	探测传感器

【注意事项】

无

【相关数据类型及接口】

无

3.3.49 MIPI_TX0_NAME

【说明】

MIPI TX 设备 0 的路径定义。

【定义】

```
#define MIPI_TX0_NAME "/dev/soph-mipi-tx0"
```

【注意事项】

无

【相关数据类型及接口】

无

3.3.50 MIPI_TX1_NAME

【说明】

MIPI TX 设备 1 的路径定义。

【定义】

```
#define MIPI_TX1_NAME "/dev/soph-mipi-tx1"
```

【注意事项】

无

【相关数据类型及接口】

无

3.3.51 CMD_MAX_NUM

【说明】

最大命令数量。

【定义】

```
#define CMD_MAX_NUM 128
```

【注意事项】

无

【相关数据类型及接口】

无

3.3.52 RX_MAX_NUM

【说明】

最大接收通道数。

【定义】

```
#define RX_MAX_NUM 4
```

【注意事项】

无

【相关数据类型及接口】

无

3.3.53 LANE_MAX_NUM

【说明】

最大通道数。

【定义】

```
#define LANE_MAX_NUM 5
```

【注意事项】

无

【相关数据类型及接口】

无

3.3.54 output_mode_e

【说明】

输出模式枚举类型。

【定义】

```
enum output_mode_e {  
    OUTPUT_MODE_CSI          = 0x0,  
    OUTPUT_MODE_DSI_VIDEO    = 0x1,  
    OUTPUT_MODE_DSI_CMD      = 0x2,  
    OUTPUT_MODE_BUTT  
};
```

【成员】

成员名称	描述
OUTPUT_MODE_CSI	CSI 模式
OUTPUT_MODE_DSI_VIDEO	DSI 视频模式
OUTPUT_MODE_DSI_CMD	DSI 命令模式
OUTPUT_MODE_BUTT	结束标志

【注意事项】

无

【相关数据类型及接口】

无

3.3.55 video_mode_e

【说明】

视频模式枚举类型。

【定义】

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

【成员】

成员名称	描述
BURST_MODE	突发模式
NON_BURST_MODE_SYNC_PULSES	非突发同步脉冲模式
NON_BURST_MODE_SYNC_EVENTS	非突发同步事件模式

【注意事项】

无

【相关数据类型及接口】

无

3.3.56 output_format_e

【说明】

输出格式枚举类型。

【定义】

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

【成员】

成员名称	描述
OUT_FORMAT_RGB_16_BIT	RGB 16 位格式
OUT_FORMAT_RGB_18_BIT	RGB 18 位格式
OUT_FORMAT_RGB_24_BIT	RGB 24 位格式
OUT_FORMAT_RGB_30_BIT	RGB 30 位格式
OUT_FORMAT_YUV420_8_BIT_NORMAL	YUV420 8 位标准格式
OUT_FORMAT_YUV420_8_BIT_LEGACY	YUV420 8 位传统格式
OUT_FORMAT_YUV422_8_BIT	YUV422 8 位格式
OUT_FORMAT_BUTT	结束标志

【注意事项】

无

【相关数据类型及接口】

无

3.3.57 mipi_tx_lane_id

【说明】

MIPI TX 通道 ID 枚举类型。

【定义】

```
enum mipi_tx_lane_id {  
    MIPI_TX_LANE_CLK = 0,  
    MIPI_TX_LANE_0,  
    MIPI_TX_LANE_1,  
    MIPI_TX_LANE_2,  
};
```

(下页继续)

(续上页)

```
MIPI_TX_LANE_3,
MIPI_TX_LANE_MAX
};
```

【成员】

成员名称	描述
MIPI_TX_LANE_CLK	时钟通道
MIPI_TX_LANE_0	数据通道 0
MIPI_TX_LANE_1	数据通道 1
MIPI_TX_LANE_2	数据通道 2
MIPI_TX_LANE_3	数据通道 3
MIPI_TX_LANE_MAX	最大通道数

【注意事项】

无

【相关数据类型及接口】

无

3.3.58 sync_info_s

【说明】

视频同步时序参数结构体。

【定义】

```
struct sync_info_s {
    unsigned short vid_hsa_pixels;
    unsigned short vid_hbp_pixels;
    unsigned short vid_hfp_pixels;
    unsigned short vid_hline_pixels;
    unsigned short vid_vsa_lines;
    unsigned short vid_vbp_lines;
    unsigned short vid_vfp_lines;
    unsigned short vid_active_lines;
    unsigned short edpi_cmd_size;
    bool vid_vsa_pos_polarity;
    bool vid_hsa_pos_polarity;
};
```

【成员】

成员名称	描述
vid_hsa_pixels	水平同步有效像素数
vid_hbp_pixels	水平后肩像素数
vid_hfp_pixels	水平前肩像素数
vid_hline_pixels	水平总像素数
vid_vsa_lines	垂直同步有效行数
vid_vbp_lines	垂直后肩行数
vid_vfp_lines	垂直前肩行数
vid_active_lines	有效显示行数
edpi_cmd_size	EDPI 命令大小
vid_vsa_pos_polarity	垂直同步极性
vid_hsa_pos_polarity	水平同步极性

【注意事项】

无

【相关数据类型及接口】

无

3.3.59 combo_dev_cfg_s

【说明】

MIPI TX 设备配置参数结构体。

【定义】

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

【成员】

成员名称	描述
devno	设备号
lane_id	通道 ID 数组
output_mode	输出模式
video_mode	视频模式
output_format	输出格式
sync_info	同步信息
phy_data_rate	物理层数据速率 (Mbps)
pixel_clk	像素时钟 (KHz)
lane_pn_swap	通道 PN 交换标志

【注意事项】

无

【相关数据类型及接口】

无

3.3.60 cmd_info_s

【说明】

命令信息结构体。

【定义】

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

【成员】

成员名称	描述
devno	设备号
data_type	数据类型
cmd_size	命令大小
cmd	命令数据指针

【注意事项】

无

【相关数据类型及接口】

无

3.3.61 get_cmd_info_s

【说明】

获取命令信息结构体。

【定义】

```
struct get_cmd_info_s {  
    unsigned int devno;  
    unsigned short data_type;  
    unsigned short data_param;  
    unsigned short get_data_size;  
    unsigned char *get_data;  
};
```

【成员】

成员名称	描述
devno	设备号
data_type	数据类型
data_param	数据参数
get_data_size	读取数据大小
get_data	读取数据内存地址

【注意事项】

无

【相关数据类型及接口】

无

3.3.62 hs_settle_s

【说明】

高速时序配置结构体。

【定义】

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

【成员】

成员名称	描述
prepare	准备时间
zero	零时间
trail	拖尾时间

【注意事项】

无

【相关数据类型及接口】

无

3.3.63 dsc_instr

【说明】

DSC 指令结构体。

【定义】

```
struct dsc_instr {  
    unsigned char delay;  
    unsigned char data_type;  
    unsigned char size;  
    unsigned char *data;  
};
```

【成员】

成员名称	描述
delay	延迟时间
data_type	数据类型
size	数据大小
data	数据指针

【注意事项】

无

【相关数据类型及接口】

无

3.4 用户函数

- SAMPLE_COMM_VI_ParseIni：解析 ini 文本。
- SAMPLE_COMM_VI_IniToVICfg：使用解析的信息对 VI 使用的结构体进行配置。
- SAMPLE_COMM_VI_GetSensorInfo：配置 sensor 的默认信息。
- SAMPLE_COMM_VI_GetSizeBySensor：通过 sensor 的类型获得图片的大小。
- SAMPLE_COMM_VI_StartMIPI：初始 MIPI 的时钟设置 MIPI 属性。
- SAMPLE_COMM_VI_SensorProbe：检测 SENSOR 是否初始化正常。
- SAMPLE_COMM_VI_ResetSensor：重置 SENSOR 配置。

- `SAMPLE_COMM_VI_ResetMipi` : 重置 MIPI 配置。
- `SAMPLE_COMM_VI_UnresetSensor` : 取消重置 SENSOR 配置。
- `SAMPLE_COMM_VI_UnresetMipi` : 取消重置 MIPI 配置。
- `SAMPLE_COMM_VI_SetMipiAttr` : 设置 MIPI 属性。
- `SAMPLE_COMM_VI_EnableSensorClock` : 使能 SENSOR 时钟。
- `SAMPLE_COMM_VI_SuspendSensor` : 悬挂 SENSOR 时钟。
- `SAMPLE_COMM_VI_ResumeSensor` : 恢复 SENSOR 时钟。
- `SAMPLE_COMM_VI_SuspendMipi` : 悬挂 MIPI 通道。
- `SAMPLE_COMM_VI_ResumeMipi` : 恢复 MIPI 通道。
- `SAMPLE_COMM_VI_StartSensor` : 启动 SENSOR。
- `SAMPLE_COMM_SYS_GetPicSize` : 获得图片的大小, 传递给 `stSize`。
- `SAMPLE_PLAT_SYS_INIT` : 相关内存 (VBPool) 初始化。
- `SAMPLE_PLAT_VI_INIT` : VI 的 platform 初始化
- `SAMPLE_COMM_ISP_SetSnsObj` : 根据 ini 设置 SENSOR 的类型。
- `SAMPLE_COMM_ISP_GetSnsObj` : 通过 ini 设置的 SENSOR type 获得 sensor 对象。
- `SAMPLE_COMM_ISP_PatchSnsObj` : 根据 ini 文件的设定, 修改 SENSOR 的默认设定。
- `SAMPLE_COMM_ISP_Sensor_Register_callback` : SENSOR AE 回调, AWB 曝光, AE 曝光。
- `SAMPLE_COMM_ISP_SetSensorMode` : 设置 SENSOR 的工作模式, WDR 或者 Line 模式, 并且设置 WDR 的属性。

3.4.1 SAMPLE_COMM_VI_ParseIni

【描述】

解析 ini 文本。

【定义】

```
CVI_S32 SAMPLE_COMM_VI_ParseIni(SAMPLE_INI_CFG_S *pstIniCfg)
```

【参数】

参数名称	描述	输入/输出
<code>pstIniCfg</code>	保存 Ini 信息配置的结构体指针。	输入

【返回值】

返回值	描述
0	成功
非 0	失败, 请参考错误码

【需求】

- 头文件: sample_comm.h

【注意】

返回 0 要关注打印出来的信息，不一定代表成功。

【举例】

无

【相关主题】

无

3.4.2 SAMPLE_COMM_VI_IniToVICfg

【描述】

使用解析的信息对 VI 使用的结构体进行配置。

【定义】

```
CVI_S32 SAMPLE_COMM_VI_IniToVICfg(SAMPLE_INI_CFG_S *pstIniCfg, SAMPLE_VI_
→CONFIG_S *pstVICfg)
```

【参数】

参数名称	描述	输入/输出
pstIniCfg	保存 Ini 信息配置的结构体指针。	输入
pstVICfg	配置 VI 的结构体指针。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件: sample_comm.h

【注意】

需要先调用 SAMPLE_COMM_VI_ParseIni。

【举例】

无

【相关主题】

无

3.4.3 SAMPLE_COMM_VI_GetSensorInfo

【描述】

配置 sensor 的信息。

【定义】

```
void SAMPLE_COMM_VI_GetSensorInfo(SAMPLE_VI_CONFIG_S *pstVConfig)
```

【参数】

参数名称	描述	输入/输出
pstVConfig	保存 VI 配置信息的结构体指针。	输入

【返回值】

无

【需求】

- 头文件: sample_comm.h

【注意】

需要先初始化 astVInfo 结构体。

【举例】

无

【相关主题】

无

3.4.4 SAMPLE_COMM_VI_GetSizeBySensor

【描述】

通过 sensor 的类型获得图片的大小。

【定义】

```
CVI_S32 SAMPLE_COMM_VI_GetSizeBySensor(CVI_SNS_TYPE_E enMode, PIC_SIZE_E_  
→*penSize)
```

【参数】

参数名称	描述	输入/输出
enMode	模式种类枚举。	输入
penSize	图像大小枚举指针。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

无

3.4.5 SAMPLE_COMM_VI_StartMIPI

【描述】

初始 mipi 的时钟 clock，设置 mipi 属性。

【定义】

```
CVI_S32 SAMPLE_COMM_VI_StartMIPI(SAMPLE_VI_CONFIG_S *pstVConfig)
```

【参数】

参数名称	描述	输入/输出
pstVConfig	VI 配置信息结构体。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

无

3.4.6 SAMPLE_COMM_VI_SensorProbe

【描述】

检测 sensor 是否初始化正常。

【定义】

```
CVI_S32 SAMPLE_COMM_VI_SensorProbe(SAMPLE_VI_CONFIG_S *pstVConfig)
```

【参数】

参数名称	描述	输入/输出
pstVConfig	VI 配置信息结构体。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

无

3.4.7 SAMPLE_COMM_VI_ResetSensor

【描述】

重置 sensor 配置。

【定义】

```
CVI_S32 SAMPLE_COMM_VI_ResetSensor(SAMPLE_VI_CONFIG_S *pstVConfig)
```

【参数】

参数名称	描述	输入/输出
pstVConfig	VI 配置信息结构体。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

· 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

CVI_MIPI_SetSensorReset

3.4.8 SAMPLE_COMM_VI_ResetMipi

【描述】

重置 mipi 配置。

【定义】

```
CVI_S32 SAMPLE_COMM_VI_ResetMipi(SAMPLE_VI_CONFIG_S *pstVConfig)
```

【参数】

参数名称	描述	输入/输出
pstVConfig	VI 配置信息结构体。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

· 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

CVI_MIPI_SetMipiReset

3.4.9 SAMPLE_COMM_VI_UnresetSensor

【描述】

取消重置 sensor 配置。

【定义】

```
CVI_S32 SAMPLE_COMM_VI_UnresetSensor(SAMPLE_VI_CONFIG_S *pstVConfig)
```

【参数】

参数名称	描述	输入/输出
pstVConfig	VI 配置信息结构体。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

CVI_MIPI_SetSensorReset

3.4.10 SAMPLE_COMM_VI_UnresetMipi

【描述】

取消重置 MIPI 配置。

【定义】

```
CVI_S32 SAMPLE_COMM_VI_UnresetMipi(SAMPLE_VI_CONFIG_S *pstVConfig)
```

【参数】

参数名称	描述	输入/输出
pstVConfig	VI 配置信息结构体。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

CVI_MIPI_SetMipiReset

3.4.11 SAMPLE_COMM_VI_SetMipiAttr

【描述】

设置 MIPI 属性。

【定义】

```
CVI_S32 SAMPLE_COMM_VI_SetMipiAttr(SAMPLE_VI_CONFIG_S *pstViConfig)
```

【参数】

参数名称	描述	输入/输出
pstViConfig	VI 配置信息结构体。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

CVI_MIPI_SetClkEdge、CVI_MIPI_SetMipiAttr、SAMPLE_COMM_ISP_GetSnsObj

3.4.12 SAMPLE_COMM_VI_EnableSensorClock

【描述】

使能 SENSOR 时钟。

【定义】

```
CVI_S32 SAMPLE_COMM_VI_SetMipiAttr(SAMPLE_VI_CONFIG_S *pstViConfig)
```

【参数】

参数名称	描述	输入/输出
pstViConfig	VI 配置信息结构体。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

CVI_MIPI_SetSensorClock

3.4.13 SAMPLE_COMM_VI_SuspendSensor

【描述】

悬挂注册的 SENSOR。

【定义】

```
static CVI_S32 SAMPLE_COMM_VI_SuspendSensor(CVI_VOID *pvData)
```

【参数】

参数名称	描述	输入/输出
pvData	包含 VI 通道、传感器节点等信息的结构体指针。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

SAMPLE_COMM_ISP_GetSnsObj

3.4.14 SAMPLE_COMM_VI_ResumeSensor

【描述】

恢复 SENSOR 时钟。

【定义】

```
static CVI_S32 SAMPLE_COMM_VI_ResumeSensor(CVI_VOID *pvData)
```

【参数】

参数名称	描述	输入/输出
pvData	包含 VI 通道、传感器节点等信息的结构体指针。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

SAMPLE_COMM_ISP_GetSnsObj

3.4.15 SAMPLE_COMM_VI_SuspendMipi

【描述】

悬挂 MIPI 通道。

【定义】

```
static CVI_S32 SAMPLE_COMM_VI_SuspendMipi(CVI_VOID *pvData)
```

【参数】

参数名称	描述	输入/输出
pvData	包含 VI 通道、传感器节点等信息的结构体指针。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

CVI_MIPI_SetSensorClock

3.4.16 SAMPLE_COMM_VI_ResumeMipi

【描述】

恢复 MIPI 通道。

【定义】

```
static CVI_S32 SAMPLE_COMM_VI_ResumeMipi(CVI_VOID *pvData)
```

【参数】

参数名称	描述	输入/输出
pvData	包含 VI 通道、传感器节点等信息的结构体指针。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

SAMPLE_COMM_ISP_GetSnsObj、CVI_MIPI_SetSensorClock、CVI_MIPI_SetMipiReset、CVI_MIPI_SetClkEdge、CVI_MIPI_SetMipiAttr、CVI_MIPI_SetSensorReset

3.4.17 SAMPLE_COMM_VI_StartSensor

【描述】

启动 SENSOR。

【定义】

```
CVI_S32 SAMPLE_COMM_VI_StartSensor(SAMPLE_VI_CONFIG_S *pstVConfig)
```

【参数】

参数名称	描述	输入/输出
pstVConfig	VI 配置信息结构体。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

SAMPLE_COMM_ISP_SetSensorMode

3.4.18 SAMPLE_COMM_SYS_GetPicSize

【描述】

获得图片的大小，传递给 stSize。

【定义】

```
CVI_S32 SAMPLE_COMM_SYS_GetPicSize(PIC_SIZE_E enPicSize, SIZE_S *pstSize)
```

【参数】

参数名称	描述	输入/输出
penSize	图像大小种类枚举。	输入
pstSize	图像大小结构体指针。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

无

3.4.19 SAMPLE_PLAT_SYS_INIT

【描述】

相关内存初始化。

【定义】

```
CVI_S32 SAMPLE_PLAT_SYS_INIT(SIZE_S stSize)
```

【参数】

参数名称	描述	输入/输出
stSize	图像大小结构体。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

无

3.4.20 SAMPLE_PLAT_VI_INIT

【描述】

VI 的 platform 初始化。

【定义】

```
CVI_S32 SAMPLE_PLAT_VI_INIT(SAMPLE_VI_CONFIG_S *pstVConfig)
```

【参数】

参数名称	描述	输入/输出
pstVConfig	保存 VI 配置信息的结构体指针。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

无

3.4.21 SAMPLE_COMM_ISP_SetSnsObj

【描述】

根据 ini 设置 SENSOR 的类型。

【定义】

```
CVI_S32 SAMPLE_COMM_ISP_SetSnsObj(CVI_U32 u32SnsId, CVI_SNS_TYPE_E enSnsType)
```

【参数】

参数名称	描述	输入/输出
u32SnsId	传感器序号。	输入
enSnsType	传感器型号枚举。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

无

3.4.22 SAMPLE_COMM_ISP_GetSnsObj

【描述】

通过 ini 设置的 SENSOR type 获得 SENSOR 对象。

【定义】

```
CVI_VOID *SAMPLE_COMM_ISP_GetSnsObj(CVI_U32 u32SnsId)
```

【参数】

参数名称	描述	输入/输出
u32SnsId	传感器序号。	输入

【返回值】

返回值	描述
CVI_VOID *	Sensor 对象指针。

【需求】

- 头文件: sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

无

3.4.23 SAMPLE_COMM_ISP_PatchSnsObj

【描述】

根据 ini 文件的设定，修改 SNESOR 的默认设定。

【定义】

```
CVI_S32 SAMPLE_COMM_ISP_PatchSnsObj(CVI_U32 u32SnsId, SAMPLE_SENSOR_INFO_S_  
→*pstSnsInfo)
```

【参数】

参数名称	描述	输入/输出
u32SnsId	传感器序号。	输入
pstSnsInfo	传感器信息结构体指针。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

无

3.4.24 SAMPLE_COMM_ISP_Sensor_Register_callback

【描述】

SENSOR AE 回调，AWB 曝光，AE 曝光。

【定义】

```
CVI_S32 SAMPLE_COMM_ISP_Sensor_Register_callback(ISP_DEV IspDev, CVI_U32 u32SnsId,  
→CVI_S32 s32BusId, CVI_S32 s32I2cAddr)
```

【参数】

参数名称	描述	输入/输出
IspDev	ISP 设备结构体。	输入
u32SnsId	传感器序号。	输入
s32BusId	总线序号。	输入
s32I2cAddr	I2C 地址。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

无

3.4.25 SAMPLE_COMM_ISP_SetSensorMode

【描述】

设置 SENSOR 的工作模式，这些模式包括 SENSOR 输出图像的宽高，是否为 WDR。

【定义】

```
CVI_S32 SAMPLE_COMM_ISP_SetSensorMode(SAMPLE_VI_CONFIG_S *pstVIFConfig)
```

【参数】

参数名称	描述	输入/输出
pstVIFConfig	VI 配置信息结构体。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：sample_comm.h

【注意】

无。

【举例】

无

【相关主题】

无

3.5 MIPI 函数：

- `mipi_open_dev`：开启设备的 MIPI 协议。
- `CVI_MIPI_SetMipiReset`：复位 MIPI 通道。
- `CVI_MIPI_SetSensorClock`：设置 MIPI 协议时钟。
- `CVI_MIPI_SetSensorReset`：复位 SENSOR。
- `CVI_MIPI_SetMipiAttr`：设置 MIPI 协议属性。
- `CVI_MIPI_SetClkEdge`：设置 MIPI 电平触发。
- `CVI_MIPI_SetSnsMclk`：设置 SENSOR 时钟。

3.5.1 `mipi_open_dev`

【描述】

开启设备的 MIPI 协议。

【定义】

```
CVI_S32 mipi_open_dev(CVI_VOID)
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：CVI_mipi.h

【注意】

必须先开启 MIPI 协议，才能调用 MIPI 相关的 CVI 函数。

【举例】

无

【相关主题】

`CVI_MIPI_SetMipiReset`、`CVI_MIPI_SetSensorClock`、`CVI_MIPI_SetSensorReset`、`CVI_MIPI_SetMipiAttr`、`CVI_MIPI_SetClkEdge`、`CVI_MIPI_SetSnsMclk`

3.5.2 CVI_MIPI_SetMipiReset

【描述】

复位 MIPI 通道。

【定义】

```
CVI_S32 CVI_MIPI_SetMipiReset(CVI_S32 devno, CVI_U32 reset)
```

【参数】

参数名称	描述	输入/输出
devno	设备号。	输入
reset	复位标识符。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：CVI_mipi.h

【注意】

无。

【举例】

无

【相关主题】

无

3.5.3 CVI_MIPI_SetSensorClock

【描述】

设置 MIPI 协议时钟。

【定义】

```
CVI_S32 CVI_MIPI_SetSensorClock(CVI_S32 devno, CVI_U32 enable)
```

【参数】

参数名称	描述	输入/输出
devno	设备号。	输入
enable	使能标识符。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：CVI_mipi.h

【注意】

无。

【举例】

无

【相关主题】

无

3.5.4 CVI_MIPI_SetSensorReset

【描述】

复位 SENSOR。

【定义】

```
CVI_S32 CVI_MIPI_SetSensorReset(SNS_RST_CONFIG *pstSnsrstInfo, CVI_U32 reset)
```

【参数】

参数名称	描述	输入/输出
pstSnsrstInfo	传感器复位配置。	输入
reset	复位标识符。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：CVI_mipi.h

【注意】

无。

【举例】

无

【相关主题】

无

3.5.5 CVI_MIPi_SetMipiAttr

【描述】

设置 MIPI 协议属性。

【定义】

```
CVI_S32 CVI_MIPi_SetMipiAttr(CVI_S32 VIPipe, const CVI_VOID *devAttr)
```

【参数】

参数名称	描述	输入/输出
VIPipe	VI 通道。	输入
devAttr	设备属性指针。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件：CVI_mipi.h

【注意】

无。

【举例】

无

【相关主题】

无

3.5.6 CVI_MIPI_SetClkEdge

【描述】

设置 MIPI 电平触发。

【定义】

```
CVI_S32 CVI_MIPI_SetClkEdge(CVI_S32 devno, CVI_U32 is_up)
```

【参数】

参数名称	描述	输入/输出
devno	设备号。	输入
is_up	是否上升沿触发。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

· 头文件：CVI_mipi.h

【注意】

无。

【举例】

无

【相关主题】

无

3.5.7 CVI_MIPI_SetSnsMclk

【描述】

设置 SENSOR 时钟。

【定义】

```
CVI_S32 CVI_MIPI_SetSnsMclk(SNS_MCLK_S *mclk)
```

【参数】

参数名称	描述	输入/输出
mclk	Sensor 时钟配置信息结构体指针。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，请参考错误码

【需求】

- 头文件: CVI_mipi.h

【注意】

无。

【举例】

无

【相关主题】

无

3.6 Proc 信息

3.6.1 MIPI_RX Proc 信息

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

【调试信息】

Module: [MIPI_RX], Build Time[#1 SMP Wed Apr 19 10:09:18 CST 2023]

-----Combo DEV ATTR-----

Devno	WorkMode	DataType	WDRMode	SyncMode	SyncMode	SyncCode	Endian
0	MIPI	RAW12	NONE	N/A	N/A	N/A	
LinkId		PN Swap					
0, 1, 2, 3, 4, 5, 6, 7, 8		0, 0, 0, 0, 0, 0, 0, 0, 0					

-----MIPI info-----

Devno	EccErr	CrcErr	HdrErr	WcErr	fifofull	decode					
0	0	0	0	2	0	raw12					
Physical:	D0	D1	D2	D3	D4	D5	D6	D7	D8		
	0	ff	6b	ff	ff	ff	ff	ff			
Physical:	D9	D10	D11	D12	D13	D14	D15	D16	D17		
	0	0	0	0	0	0	0	0			
Digital:	D0	D1	D2	D3	CK_HS	CK_ULPS	CK_STOP	CK_ERR	Deskew		
	hs_idle	hs_idle	hs_idle	hs_idle	1	0	0	0	idle		

【调试信息分析】

- MIPI Rx 通过 PHY Wrapper 接收 Sensor 的 MIPI-CSI/SubLVDS/TTL 讯号，由 MAC 内对应的接口进行同步头检测与对齐。
- MAC 将各 Lane 的数据合并成 Pixel 数据，并将数据发送给后级的 ISP。

- PHY Wrapper 由 sensor 的 pixel clock 提供时钟。MAC 内的时钟与后级的 ISP 相同。
- 若要操作数据的 Crop, 可由后级的 ISP 来调试。

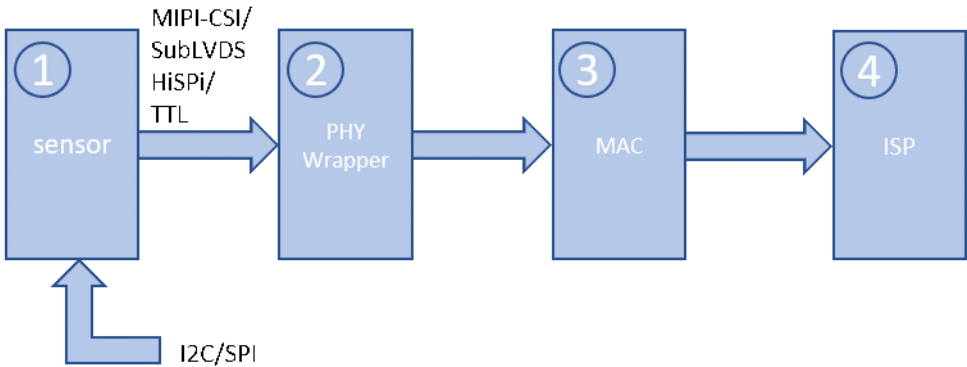


图 3.6: 数据示例图 1

【参数说明】

参数		描述
MIPI DEV ATTR	Devno	MIPI 设备号
	WorkMode	MIPI 设备工作模式: MIPI/SUBLVDS/CMOS 等模式
	DataType	RAW8/RAW10/RAW12 等类型
	WDRMode	WDR 模式: <ul style="list-style-type: none">· NONE· VC· DT· DOL· MANUAL
	LaneId	Lane id
	PN Swap	PN 讯号交换
LVDS DEV ATTR	Devno	MIPI 设备号
	WorkMode	MIPI 设备工作模式: MIPI/SUBLVDS/CMOS 等模式
	DataType	RAW8/RAW10/RAW12 等类型
	WDRMode	WDR 模式: <ul style="list-style-type: none">· NONE· 2To1· 3To1· DOL2To1· DOL3To1
	LaneId	Lane id

下页继续

表 3.1 – 续上页

参数		描述
	PN Swap	PN 讯号交换
	SyncMode	LVDS 的同步码模式: <ul style="list-style-type: none"> · SOF · SAV
	DataEndian	Data 的比特位大小端模式
	SyncCodeEndian	同步码的比特位大小端模式
MIPI Info (仅 MIPI 模式可见)	Devno	MIPI 设备号
	EccErr	ECC 错误的中断计数
	CrcErr	CRC 错误的中断计数
	HdrErr	HDR Flag 错误的中断计数
	WcErr	Word Count 错误的中断计数
	fifofull	Fifofull 的中断计数
	Physical: D0	MIPIRX_PAD0 收到的资料
	Physical: D1	MIPIRX_PAD1 收到的资料
	Physical: D2	MIPIRX_PAD2 收到的资料
	Physical: D3	MIPIRX_PAD3 收到的资料
	Physical: D4	MIPIRX_PAD4 收到的资料
	Physical: D5	MIPIRX_PAD5 收到的资料
	Physical: D6	MIPIRX_PAD6 收到的资料
	Physical: D7	MIPIRX_PAD7 收到的资料
	Physical: D8	MIPIRX_PAD8 收到的资料
	Physical: D9	MIPIRX_PAD9 收到的资料
	Physical: D10	MIPIRX_PAD10 收到的资料
	Physical: D11	MIPIRX_PAD11 收到的资料
	Physical: D12	MIPIRX_PAD12 收到的资料
	Physical: D13	MIPIRX_PAD13 收到的资料
	Physical: D14	MIPIRX_PAD14 收到的资料
	Physical: D15	MIPIRX_PAD15 收到的资料
	Physical: D16	MIPIRX_PAD16 收到的资料
	Physical: D17	MIPIRX_PAD17 收到的资料
	Digital: D0	Sensor data lane 0 state
	Digital: D1	Sensor data lane 1 state
	Digital: D2	Sensor data lane 2 state
	Digital: D3	Sensor data lane 3 state
	Digital: D4	Sensor data lane 4 state
	Digital: D5	Sensor data lane 5 state
	Digital: D6	Sensor data lane 6 state

下页继续

表 3.1 – 续上页

参数	描述
Digital: D7	Sensor data lane 7 state
Digital: D8	Sensor data lane 8 state
Digital: D9	Sensor data lane 9 state
Digital: D10	Sensor data lane 10 state
Digital: D11	Sensor data lane 11 state
Digital: D12	Sensor data lane 12 state
Digital: D13	Sensor data lane 13 state
Digital: D14	Sensor data lane 14 state
Digital: D15	Sensor data lane 15 state
Digital: D16	Sensor data lane 16 state
Digital: D17	Sensor data lane 17 state
CK_HS/CK_ULPS/CK_STOP/CK_IDLE	Sensor clock lane state
Deskew	Deskew 结果

3.7 FAQ

3.7.1 Land id 如何配置

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

下面举例说明，例如 MIPI 与 SENSOR 的引脚硬件连接如下表所示。

SENSOR 管脚	MIPI Lane 管脚
Clock Lane (index = 0)	MIPIRX1_PAD0 (value = 0)
Lane 0 (index = 1)	MIPIRX1_PAD1 (value = 1)
Lane 1 (index = 2)	MIPIRX1_PAD2 (value = 2)
Lane 2 (index = 3)	MIPIRX1_PAD3 (value = 3)
Lane 3 (index = 4)	MIPIRX1_PAD4 (value = 4)
Lane 4 (index = 5)	MIPIRX1_PAD4 (value = 5)
Lane 5 (index = 6)	MIPIRX1_PAD4 (value = 6)
Lane 6 (index = 7)	MIPIRX1_PAD4 (value = 7)
Lane 7 (index = 8)	MIPIRX1_PAD4 (value = 8)
Lane 8 (index = 9)	MIPIRX1_PAD4 (value = 9)
Lane 9 (index = 10)	MIPIRX1_PAD4 (value = 10)
Lane 10 (index = 11)	MIPIRX1_PAD4 (value = 11)
Lane 11 (index = 12)	MIPIRX1_PAD4 (value = 12)
Lane 12 (index = 13)	MIPIRX1_PAD4 (value = 13)
Lane 13 (index = 14)	MIPIRX1_PAD4 (value = 14)
Lane 14 (index = 15)	MIPIRX1_PAD4 (value = 15)
Lane 15 (index = 16)	MIPIRX1_PAD4 (value = 16)
Lane 16 (index = 17)	MIPIRX1_PAD4 (value = 17)
Lane 17 (index = 18)	MIPIRX1_PAD4 (value = 18)

MIPI 的最大 Lane 数加上 Clock 为 18，所以 lane_id 配置如下：

```

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

      索引sensor_lane4, sensor_lane5, sensor_lane6, sensor_lane7, sensor_lane8
      ↓      ↓      ↓      ↓      ↓
land_id={MIPIRX1_PAD5,MIPIRX1_PAD6,MIPIRX1_PAD7,MIPIRX1_PAD8,MIPIRX1_PAD9}

```

3.7.2 MIPI 频率说明

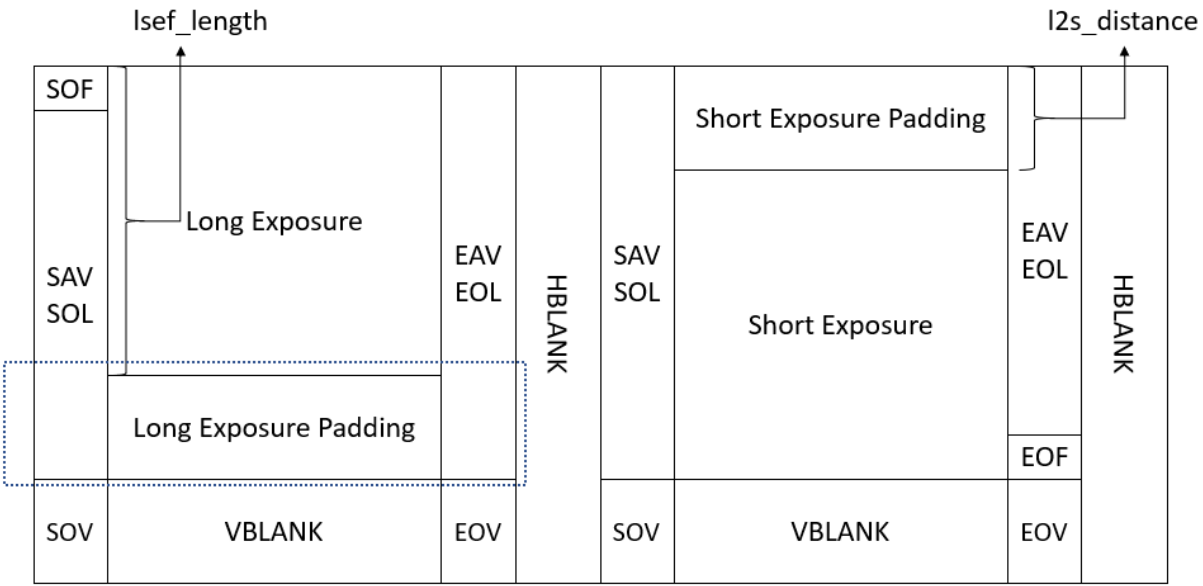
使用以下公式计算 MIPI 每 Lane 最高频率与 VI MAC 的工作频率：

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

3.7.3 Manual WDR 模式使用说明

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

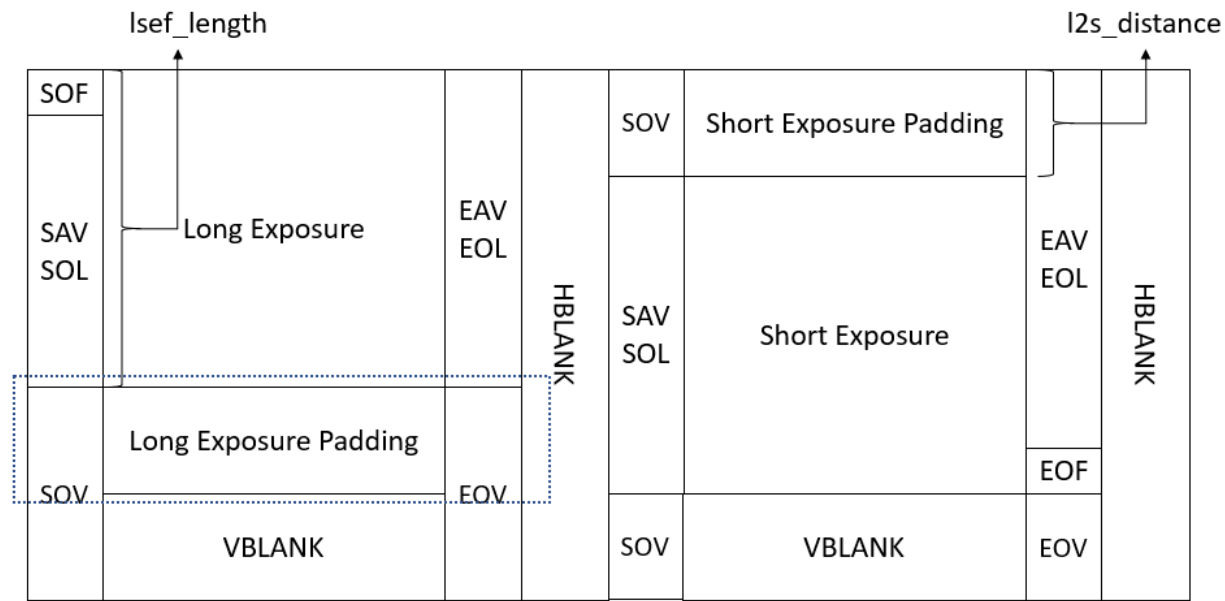
- l2s_distance: 从第一行到第 l2s_distance 行都是长曝光数据，第 l2s_distance+1 行开始长曝光与短曝光交错分配。
- lsef_length: 第 lsef_length+1 行开始都是短曝光帧数据。直到下个垂直同步信号为止。
- 当 discard_padding_lines=1 时，1 到 l2s_distance 行分配方式为 {长-ignore-长-ignore ...}，第 l2s_distance+1 到 lsef_length 行分配方式为 {长-短-长-短...}。第 lsef_length+1 行到下个垂直同步信号分配方式为 {短-ignore-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



Padding data is sent as active lines, `discard_padding_lines = 1`

图 3.7: 数据示例图 2

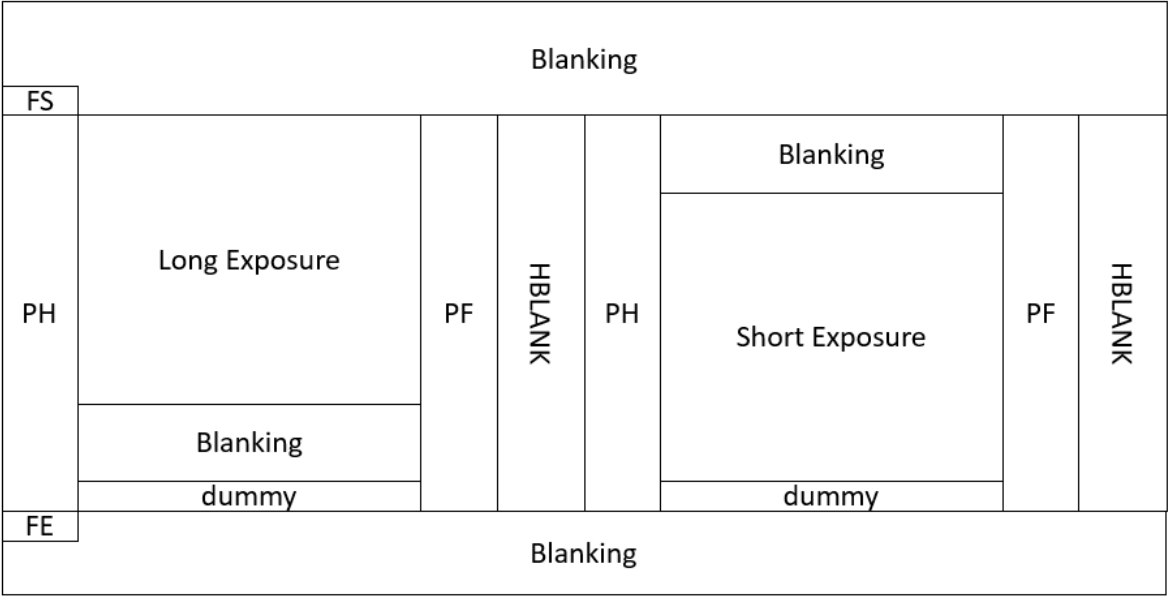
- 当 `discard_padding_lines=0` 时，1 到 `l2s_distance` 行分配方式为 {长-长-长...}，第 `l2s_distance+1` 到 `lsef_length` 行分配方式为 {长-短-长-短...}。第 `lsef_length+1` 行到下个垂直同步信号分配方式为 {短-短-短...}。



Padding data is sent as blanking lines, discard_padding_lines = 0

图 3.8: 数据示例图 3

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



discard_padding_lines = 1
l2s_distance = 0
lsef_length = 1FFF

图 3.9: 数据示例图 4