



CV186AH Debug SOP

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	各模块 proc 信息的开启	3
2.1	MIPI-RX	3
2.2	VI	5
2.3	VPSS	8
2.4	VO	13
2.5	VB	15
2.6	DWA	18
2.7	SYS	20
2.8	VDEC	21
2.9	VENC	23
2.10	RC	26
2.11	jpege	28
2.12	h264e	30
2.13	h265e	32
2.14	REGION	32
2.15	AI (audio input)	35
2.16	AO (audio output)	35
3	Dump Frame 定位问题模块	36
3.1	sensor-raw	36
3.2	vi yuv	36
3.3	vpss yuv	37
3.4	venc yuv	37
3.5	vdec yuv	37
3.6	vo yuv	37
3.7	Audio	38
4	字节要求对齐	40
4.1	VB size 对齐	40
4.2	VDEC 宽高对齐	40
5	系统限制	41
5.1	VI	41
5.2	VPSS	41
5.3	VCODEC	41
5.4	REGION	42
5.5	AUDIO	42
6	历史板子常见问题收集	43

6.1	Sensor	43
6.2	VI	47
6.3	VPSS	48
6.4	VO	51
6.5	REGION	52
6.6	VB/SYS	52
6.7	Audio	53
6.8	VENC	55
6.9	VDEC	55
6.10	Others	56
6.11	工具类	57

修订记录

Revision	Date	Description
v1.0	2023/10/24	First Draft

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 各模块 proc 信息的开启

2.1 MIPI-RX

```
[root@evitek]/mnt/sd# cat /proc/soph/mipi-rx
Module: [MIPI_RX], Build Time[#1 SMP Tue Nov 7 11:43:52 CST 2023]

-----Combo DEV ATTR-----
  Devno  WorkMode  DataType  WDRMode  SyncMode  DataEndian  SyncCodeEndian
    0      MIPI      RAW10      NONE      N/A        N/A          N/A
  LinkId
1, 2, 0, -1, -1, -1, -1, -1  1, 1, 1, 0, 0, 0, 0, 0

-----MIPI info-----
  Devno  EccErr  CrcErr  HdrErr  WcErr  fifofull  decode
    0      1      0      0      0      0      raw10
Physical:  D0      D1      D2      D3      D4      D5      D6      D7      D8
           75      44      f5      0      0      0      0      0
Physical:  D9      D10     D11     D12     D13     D14     D15     D16     D17
           0      0      0      0      0      0      0      0      0
Digital:   D0      D1      D2      D3      D4      D5      D6      D7
           hs_hst  hs_hst  hs_idle hs_idle hs_idle hs_idle hs_idle hs_idle
Digital:   CK_HS   CK_ULPS CK_STOP CK_ERR  Deskew
           1      0      1      0      start
```

参数	含义
Devno	sensor 编号。例如 0 表示 sensor0、1 表示 sensor1、2 表示 sensor2，目前最大只支持 6 路 sensor 同时输入
Work-Mode	表示哪种接口类型 (mipi/sublvds /BT656...)
Date-Type	表示 sensor 数据格式 (raw8/raw10/raw12/ YUV422_8BIT...)
WDR-Mode	wdr 模式 (none 表示非 wdr，常见 wdr mode:VC、DT、Manual)
LinkId	lane 线序配置
PN swap	PN 反转。如果有反转，要将该 lane 配置成 1
Sync-Mode/DataEndian/SyncCodeEnddian	对于 mipi 接口不支持因此无需配置，对于 sublvds 则需要配置
Mipi info	主要是 mipi-rx 解析到的信息
Ec-cErr、crcErr、HDR-Err、WcErr	如果不为 0 表示有出现 Ecc、crc、wc 等校验相关 err，那么需要 check lane mapping 的正确性、mipi 时序、lane 硬件电路
fifofull	如果不为 0 表示 mac 速率太慢，需要提升 mac clk
decode	表示解析出的数据 l 类型 (Raw12/raw10/raw8/YUV422...)
Physical	D0-D8 表示 lane 总线上的数据，进入 hi speed state 后，D0-D4 会有数据跳变
Digital	D0-D8 显示进入 hi speed state 后的每条 data lane 状态。CK_HS、CK_ULPS、CK_ERR、Deskew 表示 clk lane 状态，正常情况下 CK_HS=1，其余的为 0，但是非 continues clk 的情况下也会跳出 CK_HS=1，CK_STOP=1

2.2 VI

```
[root@vitek]#mt/sd# cat /proc/soph/vi
-----MODULE PARAM-----
DetectErrFrame 0
DropErrFrame 0
-----VI MODE-----
DevID PrerawFE PrerawBE Postraw Scaler
0 0 online 0 online 0 offline
-----VI DEV ATTR1-----
DevID DevEn BindPipe Width Height IntfM WkM ScanM
0 0 Y Y 2560 1440 N/A IMUX 1MUX P
-----VI DEV ATTR2-----
DevID AD0 AD1 AD2 AD3 Seq DataType WDRMode
0 0 -1 -1 -1 -1 N/A RGB None
-----VI BIND ATTR-----
DevID PipeNum PipeID
0 1 0
-----VI DEV TIMING ATTR-----
DevID DevTimingEn DevFrmRate DevWidth DevHeight
0 0 0 2560 1440
-----VI CHN ATTR1-----
DevID ChnID Width Height Mirror Flip SrcRate DstRate PkxFmt VideoFmt BindPool
0 0 2560 1440 N N -1 -1 NV21 SDR8 -1
-----VI CHN ATTR2-----
DevID ChnID CompressMode Depth Align
0 0 None 0 32
-----VI CHN OUTPUT RESOLUTION-----
DevID ChnID Mirror Flip Width Height PkxFmt VideoFmt CompressMode FrameRate
0 0 N N 2560 1440 NV21 SDR8 None -1
-----VI CHN ROTATE INFO-----
DevID ChnID Rotate
0 0 0
-----VI CHN EARLY INTERRUPT INFO-----
DevID ChnID Enable LineCnt
0 0 N 0
-----VI CHN CROP INFO-----
DevID ChnID CropEn CoorType CoorX CoorY Width Height TrimX TrimY TrimWid TrimHgt
0 0 0 RAT 0 0 0 0 0 0 0 0
-----VI CHN STATUS-----
DevID ChnID Enable FrameRate IntCnt RecvPic LostFrame VbFail Width Height
0 0 Y 25 9539 9539 0 0 2560 1440
```

【调试信息分析】

记录当前视频输入设备及信道的属性配置以及状态信息。

【参数说明】

参数		描述
MODULE PARAM 模块 参数	DetectErrFrame	信号不稳定时，实时丢弃侦测到的错策略。该参数仅限于调试时使用，正式产品中不建议使用，把该值配成 0，实时丢掉错误帧。 >0: 当检测到连续错误帧数大于该值时，则认为是时序配错，后面的帧不再丢弃； 0: 默认值，表示实时丢弃检测到的错误帧图像。 <0: 关闭检测错误帧功能。
	DropErrFrame	当检测到当前帧是错误帧时，认为接下来的几帧也可能是错误帧，应该丢弃。 0: 默认值，表示不开启连续丢帧功能，只丢弃当前的错误帧； > 0: 该参数表示当检测到图像错误时，连续丢 drop_err_frame 帧 (含当前帧)，不管后面的图像是否正确。
VI MODE VI PIPE 的工作模式	DevID	设备号。有效范围: [0, VI_MAX_DEV_NUM)。
	PrerawFE	PrerawFE 工作模式
	PrerawBE	PrerawBE 工作模式
	Postraw	Postraw 工作模式
	Scaler	Scaler 工作模式

下页继续

表 2.1 – 续上页

参数		描述
VI DEV ATTR1 视频输入设备属性 1	DevID	设备号。有效范围: [0, VI_MAX_DEV_NUM)。
	DevEn	设备使能。 N: 关闭; Y: 打开。
	BindPipe	设备是否绑定 pipe。 N: 未绑定; Y: 已绑定。
	Width	设备宽度, 单位: 像素。
	Height	设备高度, 单位: 像素。
	IntfM	输入模式。MIPI or BT or LVDS
	WkM	工作模式, 现在只有 1MUX
	ScanM	隔行或逐行输入。 取值: {I, P}。VI_SCAN_INTERLACED = I, VI_SCAN_PROGRESSIVE = P
VI DEV ATTR2 视频输入设备属性 2	DevID	设备号。有效范围: [0, VI_MAX_DEV_NUM)。
	AD0	AD 号。
	AD1	AD 号。
	AD2	AD 号。
	AD3	AD 号。
	Seq	数据顺序。 取值: {VUVU, UVUV, UYVY, VYUY, YUYV, YVYU}。
	DataType	输入数据类型, 默认为 RGB。
VI BIND ATTR 绑定关系	WDRMode	WDR 模式。 WDR_2L1: 二合一帧模式 WDR_2F1: 二合一帧模式 WDR_3L1: 三合一帧模式 WDR_3F1: 三合一帧模式 WDR_4L1: 四合一帧模式 WDR_4F1: 四合一帧模式
	DevID	设备号。有效范围: [0, VI_MAX_DEV_NUM)。
	PipeNum	Pipe 数目
	PipeId	PIPE 号。 有效范围: [0, VI_MAX_PIPE_NUM)。
VI DEV TIMING ATTR 自产生时序属性	DevID	设备号。有效范围: [0, VI_MAX_DEV_NUM)。
	DevTimingEn	是否使能自产生时序功能: (is_offline_preraw) N: 关闭; Y: 打开;
	DevFrmRate	用户设置的自产生时序帧率; (当其大于设备支持的最大帧率时, 返回设备最大帧率)

下页继续

表 2.1 – 续上页

参数		描述
	DevWidth	设备宽度, 单位: 像素;
	DevHeight	设备高度, 单位: 像素;
VI CHN ATTR1 物理通道属性 1	DevID	设备号。有效范围: [0, VI_MAX_DEV_NUM)。
	ChnID	通道号。
	Width	通道输出宽度。
	Height	通道输出高度。
	Mirror	mirror 使能 N: 关闭; Y: 打开。
	Flip	flip 使能 N: 关闭; Y: 打开。
	SrcFRate	源帧率。
	DstFRate	目的帧率。
	PixFmt	输出像素格式。
	VideoFmt	输出视频格式。
VI CHN ATT2 物理通道属性 2	DevID	设备号。有效范围: [0, VI_MAX_DEV_NUM)。
	ChnID	通道号。
	CompressMode	是否压缩 N: 关闭; Y: 打开。
	Depth	用户获取通道帧的队列深度。
	Align	通道图像行 stride 对齐。
VI CHN OUTPUT-RESOLUTIONVI 通道输出的图像属性	DevID	设备号。有效范围: [0, VI_MAX_DEV_NUM)。
	ChnID	通道号。
	Mirror	mirror 使能 N: 关闭; Y: 打开。
	Flip	flip 使能 N: 关闭; Y: 打开。
	Width	通道输出宽度。
	Height	通道输出高度。
	PixFmt	输出像素格式。
	VideoFmt	输出视频格式。
	CompressMode	是否压缩 N: 关闭; Y: 打开。
	FrameRate	帧率。
VI CHN ROTATE INFOVI 通道旋转属性	DevID	设备号。有效范围: [0, VI_MAX_DEV_NUM)。
	ChnID	通道号。
	Rotate	通道旋转角度。
VI CHN EARLYIN-TERRUPT INFO 提前上报中断信息	DevID VI_MAX	设备号。有效范围: [0, VI_MAX_DEV_NUM)。
	ChnID	通道号。
	Enable	是否使能提前上报中断的功能。N: 关闭; Y: 打开。
	LineCnt	提前上报中断的行数。
VI CHN CROP INFOVI 通道裁剪属性	DevID	设备号。有效范围: [0, VI_MAX_DEV_NUM)。
	ChnID	通道号。

下页继续

表 2.1 – 续上页

参数		描述
	CropEn	是否使能 CROP 功能。(cvi_isp_s_selection, V4L2_SEL_TGT_CROP) N: 关闭; Y: 打开。
	CoorType	坐标类型。RAT: 相对坐标; ABS: 绝对坐标。
	CoorX	水平方向起始坐标。坐标类型为相对坐标时, 合法取值范围为 [0, 999]; 坐标类型为绝对坐标时, 合法取值范围为 [0, VI_CHN_MAX_WIDTH]。
	CoorY	垂直方向起始坐标。坐标类型为相对坐标时, 合法取值范围为 [0, 999]; 坐标类型为绝对坐标时, 合法取值范围为 [0, VI_CHN_MAX_HEIGHT]。
	Width	CROP RECT 的宽, 不能超出最大图像宽度。
	Height	CROP RECT 的高, 不能超出最大图像高度。
	TrimX	实际图像起始点坐标。
	TrimY	实际图像起始点坐标。
	TrimWid	实际图像宽, 单位: 像素。
	TrimHgt	实际图像高, 单位: 像素。
VI CHN STATUS 通道状态信息	DevID	设备号。有效范围: [0, VI_MAX_DEV_NUM)。
	ChnID	通道号。
	Enable	通道使能。0: 不使能; 1: 使能。
	FrameRate	帧率。
	IntCnt	通道中断计数。
	RecvPic	接收到的图像数目。
	LostFrame	通道丢帧数。
	VbFail	通道获取 VB 失败计数。
	Width	通道宽度。
	Height	通道高度。

2.3 VPSS

```
cat /proc/soph/vpss
```

获取 VPSS 的 proc 信息, 比较各参数, 是否和设置 grp/chn 属性相符合。

```

vpss@cvitek:~$ cat /proc/soph/vpss
Module: [VPSS], Build Time[1 SMP Tue Nov 7 11:43:52 CST 2023]

-----MODULE PARAM-----
vpss_vb_source
CommonVB

-----VPSS GRP ATTR-----
GrpID  MaxW  MaxH  PixFmt  SrcFRate  DstFRate
# 0    2560  1440  NV21    -1          -1

-----VPSS CHN ATTR-----
GrpID  PhyChnID  Enable  MirrorEn  FlipEn  SrcFRate  DstFRate
# 0    # 0      Y       N         N       0         0
# 0    # 1      Y       N         N       -1        -1
# 0    # 2      N       N         N       0         0
# 0    # 3      N       N         N       0         0
# 0    NONE     0       0         0       0         0

-----VPSS GRP CROP INFO-----
GrpID  CropEn  CoordType  Coord  Coord  Width  Height
# 0    N     RAT       0      0      0      0

-----VPSS CHN CROP INFO-----
GrpID  ChnID  CropEn  CoordType  CoordX  CoordY  Width  Height
# 0    # 0  N     RAT       0        0      0      0
# 0    # 1  N     RAT       0        0      0      0
# 0    # 2  N     RAT       0        0      0      0
# 0    # 3  N     RAT       0        0      0      0

-----VPSS GRP WORK STATUS-----
GrpID  RecvCnt  LostCnt  VPSS  StartFailCnt  bStart  CostTime(us)  MaxCostTime(us)  HwCostTime(us)  HwMaxCostTime(us)
# 0    9835    0        0        0              Y       16492          32284            5699            5715

-----VPSS CHN OUTPUT RESOLUTION-----
GrpID  ChnID  Enable  Width  Height  PixFmt  VideoFmt  SendOK  FrameRate
# 0    # 0  Y       2560  1440  NV21    LINEAR    9834    25
# 0    # 1  Y       1280  720    NV21    LINEAR    9834    25
# 0    # 2  N       0      0      RGB_888 LINEAR    0        0
# 0    # 3  N       0      0      RGB_888 LINEAR    0        0

-----VPSS CHN ROTATE INFO-----
GrpID  ChnID  Rotate
# 0    # 0  0
# 0    # 1  0
# 0    # 2  0
# 0    # 3  0

-----VPSS CHN LDC INFO-----
GrpID  ChnID  Enable  Aspect  XRatio  YRatio
# 0    # 0  N     0        0        0
# 0    # 1  N     0        0        0
# 0    # 2  N     0        0        0
# 0    # 3  N     0        0        0

-----VPSS HW STATUS-----
ID  Dev  Online  Status  StartCnt  IntCnt  DutyRatio
# 0  vpss_v0  N  Idle  0  0  0
# 1  vpss_v1  N  Idle  0  0  0
# 2  vpss_v2  N  Idle  0  0  0
# 3  vpss_v3  N  Idle  0  0  0
# 4  vpss_t0  N  Running  13857  13856  13
# 5  vpss_t1  N  Running  13857  13856  13
# 6  vpss_t2  N  Idle  0  0  0
# 7  vpss_t3  N  Idle  0  0  0
# 8  vpss_d0  N  Idle  13856  13856  3
# 9  vpss_d1  N  Idle  0  0  0

```

【调试信息分析】

记录当前 VPSS 属性配置以及状态信息。

【参数说明】

参数		描述
MODULE PARAM	vpss_vb_source	视频缓存池类型。 0: CommonVB 1: UserVB 2: UserIon
VPSS GRP ATTR VPSS GRP 属性。	GrpID	GRP ID 号，有效范围： [0, VPSS_MAX_GRP_NUM)
	MaxW	组输入图像最大宽度。
	MaxH	组输入图像最大高度。
	PixFmt	组输入图像像素格式。
	SrcFRate	GRP 源帧率。
	DstFRate	GRP 目标帧率。
VPSS CHN ATTR VPSS 各物理通道属性	GrpID	GRP ID 号，有效范围： [0, VPSS_MAX_GRP_NUM)
	PhyChnID	通道 ID 号，有效范围： [0, VPSS_MAX_CHN_NUM)

下页继续

表 2.2 – 续上页

参数		描述
	Enable	是否使能该通道。 N: 关闭; Y: 打开。
	MirrorEn	是否使能 mirror 功能。 N: 关闭; Y: 打开。
	FlipEn	是否使能 flip 功能。 N: 关闭; Y: 打开。
	SrcFRate	通道帧率控制: 源帧率。
	DstFRate	通道帧率控制: 目标帧率。
	Depth	用户获取通道图像的队列长度。
	Aspect	幅形比模式 NONE: 关闭幅形比 AUTO: 自动模式 MANUAL: 手动模式
	videoX	视频位置 X 坐标。手动模式有效。
	videoY	视频位置 Y 坐标。手动模式有效。
	videoW	视频宽度。手动模式有效。
	videoH	视频高度。手动模式有效。
	BgColor	幅形比背景色。有效范围: [0x0, 0xFFFFFFFF]
VPSS GRP CROP INFO VPSS 组 CROP 信息	GrpID	GRP ID 号, 有效范围: [0, VPSS_MAX_GRP_NUM)
	CropEn	是否使能 CROP 功能。 N: 关闭 Y: 打开
	CoorType	坐标类型 RAT: 相对坐标 ABS: 绝对坐标。
	CoorX	水平方向起始坐标。
	CoorY	垂直方向起始坐标。
	Width	CROP RECT 的宽, 不能超出最大图像宽度。
	Height	CROP RECT 的高, 不能超出最大图像高度。

下页继续

表 2.2 – 续上页

参数		描述
VPSS CHN CROP INFO 通道 CROP 信息	GrpID	GRP ID 号，有效范围： [0, VPSS_MAX_GRP_NUM)
	ChnID	通道 ID 号，有效范围： [0, VPSS_MAX_CHN_NUM)
	CropEn	是否使能 CROP 功能。 N：关闭 Y：打开。
	CoorType	坐标类型 RAT：相对坐标 ABS：绝对坐标。
	CoorX	水平方向起始坐标。
	CoorY	垂直方向起始坐标。
	Width	CROP RECT 的宽，不能超出最大图像宽度。
	Height	CROP RECT 的高，不能超出最大图像高度。
VPSS GRP WORKSTATUS VPSSGRP 工作状态 信息	GrpID	GRP ID 号，有效范围： [0, VPSS_MAX_GRP_NUM)
	RecvCnt	接收到的图像数
	LostCnt	因队列满而丢弃的图像数。
	StartFailCnt	Start 任务失败次数。
	bStart	是否开始接收图像。
	CostTime(us)	当前完成任务的耗时。
	MaxCostTime(us)	历史上耗时最长任务的执行时间。
	HwCostTime(us)	当前完成任务的硬件耗时
	HwMaxCost-Time(us)	历史上硬件耗时最长任务的执行时间。
VPSS CHN OUTPUT RESOLUTION VPSS 通道输出分辨率	GrpID	GRP ID 号，有效范围： [0, VPSS_MAX_GRP_NUM)
	ChnID	通道 ID 号，有效范围： [0, VPSS_MAX_CHN_NUM)

下页继续

表 2.2 – 续上页

参数		描述
	Enable	通道使能。N: 关闭 Y: 打开。
	Width	目标图像的宽度, 以像素为单位。
	Height	目标图像的高度, 以像素为单位。
	Pixfmt	目标图像的像素格式。
	Videofmt	目标图像的视频格式。
	SendOK	成功发送的图像数量。
	FrameRate	通道输出的实时帧率。
VPSS CHN ROTATE INFO 通道旋转的信息	GrpID	GRP ID 号, 有效范围: [0, VPSS_MAX_GRP_NUM)
	ChnID	通道 ID 号, 有效范围: [0, VPSS_MAX_CHN_NUM)
	Rotate	旋转的角度枚举。
VPSS CHN LDC INFO VPSS 物理通道 LDC 属性	GrpID VPSS	GRP ID 号, 有效范围: [0, VPSS_MAX_GRP_NUM)
	ChnID	通道 ID 号, 有效范围: [0, VPSS_MAX_CHN_NUM)
	Enable	LDC 开关。N: 关闭; Y: 打开。
	Aspect	是否保持幅形比。 N: 不保持幅形比 Y: 保持幅形比
	XRatio	不保持幅形比时候生效。水平方向上的 Crop 比例。
	YRatio	不保持幅形比时候生效。垂直方向上的 Crop 比例。
	XYRatio	保持幅形比时候生效。水平和垂直方向上的整体 Crop 比例。
	XOffset	校正中心点的 X 坐标偏移。
	YOffset	校正中心点的 Y 坐标偏移。
	DistortionRatio	校正强度系数。
VPSS HW STATUS VPSS 硬件工作状态	ID	vpss 硬件编号
	Dev	vpss 硬件名称。

下页继续

表 2.2 – 续上页

参数		描述
	Online	是否 online。 N: offline Y: online
	Status	HW 的工作状态。 Idle: 空闲 Running: 运行中 End: 运行结束 online: 和 isp online
	StartCnt	启动次数。
	IntCnt	中断次数。

2.4 VO

```
[root@vitek]/mnt/sd# cat /proc/soph/codec
Module: [CodecInst] System Build Time [#1 SMP Tue Nov 7 11:43:52 CST 2023]
[root@vitek]/mnt/sd# cat /proc/soph/vb
Module: [VB], Build Time[#1 SMP Tue Nov 7 11:43:52 CST 2023]
-----VB Pools Config-----
MaxPoolCnt(512), MaxBlkCnt(128)
-----COMMON POOL CONFIG-----
PoolId( 0)      Size( 5652480)      Count( 7)
PoolId( 1)      Size( 1474560)      Count( 5)
-----
PoolName      : vbpool
PoolId        : 0
PhysAddr      : 0x16808e000
VirtAddr      : 0xffffffff0f110000
IsComm        : 1
Owner         : -1
BlkSz         : 5652480
BlkCnt        : 7
Free          : 5
MinFree       : 3
-----
BLK  VI    VPSS  VO    RGN  GDC   DMA   DPU   STITCH  IVE   VENC  VDEC  USER
#0    1    0    0    0    0    0    0    0    0    0    0    0
#1    0    0    0    0    0    0    0    0    0    0    0    0
#2    0    0    0    0    0    0    0    0    0    0    0    0
#3    0    0    0    0    0    0    0    0    0    0    0    0
#4    0    0    0    0    0    0    0    0    0    0    0    0
#5    0    0    0    0    0    0    0    0    0    0    0    0
#6    1    0    0    0    0    0    0    0    0    0    0    0
Sum    2    0    0    0    0    0    0    0    0    0    0    0
-----
PoolName      : vbpool
PoolId        : 1
PhysAddr      : 0x16a5ca000
VirtAddr      : 0xffffffffd856000
IsComm        : 1
Owner         : -1
BlkSz         : 1474560
BlkCnt        : 5
Free          : 5
MinFree       : 0
-----
BLK  VI    VPSS  VO    RGN  GDC   DMA   DPU   STITCH  IVE   VENC  VDEC  USER
#0    0    0    1    0    0    0    0    0    0    0    0    0
#1    0    0    1    0    0    0    0    0    0    0    0    0
#2    0    0    0    0    0    0    0    0    0    0    0    0
#3    0    0    0    0    0    0    0    0    0    0    0    0
#4    0    0    1    0    0    0    0    0    0    0    0    0
Sum    0    0    3    0    0    0    0    0    0    0    0    0
-----
```

cat /proc/soph/vo

记录当前 VO 的使用状况及其属性配置，包含设备状态、视频层状态和通道状态。可用于动态获取当前 VO 的使用状态以便于调试或测试。

【调试信息分析】

记录当前 VO 的使用状况及其属性配置，包含设备状态、视频层状态和通道状态。可用于动态获取当前 VO 的使用状态以便于调试或测试。

【参数说明】

参数		描述
DEVICE CONFIG	DevID	设备 ID。 取值范围：[0,VO_MAX_DEV_NUM)
	DevEn	设备是否使能。 N：禁止； Y：使能。
	IntfType	接口类型。 取值范围： CVBS, YPBPR, VGA, BT656, BT1120, LCD, LCD_18BIT, LCD_24BIT, LCD_30BIT, MIPI, MIPI_SLAVE, HDMI, I40
	IntfSync	接口时序。 取值范围：[0,VO_OUTPUT_BUTT)。
	BkClr	设备背景色。十六进制 RGB888 格式。
VIDEO LAYER STATUS 1	DevFrt	设备帧率，即刷新率，与时序相关。
	LayerId	视频层 ID。 取值范围：[0,VO_MAX_LAYER_NUM)。
	VideoEn	视频层是否使能。 N：禁止； Y：使能。
	PixFmt	输入图像像素格式。
	ImgW	视频层画布宽度。
	ImgH	视频层画布高度。
	DispX	显示区域起始横坐标。
	DispY	显示区域起始纵坐标。
	DispW	显示区域宽度。
	DispH	显示区域高度。
	DispFrt	视频层显示帧率。
	LayerId	视频层 ID。 取值范围：[0,VO_MAX_LAYER_NUM)。
	DevId	视频层绑定的设备 ID。 取值范围：[0,VO_MAX_DEV_NUM)。
	EnChNum	通道使能计数。即该视频层有多少个通道处于使能状态。 取值范围：[0,VO_MAX_CHN_NUM)。
	Luma	亮度。 取值范围：[0, 100]。
	Cont	对比度。 取值范围：[0, 100]。
	Hue	色调。 取值范围：[0, 100]。
	Satu	饱和度。 取值范围：[0, 100]。
VIDEO LAYER STATUS 2(continue)	BufLen	显示 buffer 长度。
	LayerId	视频层 ID。 取值范围：[0,VO_MAX_LAYER_NUM)。
	ChnId	通道 ID。 取值范围：[0,VO_MAX_CHN_NUM)。
CHN BASIC INFO		

下页继续

表 2.3 – 续上页

参数		描述
	ChnEn	通道是否使能。 Y: 是; N: 否。
	Prio	通道优先级。 取值范围: [0,VO_MAX_CHN_NUM)。
	ChnX	通道起始横坐标。
	ChnY	通道起始纵坐标。
	ChnW	通道宽度。
	ChnH	通道高度。
	RotAngle	通道旋转角度。 取值范围: [0,ROTATION_BUTT)
CHN PLAY INFO	LayerId	视频层 ID。 取值范围: [0,VO_MAX_LAYER_NUM)。
	ChnId	通道 ID。 取值范围: [0,VO_MAX_CHN_NUM)。
	Show	通道是否显示。 N: 隐藏; Y: 显示。
	Pause	通道是否暂停。 N: 禁止; Y: 使能。
	Thrshd	通道缓冲队列最大可接收的图像帧数。
	ChnFrt	通道帧率。通道播放控制可以通过该数值反映。
	ChnGap(us)	通道帧间隔。与通道帧率成反比。单位: us。
	DispPts	当前正在显示帧的时间戳。单位: us。
	PreDonePts	前一张完成显示帧的时间戳。单位: us。

2.5 VB

```
[root@cvitek]#mnt/sd/install# cat /proc/cvitek/vb
Module: [VB], Build Time[#1 SMP Tue Oct 24 08:55:46 CST 2023]
-----VB PUB CONFIG-----
MaxPoolCnt(512), MaxBlkCnt(128)

-----COMMON POOL CONFIG-----
PoolId( 0)      Size( 5652480)      Count( 7)
PoolId( 1)      Size( 1474560)      Count( 5)

-----vbpool-----
PoolName : vbpool
PoolId : 0
PhysAddr : 0x160002000
VirtAddr : 0xffffffffc01511000
IsComm : 1
Owner : -1
BlkSz : 5652480
BlkCnt : 7
Free : 5
MinFree : 3

BLK  VI  VPSS  VO  RGN  GDC  DMA  DPU  STITCH  IVE  VENC  VDEC  USER
#0  0  0  0  0  0  0  0  0  0  0  0  0
#1  0  0  0  0  0  0  0  0  0  0  0  0
#2  0  0  0  0  0  0  0  0  0  0  0  0
#3  0  0  0  0  0  0  0  0  0  0  0  0
#4  0  0  0  0  0  0  0  0  0  0  0  0
#5  0  0  0  0  0  0  0  0  0  0  0  0
#6  0  0  0  0  0  0  0  0  0  0  0  0
Sum  2  0  0  0  0  0  0  0  0  0  0  0

-----vbpool-----
PoolName : vbpool
PoolId : 1
PhysAddr : 0x1625b0000
VirtAddr : 0xffffffffc0189d5000
IsComm : 1
Owner : -1
BlkSz : 1474560
BlkCnt : 5
Free : 5
MinFree : 4

BLK  VI  VPSS  VO  RGN  GDC  DMA  DPU  STITCH  IVE  VENC  VDEC  USER
#0  0  0  0  0  0  0  0  0  0  0  0  0
#1  0  0  0  0  0  0  0  0  0  0  0  0
#2  0  0  0  0  0  0  0  0  0  0  0  0
#3  0  0  0  0  0  0  0  0  0  0  0  0
#4  0  0  0  0  0  0  0  0  0  0  0  0
Sum  0  0  0  0  0  0  0  0  0  0  0  0
```

```
cat /proc/soph/vb
```

获取当前 VB 模块的 buffer 使用情况。

【调试信息分析】

记录当前 VB 模块的 buffer 使用情况。

【参数说明】

参数		描述
VB PUB CONFIG	MaxPoolCnt	最大的缓存池的个数
	MaxBlkCnt	最大的缓存块的个数
	PoolId	公共缓存池的句柄。
COMMON POOL CONFIG	Size	缓存池内块的大小。
	Count	缓存池内块的个数。
	PoolName	公共/私有缓存池名称，若未设定预设设为 vbpool
PER VB POOL INFO	PoolId	公共/私有缓存池的句柄
	PhysAddr	公共/私有缓存池的开始物理地址。
	VirtAddr	公共/私有缓存池的开始虚拟地址。
	IsComm	是否公共缓存池。 取值：{0, 1}。
	Owner	缓存池的拥有者。 · 2：私有池。 · 1：公共池。
	BlkSz	缓存池内缓存块的大小。
	BlkCnt	缓存池内缓存块的个数。
	Free	缓存池空闲缓存块的个数。
	MinFree	程序运行以来，空闲缓存块的最小剩余个数。 若该计数为 0，则说明可能存在因缓存块不够而丢帧的情况。
	BLK	缓存池内缓存块的句柄。
	BASE/VB/SYS/RGN/ CHNL/VDEC/VPSS/VENC/ H264E/JPEGE/MPEG4E/ H265E/JPEGD/VO/VI/ DIS/RC/AIO/AI/AO/AENC/ ADEC/AUD/VPU/ISP/ IVE/USER/PROC/LOG/ H264D/GDC/PHOTO/FB	模块名 下面对应的数字表示当前模块有多少个地方 占用缓存池内的该缓存块。 · 0：没占用。 N：占用 N 个缓存块。

2.6 DWA

```
[root@vitek]#mt/sd# cat /proc/soph/dwa
Module: [DWA], Build Time[#1 SMP Tue Nov 7 11:43:52 CST 2023]

-----RECENT JOB INFO-----
SeqNo      ModName      jobName      syncIo      u32ID      TaskNum      State      CostTime(us)
-----MAX WASTE TIME JOB INFO-----
SeqNo      ModName      jobName      syncIo      u32ID      TaskNum      State      CostTime(us)
Success    0            Fail        0            BeginNum    0            WaitNum    0            ProcInNum    0
-----DWA TASK STATUS-----
Success    0            Fail        0            Begin        0            Procing     0            WaitNum      0
-----DWA HW AVG STATUS-----
AVGNum     16           TotalHwTm   0            TotalBusyTm 0            TotalCostTm 0            AVGHwTm     0            AVGBusyTm    0            AVGCosTm(us) 0
-----DWA CALL CORRECTION STATUS-----
TaskSuc     0            TaskFail    0            EndSuc       0            EndFail     0            cbCnt       0
```

cat /proc/soph/dwa

【调试信息分析】

记录 DWA 模块最近完成的若干任务、最近耗时最大的任务、历史累计信息。

【参数说明】

参数		描述
RECENT JOB INFO 最近完成的 job 的信息	SeqNo	打印序号。取值范围：[0, 15]
	ModName	提交该 job 的模块名。
	jobName	提交的 job 名。
	syncIo	是否同步。
	u32ID	提交的 job 的 id。
	TaskNum	该 job 包含的 task 数目。
	State	该 job 的处理状态。
	CostTime(us)	该 job 从提交到完成的耗时时长。 单位：us。 该时间包括针对该任务的软件、硬件及中断服务程序处理时间。
	tskNo	提交的 task 的 id。
	UseCoreID	task 用到的 core id。
	inSize(pixel)	task 的输入图像面积 单位：像素
	outSize(pixel)	task 的输出图像面积 单位：像素
	State	该 task 的处理状态。
	type	该 task 的类型。
	HwTime(us)	该 job 在硬件中处理耗时时长。 单位：us。 该时间是硬件处理的时间，一般比 CostTime 要短。
MAX WASTE TIME JOBINFO 最近耗时最大的 job 信息	各项同 RECENT 最近 JOB INFO 的成员 INFO 的成员	通过该组值可知最近的 DWA 运行性能，以及是否出现过 DWA 处理不及时的情况。

下页继续

表 2.4 – 续上页

参数		描述
DWA JOB STATUS DWA 任务状态	Success	累计成功处理的 job 数。
	Fail	累计处理失败的 job 数。 当 DWA 提交任务给驱动层并失败时加 1。该值增加时可通过查看日志了解失败原因。
	Cancel	累计的取消的 job 数。 当调用 cancel Job 接口时加 1。
	BeginNum	用户已创建任务但还未提交 (EndJob 接口) 的 job 数。
	WaitNum	用户已提交 (EndJob 接口) 但还未提交给硬件处理的 task 数。
	ProcingNum	正在进行硬件处理的 task 数。
DWA TASK STA- TUSDWA Task 状态	Success	累计成功处理的 Task 数。一个 job 包含 1 到多个 task, 所以 1 个 job 成功表明其下的多个 task 都成功, 故该值累加的比 job 的 Success 项更快。 当硬件处理一个 Job 成功时, 该值累加 job 下的 task 数。
	Fail	累计处理失败的 Task 数。 一个 job 失败将导致其下的所有 task 失败。当 Job 执行失败时, 该项累加其下失败的 task 数。 该值增加时可通过查看日志了解失败原因。
	Cancel	累计 Cancel 的 Task 数。 当调用 cancelJob 接口时, 即取消了 job 下的所有 task 的执行, 该项累加。
	Begin	用户已创建任务但还未添加给 job 的 task 数。
	Procing	提交给硬件正在处理的 task 数
	WaitNum	已添加到 Job 下, 但还未提交给硬件处理的 task 数
	AVGNum	平均的总数量。 完成一个 job, 该值加 1。
DWA HW AVG STATUS 硬件平均状态	TotalHwTm(us)	DWA 硬件总处理时间。 单位: us
	TotalBusyTm(us)	Job 提交到提交到驱动层的总时间。 单位: us
	TotalCostTm(us)	Job 提交到 job 做完的总时间。 单位: us
	AVGHwTm(us)	DWA 硬件平均处理时间。 单位: us
	AVGBusyTm(us)	Job 提交到提交到驱动层的平均时间。 单位: us
	AVGCostTm(us)	Job 提交到 job 做完的平均时间。 单位: us

下页继续

表 2.4 – 续上页

参数		描述
DWA CALL CORRECTION STATUS FISHEYE 校正处理状态	TaskSuc	添加 FISHEYEtask 成功的次数。
	TaskFail	添加 FISHEYEtask 失败的次数。
	EndSuc	DWA 提交 FISHEYEjob 成功的次数。
	EndFail	DWA 提交 FISHEYEjob 失败的次数。
	CbCnt	DWA 校正任务处理完成后回调的次数。

2.7 SYS

cat /proc/soph/sys

```
[root@cvitek]/mnt/sd# cat /proc/soph/sys
Module: [SYS], Version[cv1835_v1.2.0-3011-gfe2a9eb6e-64bit], Build Time[#1 SMP Tue Nov 7 11:43:52 CST 2023]
-----BIND RELATION TABLE-----
1stMod  1stDev  1stChn  2ndMod  2ndDev  2ndChn  3rdMod  3rdDev  3rdChn
V1      0        0      VPSS    0        0      VENC    0        0
VPSS    0        1      VD      1        0      null    0        0
-----
```

【调试信息分析】

记录当前 SYS 模块的使用情况。

【参数说明】

参数		描述
BIND RE- LATION TABLE	1stMod	绑定关系中第一级的模块名，数据由第一级发送给第二级。
	1stDev	绑定关系中第一级的设备号，数据由第一级发送给第二级。
	1stChn	绑定关系中第一级的通道号，数据由第一级发送给第二级。
	2ndMod	绑定关系中第二级的模块名，数据由第一级发送给第二级。
	2ndDev	绑定关系中第二级的设备号，数据由第一级发送给第二级。
	2ndChn	绑定关系中第二级的通道号，数据由第一级发送给第二级。
	3rdMod	绑定关系中第三级的模块名，如果有第三级绑定关系，则数据由第二级发送给第三级，否则显示为 null。
	3rdDev	绑定关系中第三级的设备号。
	3rdChn	绑定关系中第三级的通道号。

2.8 VDEC

cat /proc/soph/vdec

```

$ cat /proc/1000000000/status
Name: /usr/bin/ffmpeg
Uptime: 1000000000s
Pid: 1000000000
PPid: 1000000000
Parent: /usr/bin/ffmpeg
Arch: x86_64
OpSys: Linux
Kernel: 4.15.0-101-generic
Init: systemd
Cpus: 2
MemTotal: 131072
MemFree: 100000000
MemAvailable: 100000000
SwapTotal: 0
SwapFree: 0
SwapCached: 0
Active: 0
Inactive: 0
ActiveAnon: 0
InactiveAnon: 0
TotalPageTables: 200
Faults: 0
StalledCpuCycles: 0
VmPeak: 0
VmSize: 0
VmLss: 0
VmDss: 0
VmPss: 0
VmHss: 0
VmPss_Peak: 0
VmPss_Max: 0
VmPss_Min: 0
VmPss_Avg: 0
VmPss_StdDev: 0
VmPss_Max2: 0
VmPss_Max3: 0
VmPss_Max4: 0
VmPss_Max5: 0
VmPss_Max6: 0
VmPss_Max7: 0
VmPss_Max8: 0
VmPss_Max9: 0
VmPss_Max10: 0
VmPss_Max11: 0
VmPss_Max12: 0
VmPss_Max13: 0
VmPss_Max14: 0
VmPss_Max15: 0
VmPss_Max16: 0
VmPss_Max17: 0
VmPss_Max18: 0
VmPss_Max19: 0
VmPss_Max20: 0
VmPss_Max21: 0
VmPss_Max22: 0
VmPss_Max23: 0
VmPss_Max24: 0
VmPss_Max25: 0
VmPss_Max26: 0
VmPss_Max27: 0
VmPss_Max28: 0
VmPss_Max29: 0
VmPss_Max30: 0
VmPss_Max31: 0
VmPss_Max32: 0
VmPss_Max33: 0
VmPss_Max34: 0
VmPss_Max35: 0
VmPss_Max36: 0
VmPss_Max37: 0
VmPss_Max38: 0
VmPss_Max39: 0
VmPss_Max40: 0
VmPss_Max41: 0
VmPss_Max42: 0
VmPss_Max43: 0
VmPss_Max44: 0
VmPss_Max45: 0
VmPss_Max46: 0
VmPss_Max47: 0
VmPss_Max48: 0
VmPss_Max49: 0
VmPss_Max50: 0
VmPss_Max51: 0
VmPss_Max52: 0
VmPss_Max53: 0
VmPss_Max54: 0
VmPss_Max55: 0
VmPss_Max56: 0
VmPss_Max57: 0
VmPss_Max58: 0
VmPss_Max59: 0
VmPss_Max60: 0
VmPss_Max61: 0
VmPss_Max62: 0
VmPss_Max63: 0
VmPss_Max64: 0
VmPss_Max65: 0
VmPss_Max66: 0
VmPss_Max67: 0
VmPss_Max68: 0
VmPss_Max69: 0
VmPss_Max70: 0
VmPss_Max71: 0
VmPss_Max72: 0
VmPss_Max73: 0
VmPss_Max74: 0
VmPss_Max75: 0
VmPss_Max76: 0
VmPss_Max77: 0
VmPss_Max78: 0
VmPss_Max79: 0
VmPss_Max80: 0
VmPss_Max81: 0
VmPss_Max82: 0
VmPss_Max83: 0
VmPss_Max84: 0
VmPss_Max85: 0
VmPss_Max86: 0
VmPss_Max87: 0
VmPss_Max88: 0
VmPss_Max89: 0
VmPss_Max90: 0
VmPss_Max91: 0
VmPss_Max92: 0
VmPss_Max93: 0
VmPss_Max94: 0
VmPss_Max95: 0
VmPss_Max96: 0
VmPss_Max97: 0
VmPss_Max98: 0
VmPss_Max99: 0
VmPss_Max100: 0
VmPss_Max101: 0
VmPss_Max102: 0
VmPss_Max103: 0
VmPss_Max104: 0
VmPss_Max105: 0
VmPss_Max106: 0
VmPss_Max107: 0
VmPss_Max108: 0
VmPss_Max109: 0
VmPss_Max110: 0
VmPss_Max111: 0
VmPss_Max112: 0
VmPss_Max113: 0
VmPss_Max114: 0
VmPss_Max115: 0
VmPss_Max116: 0
VmPss_Max117: 0
VmPss_Max118: 0
VmPss_Max119: 0
VmPss_Max120: 0
VmPss_Max121: 0
VmPss_Max122: 0
VmPss_Max123: 0
VmPss_Max124: 0
VmPss_Max125: 0
VmPss_Max126: 0
VmPss_Max127: 0
VmPss_Max128: 0
VmPss_Max129: 0
VmPss_Max130: 0
VmPss_Max131: 0
VmPss_Max132: 0
VmPss_Max133: 0
VmPss_Max134: 0
VmPss_Max135: 0
VmPss_Max136: 0
VmPss_Max137: 0
VmPss_Max138: 0
VmPss_Max139: 0
VmPss_Max140: 0
VmPss_Max141: 0
VmPss_Max142: 0
VmPss_Max143: 0
VmPss_Max144: 0
VmPss_Max145: 0
VmPss_Max146: 0
VmPss_Max147: 0
VmPss_Max148: 0
VmPss_Max149: 0
VmPss_Max150: 0
VmPss_Max151: 0
VmPss_Max152: 0
VmPss_Max153: 0
VmPss_Max154: 0
VmPss_Max155: 0
VmPss_Max156: 0
VmPss_Max157: 0
VmPss_Max158: 0
VmPss_Max159: 0
VmPss_Max160: 0
VmPss_Max161: 0
VmPss_Max162: 0
VmPss_Max163: 0
VmPss_Max164: 0
VmPss_Max165: 0
VmPss_Max166: 0
VmPss_Max167: 0
VmPss_Max168: 0
VmPss_Max169: 0
VmPss_Max170: 0
VmPss_Max171: 0
VmPss_Max172: 0
VmPss_Max173: 0
VmPss_Max174: 0
VmPss_Max175: 0
VmPss_Max176: 0
VmPss_Max177: 0
VmPss_Max178: 0
VmPss_Max179: 0
VmPss_Max180: 0
VmPss_Max181: 0
VmPss_Max182: 0
VmPss_Max183: 0
VmPss_Max184: 0
VmPss_Max185: 0
VmPss_Max186: 0
VmPss_Max187: 0
VmPss_Max188: 0
VmPss_Max189: 0
VmPss_Max190: 0
VmPss_Max191: 0
VmPss_Max192: 0
VmPss_Max193: 0
VmPss_Max194: 0
VmPss_Max195: 0
VmPss_Max196: 0
VmPss_Max197: 0
VmPss_Max198: 0
VmPss_Max199: 0
VmPss_Max200: 0
VmPss_Max201: 0
VmPss_Max202: 0
VmPss_Max203: 0
VmPss_Max204: 0
VmPss_Max205: 0
VmPss_Max206: 0
VmPss_Max207: 0
VmPss_Max208: 0
VmPss_Max209: 0
VmPss_Max210: 0
VmPss_Max211: 0
VmPss_Max212: 0
VmPss_Max213: 0
VmPss_Max214: 0
VmPss_Max215: 0
VmPss_Max216: 0
VmPss_Max217: 0
VmPss_Max218: 0
VmPss_Max219: 0
VmPss_Max220: 0
VmPss_Max221: 0
VmPss_Max222: 0
VmPss_Max223: 0
VmPss_Max224: 0
VmPss_Max225: 0
VmPss_Max226: 0
VmPss_Max227: 0
VmPss_Max228: 0
VmPss_Max229: 0
VmPss_Max230: 0
VmPss_Max231: 0
VmPss_Max232: 0
VmPss_Max233: 0
VmPss_Max234: 0
VmPss_Max235: 0
VmPss_Max236: 0
VmPss_Max237: 0
VmPss_Max238: 0
VmPss_Max239: 0
VmPss_Max240: 0
VmPss_Max241: 0
VmPss_Max242: 0
VmPss_Max243: 0
VmPss_Max244: 0
VmPss_Max245: 0
VmPss_Max246: 0
VmPss_Max247: 0
VmPss_Max248: 0
VmPss_Max249: 0
VmPss_Max250: 0
VmPss_Max251: 0
VmPss_Max252: 0
VmPss_Max253: 0
VmPss_Max254: 0
VmPss_Max255: 0
VmPss_Max256: 0
VmPss_Max257: 0
VmPss_Max258: 0
VmPss_Max259: 0
VmPss_Max260: 0
VmPss_Max261: 0
VmPss_Max262: 0
VmPss_Max263: 0
VmPss_Max264: 0
VmPss_Max265: 0
VmPss_Max266: 0
VmPss_Max267: 0
VmPss_Max268: 0
VmPss_Max269: 0
VmPss_Max270: 0
VmPss_Max271: 0
VmPss_Max272: 0
VmPss_Max273: 0
VmPss_Max274: 0
VmPss_Max275: 0
VmPss_Max276: 0
VmPss_Max277: 0
VmPss_Max278: 0
VmPss_Max279: 0
VmPss_Max280: 0
VmPss_Max281: 0
VmPss_Max282: 0
VmPss_Max283: 0
VmPss_Max284: 0
VmPss_Max285: 0
VmPss_Max286: 0
VmPss_Max287: 0
VmPss_Max288: 0
VmPss_Max289: 0
VmPss_Max290: 0
VmPss_Max291: 0
VmPss_Max292: 0
VmPss_Max293: 0
VmPss_Max294: 0
VmPss_Max295: 0
VmPss_Max296: 0
VmPss_Max297: 0
VmPss_Max298: 0
VmPss_Max299: 0
VmPss_Max300: 0
VmPss_Max301: 0
VmPss_Max302: 0
VmPss_Max303: 0
VmPss_Max304: 0
VmPss_Max305: 0
VmPss_Max306: 0
VmPss_Max307: 0
VmPss_Max308: 0
VmPss_Max309: 0
VmPss_Max310: 0
VmPss_Max311: 0
VmPss_Max312: 0
VmPss_Max313: 0
VmPss_Max314: 0
VmPss_Max315: 0
VmPss_Max316: 0
VmPss_Max317: 0
VmPss_Max318: 0
VmPss_Max319: 0
VmPss_Max320: 0
VmPss_Max321: 0
VmPss_Max322: 0
VmPss_Max323: 0
VmPss_Max324: 0
VmPss_Max325: 0
VmPss_Max326: 0
VmPss_Max327: 0
VmPss_Max328: 0
VmPss_Max329: 0
VmPss_Max330: 0
VmPss_Max331: 0
VmPss_Max332: 0
VmPss_Max333: 0
VmPss_Max334: 0
VmPss_Max335: 0
VmPss_Max336: 0
VmPss_Max337: 0
VmPss_Max338: 0
VmPss_Max339: 0
VmPss_Max340: 0
VmPss_Max341: 0
VmPss_Max342: 0
VmPss_Max343: 0
VmPss_Max344: 0
VmPss_Max345: 0
VmPss_Max346: 0
VmPss_Max347: 0
VmPss_Max348: 0
VmPss_Max349: 0
VmPss_Max350: 0
VmPss_Max351: 0
VmPss_Max352: 0
VmPss_Max353: 0
VmPss_Max354: 0
VmPss_Max355:
```

【调试信息分析】

记录当前解码通道的使用状况及其属性配置。可用于检查属性配置以及当前解码通道统计状态。

【参数说明】

参数		描述
MODULEPARAM	VdecMaxChnNum	vdec 模块支持的最大通道数
	MiniBufMode	是否使用码流缓冲区压缩模式 0: 未使用 1: 使用 (仅仅在码流按照 frame 模式送入解码器时, 该参数有效)
	enVdecVBSource	分配给 VDEC 模块的 VB 类型 1: module VB 2: private VB 3: user VB
	ParallelMode	VDH 解码模式 0: non-parallel mode 1: parallel mode
	MaxPicWidth	VDEC 模块支持的图像最大宽度
	MaxPicHeight	VDEC 模块支持的图像最大高度
	MaxSliceNum	VDEC 模块支持的最大 slice 数量
	VdhMsgNum	vdh message 的数量
	VdhBinSize	VDH 解码时二进制缓冲区的大小
	VdhExtMemLevel	VDH 解码的片外内存分配级别
	MaxJpegeWidth	Jpeg 解码时支持的图像最大宽度
	MaxJpegeHeight	Jpeg 解码时支持的图像最大高度
	SupportProgressive	是否支持 progressive format 0: 否 1: 是
	DynamicAllocate	progressive format 时内存分配模式 0: 静态分配 1: 动态分配
	CapStrategy	解码图像时, 最大宽度和最大高度的确定策略 0: 基于 module 1: 基于 channel
CHNCOMMATTR &PARAMS	ID	VDEC 通道 ID
	TYPE	VDEC 支持的通道格式类型 PT_H264 PT_H265 PT_MJPEG PT_JPEG

下页继续

表 2.5 – 续上页

参数		描述
	MaxW	解码图像配置的最大宽度
	MaxH	解码图像配置的最大高度
	Width	解码图像的宽度
	Height	解码图像的高度
	StrmInputMode	VDEC 通道的码流传输模式 主要可以分为两类： FRAME, STREAM, and COMPAT: 在兼容模式下，按照 FRAME、STEAM 进行传输 BLOCK, NOBLOCK and TIMEOUT: 按照阻塞、非阻塞或者超时模式进行设置
	StrBufSize	码流的缓冲区大小
	FrmBufSize	帧缓冲区大小，仅仅在 private VB 模式下有效
	FrmBufCnt	帧缓冲区的数量，仅仅在 private VB 模式下有效
	TmvBufSize	TMV 缓冲区大小，仅仅在 private VB 模式下有效
	DispNum	要显示的帧的数量 Value range: [0, 16]
	DispMode	显示模式 Value range: PLAYBACK and PREVIEW
	SetUserPic	是否设置用户图像
	EnUserPic	是否开启用户图像设置
	Rotation	VDEC 解码时的旋转角度
	PicPoolId	帧缓冲区所在的 VB pool ID, 仅仅在 private VB 和 user VB 模式下有效
	TmvPoolId	TMV 缓冲区所在的 VB pool ID, 仅仅在 private VB 和 user VB 模式下有效
	STATE	是否 VDEC 通道开始接收码流 START: 通道开始接收码流 STOP: 通道停止接收码流
CHN VIDEO ATTR & PARAMS	ID	解码通道 ID
	VfmwID	视频固件 (VFMW) 通道 ID
	RefNum	参考帧数量 Value range: [0, 16]
	TemporalMvp	是否支持时域运动向量预测
	ErrThr	码流错误率阈值
	DecMode	解码模式
	OutPutOrder	解码图像的输出序列
	Compress	是否解码输出图像可以被压缩
	VideoFormat	待解码图像的数据格式
	MaxVPS	支持 VPS 参数集的最大数量，仅仅对 H.265 解码有效
	MaxSPS	SPS 参数集的最大数量
	MaxPPS	PPS 参数集的最大数量

下页继续

表 2.5 - 续上页

参数		描述
	MaxSlice	slice 的最大数量
CHN image ATTR & PARAMS	ID	解码通道 ID
	PixelFormat	JPEG 图像的输出格式
	Alpha	ARGB 格式的 JPEG 图像的全局 alpha 值
DEBUG STATE	VdecDebugMask	middleware debug 时的 mask 值
	VdecStartFrmIdx	middleware debug 时的开始帧位置
	VdecEndFrmIdx	middleware debug 时的结束帧位置
	VdecDumpPath	bitstream 的 dump 路径

2.9 VENC

cat /proc/soph/venc

[illegible]

【调试信息分析】

记录当前视频编码属性配置以及状态信息。

【参数说明】

参数		描述
MODULE PARAM	VencBuffer- Cache	编码码流的缓冲是否使用 cache mode 0: 否 1: 是
	Frame- BufRecycle	用于存储参考帧和高级 smart P 帧的空闲缓冲区是否在编码期间被回收 0: 不回收 1: 回收
	VencMax- ChnNum	编码支持的最大通道数量

参数		描述
VENC CHN ATTR1	ID	编码通道 ID
	Width	编码通道宽度
	Height	编码通道高度
	Type	编码通道类型
	RcMode	码率控制模式
	bIsoSendFrameEn	是否隔离 SendFrame/GetStream 0: 否 1: 是
	ByFrame	获取编码码流的模式 N(0): by packet Y(1): by frame
	Sequence	序号 当流按帧获取时，它表示帧序列号。 当以包为单位获取流时，表示包的序号。
	LeftBytes	码流缓冲区中剩余的字节数
	LeftFrm	码流缓冲区中剩余的帧数
	CurPacks	当前帧的码流数据包数量 (暂不支持)
	GopMode	GOP 模式
	prio	通道优先级

参数		描述
VENC CHN ATTR 2	VeStr	是否开始编码
	SrcFr	输入帧率
	TarFr	经过编码器控制后的输出帧率
	Timeref	繁忙队列中最新帧的时间戳
	PixFmt	正在编码的视频帧格式
	PicAddr	正在编码的视频帧地址
	Wake-UpFrmCnt	当通道使用超时或以阻塞方式获取流时，指定唤醒阻塞接口的帧数

参数		描述
VENC CROP INFO	ID	编码通道 ID
	CropEn	是否开启裁剪
	StartX	开始要裁剪的图像的水平坐标
	StartY	开始要裁剪的图像的垂直坐标
	Width	裁剪图像的宽度
	Height	裁剪图像的高度
ROI INFO	ID	通道 ID
	Index	ROI 区域的索引
	bRoiEn	是否开启 ROI 功能
	bAbsQp	当前 ROI 区域是否使用绝对 qp
	Qp	ROI 区域的 qp 值
	Width	ROI 区域宽度 (unit: pixel)
	Height	ROI 区域高度 (unit: pixel)
	StartX	ROI 区域的水平坐标 (unit: pixel)
	StartY	ROI 区域的垂直坐标 (unit: pixel)
VENC PTS STATE 通道接收到 帧的时间戳	ID	Channel ID
	RcvFirst-FrmPts	通道接收到的第一帧的时间戳
	RcvFrmPts	通道接收到的当前帧的时间戳
	No.SendFramePerSec	输入的 VENC 的 FPS (from VI or VPSS or app)
	No.EncFramePerSec	VENC 通道的输出 FPS
	HwEncTime	硬件编码时间消耗
	VencDebug-Mask	middleware 层的 debug mask
Debug Level STATE	VencStartFrmIdx	middleware debug 时的开始帧位置
	VencEndFrmIdx	middleware debug 时的结束帧位置
	VencDump-Path	YUV 的 dump 路径
	VencNo-DataTimeout	当没有数据的超时时间限制

2.10 RC

cat /proc/soph/rc

查看编码时的码率控制信息

[illegible]

【调试信息分析】

记录当前编码码率控制的信息。

【参数说明】

参数		描述
BASE PARAMS1	ChnId	VENC 通道 ID
	Gop	图像编码组 (GOP)
	StatTm	比特率统计 (单位: 秒)
	ViFr	VI 传输图像的帧率
	TrgFr	编码的目标帧率
	ProType	编码类型
	RcMode	码率控制模式 (CBR, VBR, QPMAP, or FixQp)
	Br(kbps)	比特率, 单位为 kbit/s。
	FluLev	码率波动级别, 仅对 CBR 模式有效
	IQp	I 帧 QP, 仅对 FixQp 模式有效
	PQp	P 帧 QP, 仅对 FixQp 模式有效
	BQp	B 帧 QP, 仅对 FixQp 模式有效 (当前不支持)
BASE PARAMS 2	ChnId	VENC 通道 ID
	EnableIdr	IDR 使能开关 Y: 开启 N: 关闭
	bQpMapEn	IDR 使能开关 Y: 开启 N: 关闭
	QpMapMode	CU32 或 CU64 使用的 QP 值模式 MEANQP , MINQP , and MAXQP 分别表示平均 QP, 最小 QP, 最大 QP
	u32RowQpDelta	每个宏块行相对于起始 QP 值的起始 QP 值 Range:[0, 10]
	s32InitialDelay	码率控制的初始化时延 (ms) Range:[10, 3000]
	VariFpsEn	是否开启可变 FPS 0: 关闭 1: 开启
	u32ThrdLv	控制宏块级比特率的阈值 Range:[0, 4]
	BgEnhanceEn	是否开启背景增强 0: 否 1: 是
	BgDeltaQp	设置背景区域的 DeltaQP Range:[-51, 51]

参数		描述
GOP MODE ATTR		
	ChnId	编码通道 ID
	GopMode	GOP 模式
	IpQpDelta	I 帧相对于 P 帧的 QP Delta 在 SmartP 模式下显示背景帧和 p 帧的 QP 增量。 Value range: [-10, +30]
	SPInterval	特殊 P 帧的间隔 取值范围: 小于等于 GOP 值。
	SPQpDelta	特殊 P 帧相对于普通 P 帧的 QP delta Value range: [-10, +30]
	BFrmNum	B 帧的数量 (当前不支持) Value range: [1, 3]
	BQpDelta	B 帧相对于 P 帧的 QP delta (当前不支持) Value range: [-10, +30]
	BgInterval	背景帧的间隔 取值范围: 大于等于 GOP 值。
	ViQpDelta	虚拟 I 帧相对于普通 P 帧的 QP delta
RUN CBR PARAM		更详细可以参考 VENC_PARAM_H264_CBR_S 结构体
RUN VBR PARAM		更详细可以参考 VENC_PARAM_H264_VBR_S VENC_PARAM_H264_VBR_S VENC_PARAM_H264_AVBR_S VENC_PARAM_H264_QVBR_S

2.11 jpeg

cat /proc/soph/jpeg

查看 jpeg 的编解码信息

```
[root@rockchip ~]# cat /proc/soph/jpeg
Module: [PROC] System Build Time [45 SMP Thu Nov 23 10:35:37 CST 2023]
=====
OnePack: 0   JpegPicBufMode: 0   JpegClearStreamBuf: 0   JpegDerivingMode: 0
SingleBufBuf: 0   SingleBufBufSize: 0   JpegBufSize: 0
JpegParamOrder: 1 10 12 7 2 4 0 13 0 0 0 0 0 0 0 0
=====
CPU: 0   JPEG: 0   PicType: N/A   RealWidth: 3200   RealHeight: 1800   Width: 3200   Height: 1800   BufSize: 10485760   BufFmt: 1   PCU: 1   QFactor: 0   CMode: 0   Buffer: 0
```

【调试信息分析】

记录 JPEG 编码过程中，各通道的编码属性、状态。

【参数说明】

参数		描述
MODULE PARAM	OnePack	获取流的方式 0: 按照 multi-packet 模式获取码流 1: 按照 single-packet 模式获取码流.
	JpegeMiniBufMode	分配码流缓冲区的方式 0: 根据分辨率确定码率缓冲区 1: 码流缓冲区的最小值是 32kb, 用户需要确保分配的码流缓冲区是合适的
	JpegClearStreamBuf	设置 JPEG 编码通道属性时是否清除流缓冲区。 0: 保留码流缓冲区和上下文计数 1: 清除
	JpegeDeringMode	为 JPEG 编码通道启用 Dering 效果模式 0: 关闭 1: 开启. 在相同的量化表和 Qfactor 下, 可以减少环现象, 减小图像文件大小, 但也会丢失一些图像清晰度和细节。
	SingleEsBuf	在 n 路编码中使用单个输出编码流缓冲区 0: 关闭 1: 开启
	SingleEsBufSize	n 路编码中单个 es 缓冲区的大小
	JpegFormat	Jpeg 图像格式 JPEGE_FORMAT_DEFAULT JPEGE_FORMAT_TYPE_1 JPEGE_FORMAT_CUSTOM

参数		描述
CHN ATTR	ID	通道 ID
	bMjpeg	是否开启 Mjpeg 编码 No: jpeg Yes: mjpeg 编码
	PicType	图片类型: YVU422 or YVU420
	MaxWidth	编码通道的最大宽度
	MaxHeight	编码通道的最大高度
	Width	图像宽度
	Height	图像高度
	BufSize	码流缓冲区大小 (unit: byte)
	ByFrm	是否按帧获取码流 Value: {0, 1}
	MCU	每个嵌入式 CPU 子系统 (ECS) 中最小编码单元 (mcu) 的数量
	Qfactor	当前通道 Qfactor
	C2GEn	是否启用 Color-to-gray Value: {0, 1}
	DcfEn	JPEG 图像是否有缩略图 Value: {0, 1}

2.12 h264e

cat /proc/soph/h264e

查看 264 的编解码信息

```
(root@vitek) /proc/soph# cat h264e
Module: [h264e] System Build Time [Fri SMP Tue Nov 21 13:44:14 CST 2023]
-----MODULE PARAM-----
OutBuf: 0 H264Sources: 0 PowerSaveEn: 0 MinBufMode: 0 GpbitGrnEn: 0 UserDataMaxLen: 3872
SingleBuf: 0 SingleBufSize: 0
CHN ATTR
ID: 0 MaxWidth: 1920 MaxHeight: 1080 Width: 1920 Height: 1080 C2GEn: 0 BufSize: 0 ByFrame: 1 GopMode: NORMALP MaxStrCnt: 0
RefParam: SWO Base: 0 Enhance: 0 RndRefShareBuf: 0
ID: 0 EndPred: Y
Syntax Info
ID: 0 Profile: high
```

【调试信息分析】

记录当前 H.264 视频编码属性配置以及状态信息。

【参数说明】

参数		描述
MODULE PARAM	OnePack	获取流的方式 0: 按照 multi-packet 模式获取码流 1: 按照 single-packet 模式获取码流.
	H264eVBSource	参考帧和重构帧的 VB 按何种方式获得 2: The private VBs are used. 3: The VBs are allocated by the user.
	PowerSaveEn	低功耗参数控制开关 0: 关闭 1: 开启
	MiniBufMode	分配码流缓冲区的方式 0: 根据分辨率确定码率缓冲区 1: 码流缓冲区的最小值是 32kb, 用户需要确保分配的码流缓冲区是合适的
	bQpHstgrmEn	是否在高级流信息中显示 QP 直方图 0: no 1: yes
	UserDataMaxLen	用户数据段的最大字节数
	SingleEsBuf	在 n 路编码中使用单个输出编码流缓冲区 0: disable 1: enable
	SingleEsBufSize	n 路编码中单个 es 缓冲区的大小

参数		描述
CHN ATTR	ID	通道 ID
	MaxWidth	编码通道的最大宽度
	MaxHeight	编码通道的最大高度
	Width	图像宽度
	Height	图像高度
	C2GEn	是否启用 Color-to-gray Value range: {0, 1}
	BufSize	码流缓冲区大小 (unit: byte)
	ByFrame	是否按帧获取码流 Value range: {0, 1}
	GopMode	GOP 模式
	MaxStrCnt	码流缓冲区中帧的最大数量 Default value: 200

参数		描述
RefParam INFO	ID	通道 ID
	EnPred	是否开启预测模式选择 N(0): disable Y(1): enable
	Base	基本层的周期
	Enhance	增强层的周期
	RcnRefShareBuf	是否开启重建帧和参考帧共享缓冲区 0: disable 1: enable
Syntax INFO	ID	通道 ID
	Profile	视频编码器配置项设置

2.13 h265e

cat /proc/soph/h265e

查看 265 的编解码信息

```
[root@vitek]# cat /proc/soph/h265e
Module: [H265E] System Build Time [#1 SMP Tue Nov 21 13:44:14 CST 2023]
-----H265E PARAM-----
OnPack: 0 H265ESource: 0 PowerSaveEn: 0 MinBufMode: 0 BpPstgEn: 0 UserDataMaxLen: 3072
SingleBuf: 0 SingleBufSize: 0 RefRateType: 0
---CHN ATTR-----
ID: 0 MaxWidth: 3220 MaxHeight: 1880 Width: 3220 Height: 1880 CQEN: 0 BufSize: 0 ByFrame: 1 SopMode: NORMALP MaxStrCnt: 0
---RefParam INFO-----
ID: 0 EnPred: Y Base: 0 Enhance: 0 RcnRefShareBuf: 0
---Syntax INFO-----
ID: 0 Profile: Main
```

参数同 h264e

2.14 REGION

cat /proc/soph/rgn

```
[root@vitek]# cat /proc/soph/rgn
Module: [RGn], Build Time[#1 SMP Tue Nov 21 13:44:14 CST 2023]
-----REGION STATUS OF OVERLAY-----
Hdl Type Used Pifmt W H BgColor Phy Virt Stride Cnvsum Cmpr MaxNeedion
-----REGION CHN STATUS OF OVERLAY-----
Hdl Type Mod Dev Chn bShow X Y Layer
-----REGION STATUS OF COVER-----
Hdl Type Used
-----REGION CHN STATUS OF RECT COVER-----
Hdl Type Mod Dev Chn bShow Layer CoordType
X Y W H Color
-----REGION STATUS OF COVEREX-----
Hdl Type Used
-----REGION CHN STATUS OF RECT COVEREX-----
Hdl Type Mod Dev Chn bShow Layer
X Y W H Color
-----REGION STATUS OF OVERLAYEX-----
Hdl Type Used Pifmt W H BgColor Phy Virt Stride Cnvsum
-----REGION CHN STATUS OF OVERLAYEX-----
Hdl Type Mod Dev Chn bShow X Y Layer
```

【调试信息分析】

记录当前区域资源信息。

【参数说明】

参数		描述
REGION STATUS OF COVERCOVER	Hdl	COVER 的 Handle 号。
	Type	COVER 类型，值为 1。
	Used	该资源是否被占用。 · N：未占用。 · Y：占用。
REGION CHN STATUS OF RECT COVER	Hdl	COVER 的 Handle 号。
	Type	COVER 类型，值为 1。
	Mod	Attach 的模块。
	Dev	设备号。
	Chn	通道号。
	bShow	是否在该通道显示。 N：隐藏。 Y：显示。
	X	在该通道显示的起始 X 坐标。
	Y	在该通道显示的起始 Y 坐标。
	W	COVER 宽度（单位：像素）。
	H	COVER 高度（单位：像素）。
	Color	COVER 颜色。
	Layer	在该通道显示的层次。
	CoorType	ratio coordiante or abs coordinate
REGION STATUS OF COVEREX	Hdl	COVEREX 的 Handle 号。
	Type	COVEREX 类型，值为 2。
	Used	该资源是否被占用。 N：未占用。 Y：占用。
REGION CHN STATUS OF RECT COVEREX	Hdl	COVEREX 的 Handle 号。
	Type	COVEREX 类型，值为 2。
	Mod	Attach 的模块。
	Dev	设备号。
	Chn	通道号。
	bShow	是否在该通道显示。 N：隐藏。 Y：显示。
	X	在该通道显示的起始 X 坐标。
	Y	在该通道显示的起始 Y 坐标。
	W	COVERE X 宽度（单位：像素）。

下页继续

表 2.6 – 续上页

参数		描述
	H	COVERE X 高度 (单位: 像素)。
	Color	COVEREX 颜色。
	Layer	在该通道显示的层次。
REGION STATUS OF OVERLAYEX	Hdl	OVERLAYEX 的 Handle 号。
	Type	OVERLAYEX 类型, 值为 0。
	Used	该资源是否被占用。 N: 未占用。 Y: 占用。
	PiFmt	OVERLAYEX 像素格式, 参看 PIXEL_FORMAT_E。
	W	OVERLAYE X 宽度 (单位: 像素)。
	H	OVERLAYE X 高度 (单位: 像素)。
	BgColor	OVERLAYEX 背景色。
	Phy	OVERLAYE X 占用内存的物理地址。
	Virt	OVERLAYE X 占用内存的虚拟地址。
	Stride	OVERLAYEX 内存跨度 (单位: byte)。
	CnvsNum	OVERLAYEX 内存数目
REGION CHN STATUS OF OVERLAYEX	Hdl	OVERLAYEX 的 Handle 号。
	Type	OVERLAYEX 类型, 值为 0。
	Mod	Attach 的模块。
	Dev	设备号。
	Chn	通道号。
	bShow	是否在该通道显示。 N: 隐藏。 Y: 显示。
	X	在该通道显示的起始 X 坐标。
	Y	在该通道显示的起始 Y 坐标。
	Layer	在该通道显示的层次。

2.15 AI (audio input)

```
cat /proc/asound/cv186xadc/pcm0c/sub0/info
```

作用：获取 mic 录音通道声卡信息

适用情况：当 acap 失效无法录制音频，可查看 devices 是否被占用重要参数：

参数	描述
subdevices_count	子设备数量目前仅 device 0
subdevices_avail	空闲设备个数
hw_ptr	alsa 该值为 alsa 驱动写入指针位置
appl_ptr	读取数据者的指针位置

```
cat /proc/asound/cv186xadc/pcm0c/sub0/hw_params
```

作用：查看 adc 通道参数配置

适用情况：向客户获取 hw_params 时采用。

重要参数：

参数	描述
format	当前数据位宽
channels	当前声道数
rate	当前采样率
period_size	单次处理数据的大小（单位为：帧）
buffer_size	alsa 的 buffer 大小（单位为：帧）

2.16 AO (audio output)

同 AI

```
# cat /proc/asound/cv186xdac/pcm0p/sub0/info
```

```
# cat /proc/asound/cv186xdac/pcm0p/sub0/status
```

```
# cat /proc/asound/cv186xdac/pcm0p/sub0/hw_params
```

3 Dump Frame 定位问题模块

当视频画面出现异常时，可通过 dump frame 的方式确认异常发生的初始位置，再进入具体模块做进一步分析。

如果大家遇到问题，可以先从 dump frame 角度来看，是否可以用这个方式来找到是哪个模块出了问题，然后再找相关 owner 来处理。

3.1 sensor-raw

```
1 CVI_S32 CVI_VI_GetPipeFrame(VI_PIPE ViPipe, VIDEO_FRAME_INFO_S *pstFrameInfo,
  ↪ CVI_S32 s32MilliSec);
```

通过这个 vi 接口可以获取到 raw 图，运行 cvi_test，

输入 15 选择 "dump vi raw data"，

然后根据提示 "To get raw dump from dev(0~1): "，

输入 dev (0 表示 vi pipe0，从第 0 路 sensor dump 图像，1 表示 vi pipe1，从第 1 路 sensor dump 图像)

然后根据提示 "how many loops to do (1~60)"，输入 loops(表示要 dump 多少 frame)。

3.2 vi yuv

```
1 CVI_S32 CVI_VI_GetChnFrame(VI_PIPE ViPipe, VI_CHN ViChn, VIDEO_FRAME_INFO_S
  ↪ *pstFrameInfo, CVI_S32 s32MilliSec);
```

注意其中 ViPipe 固定为 0，配置 ViChn 以区分哪一个 sensor。

通过 vi 接口获取当前 frame，再自行保存为 yuv 文件，保存方式可参考接口 SAMPLE_COMM_FRAME_SaveToFile。

示例可参考 cvi_test case 2: dump vi yuv frame。

也可直接复制 CVI_VI_GetChnFrame 到 app 中使用。

3.3 vpss yuv

目前不支持 vpss grp dump yuv, 只能在 chn dump。

```
1 CVI_S32 CVI_VPSS_GetChnFrame(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, VIDEO_
  ↳FRAME_INFO_S *pstFrameInfo, CVI_S32 s32MilliSec);
```

通过该接口获取当前 frame, 保存 YUV 文件, 可参考 SAMPLE_COMM_FRAME_SaveToFile。

注意:

注意: sample 保存 YUV 是按照 Stride 每行保存, 使用工具打开可能会存在图像问题。每行按宽度保存不会出现该问题。

3.4 venc yuv

保存送给 venc YUV 数据。

```
1 CVI_S32 CVI_VENC_SendFrame(VENC_CHN VeChn, const VIDEO_FRAME_INFO_S
  ↳*pstFrame, CVI_S32 s32MilliSec);
```

在将 YUV 送给编码器时, 可以通过保存其接口参数 pstFrame->stVFrame 中的数据来保存 YUV, 保存方式可以参考 sample 接口 SAMPLE_COMM_FRAME_SaveToFile。

VENC 编码结果的 Stream 文件, 可以直接使用 PotPlayer 打开查看。

3.5 vdec yuv

```
1 CVI_S32 CVI_VDEC_GetFrame(VDEC_CHN VdChn, VIDEO_FRAME_INFO_S *pstFrameInfo,
  ↳CVI_S32 s32MilliSec);
```

参考 SAMPLE_COMM_FRAME_SaveToFile 的方式保存成 YUV 数据。

3.6 vo yuv

vo 作为显示的最后一个环节, 不支持 dump vo yuv。

3.7 Audio

Audio 定位 dump 数据方法：

表 3.1: Audio dump 数据

序号	命令	文件名	注释
1	touch /tmp/ain_record	/tmp/ain_record.pcm	从底层获取的最原始 pcm 数据
2	touch /tmp/dump_before_aec	/tmp/dump_before_aec.pcm	vqe 以前的 pcm
3	touch /tmp/dump_after_aec	/tmp/ain_record.pcm	vqe 后的 pcm
4	touch /tmp/dump_ai_Resin	/tmp/dump_ai_Resin.pcm	Ai 重采样前的 pcm
5	touch /tmp/dump_ai_Reskout	/tmp/dump_ai_Reskout.pcm	Ai 重采样后的 pcm
6	touch /tmp/dump_aenc_in	/tmp/dump_aenc_in.pcm	编码前的 pcm
7	touch /tmp/dump_aenc_out	/tmp/dump_aenc_out.pcm	编码后的 bitsteam
8	touch /tmp/dump_adec_in	/tmp/dump_adec_in.pcm	解码前的 bitsteam
9	touch /tmp/dump_adec_out	/tmp/dump_adec_out.pcm	解码后的 pcm
10	touch /tmp/dump_ao_Resin	/tmp/dump_ao_Resin.pcm	ao 重采样前的 pcm
11	touch /tmp/dump_ao_Reskout	/tmp/dump_ao_Reskout.pcm	ao 重采样后的 pcm
12	touch /tmp/dump_ao_output	/tmp/dump_ao_output.pcm	输出给底层最后一环节前的 pcm

3.7.1 音频输入模块 Ai

```
1 CVI_S32 CVI_AI_Enable(AUDIO_DEV AiDevId);
```

可通过该 ai 接口获取初始音频数据，在 CVI_AI_Enable->AudioInputThread->pcm_read dump 数据。

通过判断音频文件是否正常来确定底层 acap 是否正常

```
1 CVI_S32 CVI_AI_GetFrame(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_FRAME_S *pstFrm,
    ↪ AEC_FRAME_S *pstAecFrm, CVI_S32 s32MilliSec);
```

从 ai chn 中获取数据，数据存储地址 pstFrm->u64VirAddr[0]，获取字节数 pstFrm->u32Len* (ai 配置 channel) *2 (only support 16bit)

以下几个 API 获取数据单位均为 frame，dump 的字节数都按此计算

3.7.2 音频编码模块 AENC

```
1 CVI_S32 CVI_AENC_GetStream(AENC_CHN AeChn, AUDIO_STREAM_S *pstStream, CVI_S32_  
↪s32MilliSec);
```

该接口将编码后数据存储至 `pstStream->pStream` 中。

若使用 `ai bind aenc` 模式可通过对比 `ai` 抓取的数据和 `encode` 后数据来判断编码效果是否理想及 `ai->aenc` 过程中是否出错。

3.7.3 音频解码模块 ADEC

```
1 CVI_S32 CVI_ADEC_GetFrame(ADEC_CHN AdChn, AUDIO_FRAME_INFO_S *pstFrmInfo,_  
↪CVI_BOOL bBlock);
```

该 `ai` 接口将获取解码后数据，数据存储地址为 `pstFrmInfo->pstFrame->u64VirAddr[0]`

可通过查看 `tong` 音文件编码后的数据质量（可将该处 `dump` 的文件用音频分析软件观察波形，`tong` 音波形比较整齐，若出现杂音易观察）来测试编码效果

若使用 `adec bind ao` 模式，当 `ao` 播放异常时，使用该接口 `dump` 数据确定送入 `ao` 模块播放音频是否正常。

3.7.4 音频输出模块 AO

```
1 CVI_S32 CVI_AO_Enable(AUDIO_DEV AoDevId);
```

`touch /tmp/ao_outinput.pcm` 即可得到送入 `alsa` 播放的音频数据（即 `CVI_AO_Enable->AudioOutputThread->pcm_write` 前数据）。

以 `adec->ao` 为例，若从 `CVI_ADEC_GetFrame`, `dump` 数据正常 `ao` 播放异常，`ao_outinput.pcm` 文件的好坏决定问题出在 `sdk` 还是底层。

4 字节要求对齐

4.1 VB size 对齐

代码中主要是调用 COMMON_GetPicBufferConfig 函数来实现 buffer 的分配，而我们大部分对于宽高的对齐都可以在这个函数里面找到细节。

- 如果是 PIXEL_FORMAT_YUV_PLANAR_420/NV12/NV21，高要为偶数。
- stride 要根据 DEFAULT_ALIGN 对齐，对应到处理器的话，186AH 需要 64 字节对齐，因此不同处理器相同分辨率的
VB 大小可能不一样。
- 最终整体的 vb size，如果是 186AH，并且是 planar 的格式 (plane num 大于 1) 要再加上 $0x1000 * 3$
- frame 的起始地址，186AH 也要对应的 0x1000 对齐

4.2 VDEC 宽高对齐

具体可以参考 VDEC_GetPicBufferConfig 中的配置。

其中 Width 采用 64 字节对齐，Height 部分 H26x 采用 64 字节对齐，JPEG 采用 16 字节对齐

5 系统限制

5.1 VI

5.2 VPSS

CV186AH 处理器目前支持的图像最大宽高为 8192x8192，最小宽高为 16x16。
yuv420 宽高都要为偶数，yuv422 宽度要为偶数。

5.3 VCODEC

- encode:
 - 最大宽高：8192x4320
 - 最小宽高：256x128
- decode:
 - 最大宽高：8192x4320
 - 最小宽高：64x64
- JPEG:
 - 最大宽高：16384x16384
 - 最小宽高：64x64
- 不支持 progressive 类型图片解码；
- 不支持量化表精度为 16bit 的图片解码

5.4 REGION

region 不支持重叠

由于硬件的限制，绑定到每个 channel 的 region 不支持重叠，如果发生重叠，例如 region1 和 region0 发生重叠会打印：

RGN_HANDLE(1) is overlapped on another RGN_HANDLE(0).

5.5 AUDIO

- Ai / Ao:
 - 支持的通道: 1 2 4
 - 支持的采样率: 8kHz - 48kHz
 - 带宽: 16bit/32bit (只有 4 通道支持 32bit)
 - 回声消除: 仅仅支持 8kHz 和 16kHz 采样率以及双通道
- Aenc/Adec
 - 支持的采样率: 8kHz、16kHz、32kHz、44.1kHz、48kHz
 - 支持的编码标准: G.726, G.711u, G.711a, ADPCM, AAC

6 历史板子常见问题收集

也可参考 cv181x 上的内容

6.1 Sensor

6.1.1 I2C 如何确认？

1. sensor i2c 属性确认

- 检查 I2C bus id
- 检查 i2c slave addr
- 检查 sensor 寄存器的 addr/data 位宽 (8bit or 16bit)
 - 位宽配置错误的报错一般会报 time out 错误

```

00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
[ 474.411235] [1] i2c_designware 4000000.i2c: controller timed out
Unable to send data: Connection timed out
[ 475.434609] [1] i2c_designware 4000000.i2c: controller timed out
Unable to send data: Connection timed out
[ 476.457937] [1] i2c_designware 4000000.i2c: controller timed out
Unable to send data: Connection timed out
[ 477.481288] [1] i2c_designware 4000000.i2c: controller timed out
Unable to send data: Connection timed out
[ 478.504626] [1] i2c_designware 4000000.i2c: controller timed out
Unable to send data: Connection timed out
[ 479.528059] [1] i2c_designware 4000000.i2c: controller timed out
Unable to send data: Connection timed out
[ 480.551465] [1] i2c_designware 4000000.i2c: controller timed out
Unable to send data: Connection timed out
[ 481.574840] [1] i2c_designware 4000000.i2c: controller timed out
Unable to send data: Connection timed out
[ 482.598313] [1] i2c_designware 4000000.i2c: controller timed out
Unable to send data: Connection timed out
[ 483.621753] [1] i2c_designware 4000000.i2c: controller timed out
Unable to send data: Connection timed out
[ 484.645161] [1] i2c_designware 4000000.i2c: controller timed out
Unable to send data: Connection timed out
[ 485.668670] [1] i2c_designware 4000000.i2c: controller timed out
Unable to send data: Connection timed out
0x1b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

- 检查 hardware 是否正常

2. 确认 dts 中的 rst, pwn, mclk 引脚配置正确

3. 配置 mclk, reset

```

echo "snsr_on 0 1 1" > /proc/soph/mipi-rx //1 表示 37.125M, 2 表示 25M, 3 表示 27M
echo "snsr_on 1 1 1" > /proc/soph/mipi-rx //1 表示 37.125M, 2 表示 25M, 3 表示 27M

```

(下页继续)

(续上页)

```
echo "snsr_r 0 0" > /proc/soph/mipi-rx
echo "snsr_r 1 0" > /proc/soph/mipi-rx
```

4. 用 `i2cdetect -y -r N` 命令测试 i2c 能否检测到。N 表示 sensor 对应的 i2c 端口
5. 检查 power on 时序是否满足 spec 要求（示波器测量 mclk, i2c）

6.1.2 lane id 如何匹配？

注意我们要配置的 lane id 要以 sensor 为参照物来配置，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，索引号 2 表示 sensor lane 1，以此类推。land_id 数组的值表示的是 soc 的 MIPI-Rx 的 Lane ID，0 表示 MIPIRX1_PAD0，1 表示 MIPIRX1_PAD1，未使用的 lane 将 lane_id 配置成-1。

假设 sensor 和 soc 的 lane 接线如下图所示，比如下面的 sensor 的 lane id 配置就是 {3, 4, 2, 0, 1}。

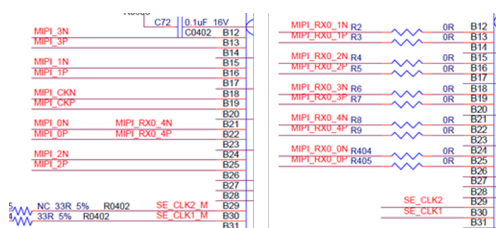
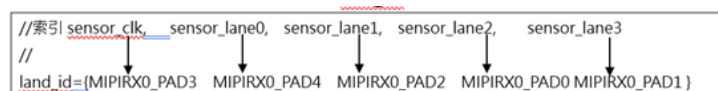


表 6.1: Audio dump 数据

sensor 管脚	MIPI Lane 管脚
MIPI_CK (index = 0)	MIPIRX0_3 (value = 0)
MIPI_0 (index = 1)	MIPIRX0_4 (value = 1)
MIPI_1 (index = 2)	MIPIRX0_2 (value = 2)
MIPI_2 (index = 3)	MIPIRX0_2 (value = 3)
MIPI_3 (index = 4)	MIPIRX0_2 (value = 4)



6.1.3 Sensor_cfg.ini 参数如何解读？

```
;section for source
[source]
;type = SOURCE_USER_FE
dev_num = 1
en_mode = 0
; section for sensor
[sensor]
; sensor name
name = GCORE_GC4653_MIPI_4M_30FPS_10BIT

bus_id = 1
sns_i2c_addr = 41

mipi_dev = 0

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

mclk_en = 1
mclk = 0

rst_pin = 69
rst_active= 0
```

- dev_num: 双目 sensor, dev_num 要配置成 2
- en_mode: 工作模式
- bus_id: sensor 使用的 i2c 端口
- sns_i2c_addr: sensor 的 i2c 设备地址
- mipi_dev: sensor 使用的 mipi-rx 编号
- lane_id: sensor 使用的 lane 线序
- pn_swap: lane 线序中的 P/N 极性配置
- mclk_en: 表示启用手动配置 mclk
- mclk: 使用哪一个 mclk, 0 表示 mclk0, 1 表示 mclk1
- rst_pin: 复位引脚 GPIO 号
- rst_active: 复位电平

6.1.4 Mipi-rx decode unknow 后如何调试？

- 确认 i2c 通路 (i2cdetect 能够扫出 sensor 地址)
- 确认 lane 线序正确
 - 如果 proc 中的 data lane 没有数据跳变, 并且伴随 CK_HS 为 0, 表示 clk lane 没找对 (请确认 clk lane)
 - 如果 proc 中的 data lane 有数据跳变, 伴随 CK_HS 为 1, 表示 clk lane 找对了, 已进入 hs 模式, 这时如果出现 ecc、crc 等错误, 这时表示 data lane 没配置对 (请确认 clk lane)

- 确认时序
 - 若前 2 点都确认正确, 但还 $CK_HS = 0$, data lane 没有数据跳变, 那么就可能是时序上未能满足进入 hs 的条件, 那么此时可以调整加大 hs-zero, hs-trail 就是为了拉长 detect period。
 - 若前 2 点都确认正确, $CK_HS = 1$, data lane 有数据跳变, 但是还任然有 ecc crc 等 err, 那么很有可能是 Hs-settle 的设定太大或是太小, 压到后面的 data。
- 确认 hw 是否损坏

6.1.5 fps、mipi clk、raw 格式、lane cnt、MACclk 之间的关系

fullines: 表述 1 帧总条 (行) 数。

曝光是以一条 H 的曝光时间为最小单位的, 所以一条 H 的曝光时间 $TH = 1/(fps * FullLines)$ 秒

有效曝光条数 = $inttime(\text{有效曝光时间}) / TH = inttime * fps * FullLines$

限制条件: $sexp + lexp < fullines$, 也就是长曝光的条数 + 短曝光的条数必须满足小于 1frame 的总 H 条数。

Mipi 速率相关:

mipi DPHY 协议有定义 MIPI 数据传输的规则, 最低 data rate 为 80M bps, 最高为 2500M bps。

Mipi clk 又称 data rate, 表示每条 lane 传输多少 bit 数据。

总数据量: $width * height * fps * pix_width$, 单位为 bit

pix_width : raw8/raw10/raw12

Data rate: 总数据量 / ($lane_num * 2$)

6.1.6 Mipi-rx decode unknow 后如何调试?

1. 确认 i2c 通路 (i2cdetect 能够扫出 sensor 地址)
2. 确认 lane 线序正确
 - a. 如果 proc 中的 data lane 没有数据跳变, 并且伴随 CK_HS 为 0, 表示 clk lane 没找对 (请确认 clk lane)
 - b. 如果 proc 中的 data lane 有数据跳变, 伴随 CK_HS 为 1, 表示 clk lane 找对了, 已进入 hs 模式, 这时如果出现 ecc、crc 等错误, 这时表示 data lane 没配置对 (请确认 data lane)
3. 确认时序
 - a. 若前 2 点都确认正确, 但还 $CK_HS = 0$, data lane 没有数据跳变, 那么就可能是时序上未能满足进入 hs 的条件, 那么此时可以调整加大 hs-zero, hs-trail 就是为了拉长 detect period。
 - b. 若前 2 点都确认正确, $CK_HS = 1$, data lane 有数据跳变, 但是还任然有 ecc crc 等 err, 那么很有可能是 Hs-settle 的设定太大或是太小, 压到后面的 data。

4. 确认 hw 是否损坏

6.2 VI

6.2.1 vi select timeout

当 vi 连续多次获取不到数据时，会出现此 log，并会自动打印 mipi-rx 和 vi_dbg 的 proc 信息。出现的原因有很多，一般先从数据流向上着手分析。

- 确认是否 sensor 端异常
 - 查看两次 vi_dbg 状态。
 - 若 VISofCnt 不增加，说明前端 sensor 没送数据，转 sensor 端分析。
 - 若 VICsiWidthGTCnt/VICsiWidthLSCnt/VICsiHeightGTCnt/VICsiHeightLSCnt 不为零，说明 sensor 送入的数据宽高值异常，此时 pre_raw 不会接收，转 sensor 端分析
- 确认是否 vi_qbuf 异常
 - 若打印” Can’ t acquire VB BLK for VI”，说明 VB buffer 不足，请检查对应的 vb pool 是否足够，或者是否被其他应用错误的占满。
 - 若打印” vi workq is full. drop new one” 或者” vi v4l2_qbuf error”，说明 driver 状态异常，请 owner 继续分析。
 - 连续查看两次 vi_dbg 状态
 - 若 VIPreraStatus 有某个模块连续为非 idle 状态，说明该模块可能 hang 住，要找 hw designer 介入分析。

6.2.2 FPS 设置

早期版本透过 CVI_ISP_SetAEFps 设置，后期版本修改为使用 ISP 统一的参数接口设定函数 CVI_ISP_SetPubAttr

参考 void callback_FPS(int fps)

6.2.3 画面黑白

- Sensor 本身是出彩图，画面黑白是由 ISP 效果决定。
- 早期版本默认 Sensor 0 出彩图，Sensor 1 出黑白图，后修改为默认全部出彩图。在 VI 初始化后，可透过 “CVI_ISP_SetMonoAttr” 接口自行配置。

6.2.4 drop frame due to gdc op blocked

VI Rotation/LDC 实际是由 gdc 硬件处理, vi thread 在处理 current frame 时, 假如发现 previous frame 在 gdc 内还没处理完, 就会有这句 error 打印并 drop current frame。

假设当前 vi framerate 为 25fps, 即 40ms 送一次中断, gdc 要保证 40ms 内处理完才不会报错。

6.2.5 DPCM 压缩模式

开启压缩模式后, ISP 内部 DMA 读写的数据减半, 以达到节省频宽的效果。

以 “cvi_test” 为例, 设置压缩模式的位置在

```
COMPRESS_MODE_E enCompressMode = COMPRESS_MODE_TILE;
```

主要是传递到 VI 的参数 “pstPipeAttr->enCompressMode”

```
CVI_S32 CVI_VI_CreatePipe(VI_PIPE ViPipe, const VI_PIPE_ATTR_S *pstPipeAttr)
```

确认 DPCM 是否有真正开启:

dmesg 或者 cat /mnt/data/log/kern, 看一下有没有这个 log

```
ISP_COMPRESS_ON(1)
```

注意:

1. 压缩模式下, vi dump raw 也为压缩格式, 需应用层自行解压缩后使用, 参考代码 ISP_GetRawBuffer。
2. 压缩模式对画质有一点点损失

6.3 VPSS

6.3.1 CVI_VPSS_GetChnFrame fail

CVI_VPSS_GetChnFrame 失败情况非常复杂, 受前端数据源和 VPSS 参数, 以及用户使用方式的影响。一般 debug 步骤如下:

- 第一步

检查代码写法, 确保对应通道属性的 u32Depth 大于 0, (注意 u32Depth 属性不能动态修改, 建议初始化参数就正确设置)

同时要确保超时时间设置的足够长。

- 第二步: 先确认前端 vi 或 vdec 是否有送数据

可以查看两次 vpss proc 信息, 看 RecvCnt 是否有增加。

还可以查看 vi proc 信息 (vi-vpss offline), 看 IntCnt 是否增加, 注意 IntCnt 增加并不能保证一定往 VPSS 送数据, 还需要保证 vi rotation/ldc 没有失败, vi 是否 bind(或 send)vpss。

- 第三步: 确认 HW 开始工作

查看 vpss proc 信息, 如果 StartFailCnt 一直增加, 有可能是 VB 获取不到导致, 这时就需要检查 VB 分配, 或者抓 mw log, 结合 log 分析, 如果出现 Can't acquire VB BLK for VPSS 则是 VB 拿不到,

可以分析 log 确定 vb 大致去向。如果 VB 分配不合理, 请加大 VB 大小或个数。

如果出现打印: Mod(VPSS) Grp(0) Chn(0), jobs wait(0) work(0) done(0)。

Work 为 0 表示 workq 中没有 buf, 此时也是 VB 拿不到, 一般在 online mode 经常出现。

- 第四步: 确认 VPSS 是否卡死。

如果前面步骤都没有问题, 这时所有 VPSS 都 CVI_VPSS_GetChnFrame 失败, 大概率是 VPSS 卡死, 卡死前会打印 select timeout, 可能是数据对齐问题或 HW 参数设置不对, 需要抓 mw log, 然后提供给 vpss module owner 分析。

6.3.2 CVI_VPSS_SendFrame fail

CVI_VPSS_SendFrame 失败一般有三种情况

- 第一: frame 的参数和 vpss grp 属性不匹配。

例如 frame 的 PixelFormat、宽高和 grp 属性不一样, log 中会有打印。

- 第二: frame 有问题。

例如 yuv420 下 frame 的宽高不为偶数。

如果 frame 不是来自 vi/vdec, 是由用户自己填充的, 一定要先把 frame 中参数初始化为 0。

- 第三: qbuf 失败, vpss grp 缓存队列 (waitq) 满了

如果 VPSS 创建很多个 grp, 造成 vpss 压力过大, VPSS 来不及处理 waitq 中的 frame, 这样就会出现 waitq is full. drop new one。如果确实是 grp 创建太多, 性能超出极限这种问题是不可避免的, 只能通过优化客户的数据流来解决。

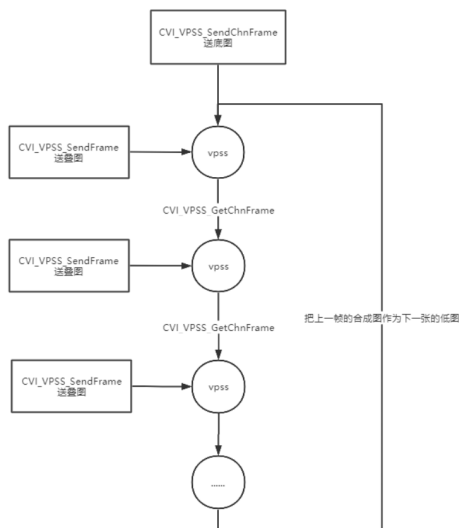
如果 VPSS 卡死也会出现 waitq full。

6.3.3 VPSS 的执行步骤

vpss grp: FRC -> crop

vpss chn: FRC -> crop -> scale -> overlayex -> mirror/flip -> LDC /rotation

6.3.4 VPSS 叠图介绍



叠图步骤：首先送一张底图到 vpss chn，再 send 一张需要叠加的图到该 VPSS grp，这时 VPSS 开始工作，VPSS HW 会写底图这块 buf，相当于在底图上作画，作画区域由 vpss chn 属性中 stAspectRatio 参数决定。

注意事项：由上面原理可知，底图的分辨率必须和 chn 的输出分辨率一样，叠图的时候 stAspectRatio 结构体参数中 bEnableBgColor 必须是 false，否则底图会覆盖背景色，一般在得到底图时会把 bEnableBgColor=true。

6.3.5 VPSS chn dump yuv 图像不正常

vpss chn frame 的 yuv 图像 stride 是 64 对齐的，如果保存 YUV 文件每行是按照 Stride 保存的，当图像宽度不是 64 的整数倍时，使用 7yuv 等工具查看会有可能异常。

以 520x56 分辨率举例：

520 宽度，64 对齐后是 576，所以 stride=576，当使用 7yuv 工具打开时，如果工具 align 不设置为 64，则图像显示完全错误，即使 align 为 64，也会出现条纹等现象，这是因为 7yuv 在 uv 的 stride 计算上和我们不相符，工具是按照 y stride 的一半 $576/2=288$ 计算，而我们实际 uv stride 是 $\text{align}((520/2), 64) = 320$ ，所以会出现图像异常。

这种情况可以通过 Python 脚本查看图像。或者在保存 YUV 文件时每行按照宽度保存。

6.3.6 VI-VPSS online mode

Online mode 注意事项：

1. Online 的 grp 必须要有一个 chn 是 Enable 状态的。
2. Chn 采用 ping-pong buffer，启动后需要两张 buf；vpss 产生一帧后，交给后端模块，然后在去 vb pool 拿一张 buf，所以正常工作一个 chn 通路需要 3 张 vb。
3. 一个 chn 最少需要 2 张 vb。vpss 产生一帧后，送给后端模块，这时再去 vb pool 拿 VB，如果该 VB pool 只有两张，这时会拿不到，由于使用 VB 订阅机制，这时就会注册一个订阅 VB 事件，只要后端模块能够在下一帧到来前（40ms 内）返还 VB，这时就会触发订阅回调拿到 VB，所以这种情况一个 chn 通路 2 张 vb 就可以满足。

6.4 VO

6.4.1 panel 接口

panel 是 VO 的子模块，VO 的 frame 要输出到相关接口的 panel 进行显示

目前已经应用或未来可能使用的 panel 接口有：

- MIPI-Tx
- LVDS
- mcu-i80
- MIPI-Tx -To- LVDS
- MIPI-Tx -To-HDMI
- MIPI-Tx -To-RGB
- BT
- BT-To-CVBS

6.4.2 如何点屏

详细内容请参考《屏幕对接指南》

6.5 REGION

6.5.1 一个 channel 最多支持绑定 8 个 region

假如某个 channel 已经绑定 0~7 8 个 region，如果绑定第 9 个会打印如下：

```
RGN_HANDLE(8) rgn's count is full
```

6.5.2 更新 region 位置、大小报 overlap 错误

通过 CVI_RGN_GetDisplayAttr/CVI_RGN_SetDisplayAttr 更新某个 region 的位置、大小，肉眼上看与其他 region 未发生重叠，实际 CVI_RGN_SetDisplayAttr 函数返回失败，并打印 overlap 信息，可能的一种情况是，某一 region 局部区域透明，即将要更新的 region 与该 region 的透明区域发生重叠

6.5.3 叠 bmp 图片的时候，需确保 region 区域大于 bmp 图片区域

6.6 VB/SYS

6.6.1 crate pool fail

确认传入是否符合规范，BlkSize 和 BlkCnt 不能为 0，BlkCnt 不能大于 128。

确认已有的 vb pool 是否达到上限 512。

确认剩余 ion 是否够用。

6.6.2 get vb block fail

```
cat /proc/soph/vb
```

确认 vb buf 是否够用

6.6.3 release vb block fail

VB release 仅是把 use_cnt 计数减一，当 usr_cnt 为一时候才做真正 release。

6.6.4 VB 获取

各个模块 (vi/vpss/gdc) 在工作时需要向 VB pool 公共缓存池中拿 buf，根据不同分辨率，不同格式，其所需要的 size 不一样。

从 VB pool 拿 VB 的原则是：老版本：尽量拿最合适大小的 pool，即拿大于等于 size，并且最小的那个 pool，如果这个 pool 剩余个数为 0，依次往上取。

新版本：拿最合适大小的 pool，即拿大于等于 size，并且最小的那个 pool，如果这个 pool 没有剩余，这时会拿不到 VB。这是基于低内存情况下做出的改动，保证不让越界去抢占大块 VB 的情况发生。

Vb 订阅机制：新版本 vpss online 下，一开始每个 chn 需要 2 张 vb，如果拿不到 vb 会订阅一张，当后端有模块释放 VB 时，就会产生订阅回调，回调中会去拿 VB，qbuf 到 driver。

vpss chn 还可以通过 API (CVI_VPSS_AttachVbPool) 到指定目标 pool 中去拿 VB。

6.6.5 VB 初始化时怎么分配 VB 大小和个数？

VB pool 大小计算：调用接口 COMMON_GetPicBufferSize 返回需要的 size，宽高调换时 size 不一样，比如 1080x1920 比 1920x1080 需要的 size 大。

VB pool 个数计算：vi chn：workq 固定 2 张，如果有 getchnframe 需要 +1，如果有设置 chn depth=n，需要再加 n，如果有做 rotation 或 ldc 再 +1；online mode 请忽略。Vpss input：固定 1 张，online mode 请忽略。

Vpss chn：offline 1 张，online 2 张；如果有 getchnframe 需要 +1，如果有设置 chn depth=n，需要再加 n，如果有做 rotation 或 ldc 再 +1。

Vo chn：workq 固定 2 张，waitq 由用户调用 CVI_VO_SetDisplayBufLen 指定，一般是 2，如果有做 rotation 再 +1。

Venc chn：venc 不会主动拿 VB，但是会缓存 VPSS 送过来的 VB，如果处理过慢，需要 +1。

6.7 Audio

audio debug

1. 确定 sdk 版本，确定客户 flow(例 Ai->Aenc) 注意：版本问题需注意 libaacenc2.ao 和 libaacdec2.so 编译：将 mw/modules/audio/src/algo/fdk_aac/fdkaac_tar_gz 移动到 mw/sample/audio/aac_sample，解压后编译即可
2. 先让客户测试 sample 主要 sample case：sample_audio 1(Ai->Ao)、sample_audio 2(Ai->Aenc)、sample_audio 3 (Adec->Ao) 若想自测 sample 内有 sample_rate、channel、是否使用 vqe、是否开启 resample、是否使用 bind 模式，参数需要选择，故测试前先找客户确认。

3. Audio 问题可分为以下几类 log 报错、杂音问题，录制无效，播音无效。
4. 使用 cooleedit，观察音频文件波形

6.7.1 Log 报错

1. Ai log 出现 chn x overrun 表示用户线程取数据太慢。查线程是否有休眠或耗时太长的代码。
2. Ao log 出现 underrun, 表示用户线程送数据太慢或者没有数据发送，开始或者结束时出现是正常的，运行过程中出现要查明原因

6.7.2 杂音

杂音又可分为录制到杂音及播放出现杂音（只要数据异常均可套用以下操作来进行初步定位）

Ai->Aenc:

第一步：Touch /tmp/ain_record 从底层获取最原始数据，只要录制出现问题，首先需要执行此操作。若文件异常，建议转交 mantis，音频文件正常再执行后续操作第二步：dump 以下数据分析

```
touch /tmp/dump_before_vqe
```

```
touch /tmp/dump_after_vqe
```

```
touch /tmp/ai_beforerres
```

```
touch /tmp/ai_afterres
```

```
touch /tmp/dump_aenc_in
```

```
touch /tmp/dump_aenc_out
```

Adec->Ao:

第一步：先检查进入解码模块前的原始数据是否正常

第二步：建议先 touch /tmp/dump_ao_output，判断输出给底层音频数据是否正常。若正常再根据通过以下方法抓的数据，判断是从编码还是重采样导致数据异常。

```
touch /tmp/dump_adec_in
```

```
touch /tmp/dump_adec_out
```

```
touch /tmp/dump_adec_out_res0
```

```
touch /tmp/dump_adec_out_res1
```

6.7.3 播音无效

若播放无声第一步：确认是否使能 speak

第二步：touch /tmp/dump_ao_output 如若异常可进入 mw code cvi_ao_internal.c -> AudioOutputThread 查看从 chn 中拿到的数据是否正常。

第三步：使用 tinyplay 播放若想确认 alsa 层以下的 ao 播放是否正常也可使用，如若异常则需要确认喇叭是否接 L 声道。

6.7.4 声音断断续续

1. 参考 2.8 节的方法 dump 声音
2. 确认 log 是不是有 underrun/overrun

6.8 VENC

6.8.1 SendFrame 报错 ERR_BUSY

1. 查看是否将 CVI_SYS_Bind 和 CVI_VENC_SendFrame 同时使用

6.8.2 SendFrame 报错 ERR_NOBU

1. 查看是否没有及时 ReleaseStream，导致 FrameBuffer 一直占用

6.8.3 CVI_VENC_GetStream 卡住获取不到 stream 码流

1. 确认是不是 bind mode
2. 如果是 bind mode，使用 ps -e | grep venc* 确认对应的编码线程状态
3. unbind mode 时请确认是否有正常发送 frame 做编码

6.9 VDEC

6.9.1 sample_vdec 使用方式

参考命令行

6.9.2 解码花屏问题

1. 在 CVI_VDEC_SendStream 前将 channel bitstream 写入文件
2. 使用 videoeye(264) 或 hevc analyzer 工具分析 (265)
3. 将 bitstream 发 RD 分析

6.10 Others

6.10.1 系统哪些模块可以做旋转

Sensor: 仅能利用 sensor flip + mirror 达到旋转 180° 的效果, 不占系统资源

VI/VPSS/VO: 通过 GDC 做旋转, 可支持 0/90/180/260

6.10.2 wdr/linear 切换的两种方式及区别

1. sensor 切换方式

Sensor 驱动包含有 wdr/linear mode 两套不同的寄存器设定, 初始化阶段会依据 sensor type 区分, 例如 SONY_IMX307_MIPI_2M_30FPS_12BIT/SONY_IMX307_MIPI_2M_30FPS_12BIT_WDR2TO1 示例可参考 cvi_test case 112。

6.10.3 isp 切换方式

在 sensor wdr mode 基础上, 关闭 ISP fswdr 功能, 达到线性输出。

区别: Sensor 切换方式是在 sensor 层面做完整模式切换, ISP 也会依据 sensor type 加载对应的 PQ bin 文件, 画面效果最完整。缺点是代码流程相对复杂, 需要把 VI 模块全部重置。

ISP 切换模式是关闭了宽动态下的长短帧合成只取长曝帧输出, 一定程度上是伪线性模式, 优点是可以动态切换, 不需要把 VI 重置。

6.10.4 IPC 公版-配置 ini 文件

第一步: 参考 IPC 公版的 README.md 编译 IPC, 在编译生成的 install 目录下会包含一个将 ini 文件转换成 bin 文件的可执行程序 ini2bin_board, 将此文件拷贝到端

第二步: 在 IPC 目录下, 保存有已经配置好的 ini 文件, 根据硬件需求, 将 config_access_entry.ini 与 config_media_comm_xxx.ini 拷贝到板端, 与步骤一的 ini2bin_board 放到同一级目录下。

第三步: 确保 config_access_entry.ini 里的 media_commom 下的 filename 与另一个 ini 文件的文件名相同。

第四步：如果要对参数进行修改，需要修改 `config_media_comm_xxx.ini`，例如对 sensor 的配置，如下图，根据实际情况修改 sensor 类型 (`sns_type`)，是否开启 wdr mode, laneid 等，需要注意的是，由于 `sns_type` 不能使用枚举，所以需要查找 sensor 对应的整型填进去

第五步：运行 `ini2bin_board`，此时会在当前目录下生成 `app_cfg_def.bin`，将此文件拷贝到 `sample_ipc` 同一级目录下，再运行 `sample_ipc` 即可

6.11 工具类

参考 mars 和双系统的 debug 文档，内容相同