



CV184X AliOS 编译使用手册

Version: 1.0

Release date: 2025-06-24

©2025 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	编译环境	3
2.1	安装 python 及 pip	3
2.2	安装 YOCTOOL 工具	3
2.3	SDK 文件说明	4
2.4	快速模式和正常启动模式	5
2.5	如何编译 AliOS	5
2.6	镜像说明	6
3	运行说明	7
4	Alios 指令集使用	9
4.1	系统信息	9
4.1.1	help	9
4.1.2	reboot	9
4.1.3	ccs	9
4.1.4	sysver	10
4.1.5	time	10
4.1.6	msleep	10
4.1.7	f	10
4.1.8	devname	10
4.1.9	err2cli	11
4.2	进程信息	11
4.2.1	多媒体进程	11
4.2.1.1	proc_vb	11
4.2.1.2	proc_sys	11
4.2.1.3	proc_vi_dbg	11
4.2.1.4	proc_vi	12
4.2.1.5	proc_gdc	13
4.2.1.6	proc_mipi_rx	13
4.2.1.7	proc_vo	13
4.2.1.8	proc_rgn	14
4.2.1.9	proc_codec	14
4.2.1.10	proc_vdec	14
4.2.1.11	proc_rc	14
4.2.1.12	proc_jpege	15
4.2.1.13	proc_h264e	15
4.2.1.14	proc_h265e	15
4.2.1.15	proc_venc	15
4.2.1.16	testMedia_switch_pipeline	16

4.2.1.17	testMedia_video_Deinit	16
4.2.1.18	testMedia_video_init	16
4.2.1.19	dump_isp_pqgram	16
4.2.1.20	proc_log	16
4.2.2	ipcm 进程	17
4.2.2.1	ipcm_dbg	17
4.2.2.2	ipcm_dbg pool_info	18
4.2.2.3	ipcm_dbg log0_on	18
4.2.2.4	ipcm_dbg log0_off	18
4.2.2.5	ipcm_dbg logall_on	18
4.2.2.6	ipcm_dbg logall_off	18
4.2.3	进程整体信息	18
4.2.3.1	tasklist	18
4.2.3.2	taskbt	19
4.2.3.3	taskbtn	20
4.2.4	CPU 信息	20
4.2.4.1	cpuusage	20
4.3	内存信息	21
4.3.1	查看内存信息	21
4.3.1.1	p	21
4.3.1.2	p_offcheck	22
4.3.2	修改内存信息	22
4.3.2.1	m	22
4.3.3	内存调试信息	23
4.3.3.1	ddumpsys	23
4.3.3.2	dumpsys mm	23
4.3.3.3	dumpsys mm_info	23
4.3.3.4	dumpsys mm_info_resv	24
4.3.3.5	mmlk	25
4.4	调试信息	25
4.4.1	debug	25

修订记录

Revision	Date	Description
v1.0	2025/06/24	Initial

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 编译环境

2.1 安装 python 及 pip

1. 安装 Python3
2. 下载 PIP 脚本
3. 安装 pip, 执行如下命令

```
python get-pip.py
```

4. 安装 python 的 serial 和 usb 驱动模块, 执行如下命令

```
pip3 install pyserial pyusb
```

5. 安装 yaml 模块, 执行如下命令

```
pip3 install pyyaml
```

注解: 如果上面 pyyaml 安装失败, 可以使用下面命令安装

```
pip3 install pyyaml -i http://pypi.douban.com/simple --trusted-host pypi.douban.com
```

注意: PS: ubuntu 版本推荐高于 20.0.4

2.2 安装 YOCTOOL 工具

1. Linux 环境中使用指令安装 yoctoool

```
sudo pip install yoctools -U
```

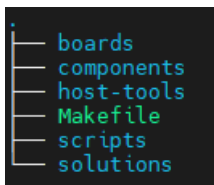
2. 使用 yoc -V 查看对应版本, 确保版本处于或者高于 2.0.25

```
jingjing.tang@cvitek:jingjing.tang $ yoc -V  
2.0.55
```

3. 输入 `product version` 查看 `product` 的版本，确保 `product` 安装成功

```
jingjing.tang@cvitek:jingjing.tang $ product version
v1.0.45
Mar 17 2023,14:16:38
```

2.3 SDK 文件说明



1. 顶层目录中各个文件夹说明：

- `boards`: 存放板卡相关配置
- `components`: 存放各种类型组件，例如 `aos` 内核，`cli` 命令行，网络组件，以及 `CSI` 驱动
- `host-tools`: 为交叉编译工具链，AliOS 使用玄铁交叉编译工具链 `Xuantie-900-gcc-elf-newlib-x86_64-V2.6.1`
- `solutions`: 为解决方案目录，主要存放业务逻辑，运行 `bin` 档等，可支持快速启动和正常启动两种解决方案。

2. `solutions` 下各目录作用说明：

- `application` 主要是应用相关部分
- `customization` 客户应用相关以及不同产品形态配置，其下各个主要文件说明：

<code>custom_sysparam.c</code>	<code>vb</code> 配置
<code>custom_viparam.c</code>	<code>vi</code> 配置 <code>sensor</code> 相关都放置在这里
<code>custom_voparam.c</code>	<code>vo</code> 配置
<code>custom_vpssparam.c</code>	主要是 <code>vpss</code> 配置
<code>custom_vencparam.c</code>	主要是 <code>venc</code> 的配置
<code>custom_moduleparam.c</code>	主要是各模块的通用配置
<code>custom_platform.c</code>	平台相关配置，如 <code>pinmux</code> 设定， <code>GPIO</code> 拉高等操作
<code>custom_event.c</code>	系统/模块事件相关配置

- `py_tool`: `python` 的工具
- `script`: 编译的脚本

3. `package.yaml` 文件说明

https://help.aliyun.com/document_detail/308617.html

`package.yaml` 主要用于定义和开启功能宏比如选择不同的 `Sensor`，以及需要依赖的组件选择

2.4 快速模式和正常启动模式

2.5 如何编译 AliOS

- 设定环境变量 (以 cv1842hp_wevb_0014a_spinor 为例)

```
$ source build/envsetup_soc.sh
```

Usage:

(1) menuconfig - Use menu to configure your board.

ex: \$ menuconfig

(2) defconfig \$CHIP_ARCH - List EVB boards(\$BOARD) by CHIP_ARCH.

** cv184x ** -> ['cv184x', 'cv1841c', 'cv1842cp', 'cv1842hp', 'cv1843hp']

ex: \$ defconfig cv184x

(3) defconfig \$BOARD - Choose EVB board settings.

ex: \$ defconfig cv1842hp_wevb_0014a_spinor

ex: \$ defconfig cv1842cp_wevb_0015a_spinand

- 选定 EVB cv1842hp_wevb_0014a_spinor

```
$ defconfig cv1842hp_wevb_0014a_spinor
```

```
==== Environment Variables =====
```

```
PROJECT: cv1842hp_wevb_0014a_spinor, DDR_CFG=ddr3_2133_x16
```

```
CHIP_ARCH: CV184X, DEBUG=0
```

```
SDK_VERSION: musl_arm, RPC=0
```

```
ATF options: ATF_KEY_SEL=default, BL32=1
```

```
Linux source folder:linux_5.10, Uboot source folder: u-boot-2021.10
```

```
ENABLE_DUAL_OS: y
```

```
CROSS_COMPILE_PREFIX: arm-none-linux-musleabihf-
```

```
ENABLE_BOOTLOGO: 0
```

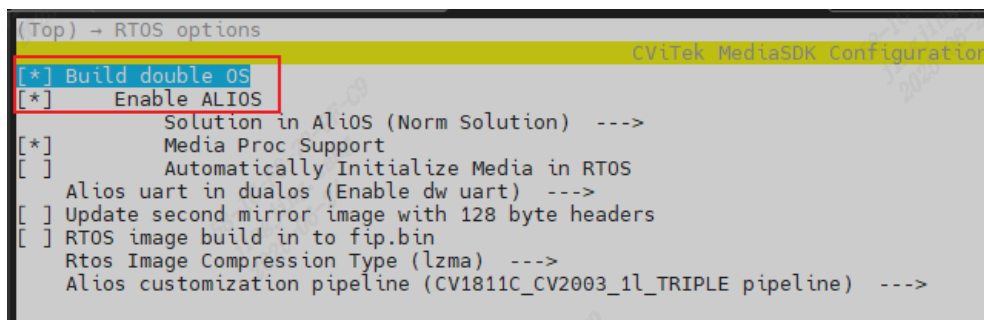
```
Flash layout xml: build/boards/cv184x/cv1842hp_wevb_0014a_spinor/partition/partition_spinor.xml
```

```
Sensor tuning bin: cvsens_cv2003
```

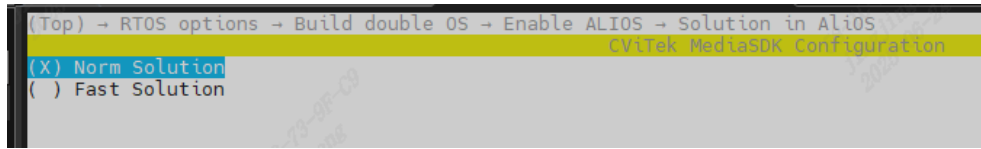
```
Output path: install/soc_cv1842hp_wevb_0014a_spinor
```

- 双系统配置

通过 menuconfig 命令, 勾选对应配置, 即可编译 AliOS 镜像。



默认是正常启动模式。可以通过勾选对应配置, 在快速启动模式和正常模式之间切换。



· 编译 AliOS

AliOS 镜像文件被打包到启动镜像中通常通过一条命令。

```
$ build_uboot
Run build_uboot() function
Run _link_uboot_logo() function
.....
[TARGET] alios-depends
.....
[TARGET] alios-build
.....
INFO: raw2cing small part size
INFO: size:b5c42, offset:200000, part_sz:100000, crc:38896f66
INFO: Packing yoc.bin done!
```

编译完成后，生成的 AliOS 镜像文件 yoc.bin 将被拷贝到 install 目录中。

2.6 镜像说明

在/solutions/normboot 的生成文件中：

1. yoc.elf: 完整的可执行文件等，可以被 objdump 或 readelf 工具解析
2. yoc.bin: 运行程序 bin 档案，烧录到 Flash 的小核镜像文件
3. yoc.asm: 反汇编文件
4. yoc.map: 内存映射文件
5. yoc_sdk/lib/*.a: 静态库
6. generated: AliOS 系统的固件镜像文件结构，文件结构描述如下：
 - boot0: 一级 Bootloader (FSBL) 的二进制文件
 - boot: 二级 Bootloader 的二进制文件 (如 U-Boot)
 - config.yaml: 分区表和固件配置的核心文件
 - imtb: 镜像表 (Image Table) 二进制文件
 - imtb.hex: 镜像表的 HEX 格式
 - prim: 主系统镜像 (yoc.bin 的别名)
 - prim.hex: 主系统镜像的 HEX 格式

3 运行说明

1. 烧录完毕后关闭电源等待 1-2S 钟，存在电容放电需要等释放完毕
2. 上电有如下打印 ###YOC### 证明内核启动完毕

```
(cli-uart)# C.SCS/0/0.USBI.BS/NOR.PS.PE.BS.BE.J. W:  
FSBL time: fsbl_exit:82 fsbl_all_time:45  
##cur_ms:85  
j 0x80040000  
##cur_ms:210  
###YoC##[Nov 1 2022,16:00:21]  
spinor: ID = ef 40 18  
ViPipe:0,===GC02M1 1200P 30fps 10bit LINEAR Init OK!===  
  
get hw version 0 !!!  
[VO-ERR] vo.c:1907:vo_irq_handler(): disp bw failed at frame#1  
retinaface_mnet0.25_342_608.cvmodel place in SD care  
  
APP_WifiInit wifi_hi3861l_register  
[error]func oal_sdio detectcard to core line 502 ret 1812 !!!  
fail to detect sdio card, ret=1812  
regeste_sdio_drv:: oal_sdio_func_probe failed:-lhcc_host_init:: oal_sdio_init fail!APP_WifiInit netmge_dev_wifi_init  
Init for MIPI-Driver-ILI9488-320x480  
[ 0.584]<I>app app_main.c[59]: app start.....  
  
APP_CustomEventStart 12.  
waiting for connect...  
(cli-uart)#
```

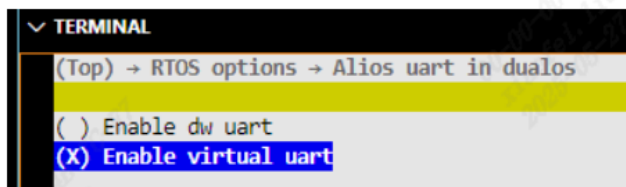
1. 输入串口回车可以正常输入输出，并且有 cli_uart 打印即运行正常

```
Init for MIPI-Driver-ILI9488-320x480  
[ 0.584]<I>app app_main.c[59]: app start.....  
  
APP_CustomEventStart 12.  
waiting for connect...  
(cli-uart)#
```

cli_uart

1. alios 虚拟串口使用需要在编译配置一下两个地方

alios中打开虚拟串口后会编译virtual下面的串口工具而不会编译dw目录下的实体串口驱动



```
cvl_alios > solutions > normboot > package_yaml > package.yaml.turnkey
178 def_config:
188 CONFIG_APP_CX_CLOUD_SUPPORT: 0
189 CONFIG_APP_VENC_SUPPORT: 1
190 CONFIG_APP_GUI_SUPPORT: 0
191 CONFIG_APP_SENSOR_IR_USE: 0
192 CONFIG_DEBUG_HOSTMCU_EMU_SUPPORT: 0
193 CONFIG_APP_AI_SUPPORT: 0
194 # UART_MODE_SYNC: 1
195 CONFIG_BOARD_CV181XC: 1
```

1. 小核 pc 值查看, 可以观察 pc 值在变化:devmem 0x1901070

4 Alios 指令集使用

4.1 系统信息

4.1.1 help

该命令将打印 alios 所有命令列表并简要描述每个命令的作用。

```
--> help
--> ===== AliOS Things Command List =====
--> help      : print this
--> reboot    : reboot system
--> proc_codec : codec info
--> proc_rc    : rc info
--> proc_jpege : jpeg encode info
--> proc_h264e : h264 encode info
--> proc_h265e : h265 encode info
--> proc_venc  : venc info
--> proc_vdec  : vdec info
--> proc_ai    : audio info
--> proc_ao    : ao info
--> proc_mipi_rx : proc mipi rx
```

4.1.2 reboot

重启 alios 系统，Linux 系统也会随之重启。

4.1.3 ccs

console cli console show，用于显示命令台界面内容。

```
--> ccs
--> ----- console show -----
--> id   name   att
--> 1    cli-uart 0  c
```

4.1.4 sysver

显示系统 kernel 版本。

```
--> sysver  
--> kernel version :12000
```

4.1.5 time

显示系统启动运动到现在的毫秒数。

```
--> time  
--> UP time 263888 ms
```

4.1.6 msleep

以毫秒时间休眠。系统休眠 2000ms 后重启。

```
--> msleep 2000  
--> #系统休眠2000ms后重启
```

4.1.7 f

表示运行一个函数（f 后需要跟函数名）。

```
--> f  
--> para error
```

4.1.8 devname

打印板端设备的名称。该设备是 cv181xc 系列的 evb 板。

```
--> devname  
--> device name: cv181xc_evb
```

4.1.9 err2cli

系统从 panic 状态转换到控制台。

```
--> err2cli 0
--> set panic_runto_cli flag:0x0

--> err2cli 1
--> set panic_runto_cli flag:0x21314916
```

4.2 进程信息

4.2.1 多媒体进程

4.2.1.1 proc_vb

该指令用于记录当前 VB 模块的 buffer 使用情况 (VB: Video Block)。

```
MaxPoolCnt(R 32), MaxBlkCntR( 32)
] vb.c:488:vb_get_pool_info(): NULL VB HANDLE
vb_pool has not initd yet
```

4.2.1.2 proc_sys

该指令用来记录当前 SYS 模块的使用情况。

```
(cli-uart)# proc_sys
-----BIND RELATION TABLE-----
1stMod  1stDev  1stChn   2ndMod   2ndDev   2ndChn   3rdMod   3rdDev   3rdChn
-----
```

4.2.1.3 proc_vi_dbg

该指令用来打印 vi 进程的 debug 信息。

```
(cli-uart)# proc_vi_dbg
```

4.2.1.4 proc_vi

该指令记录当前视频输入设备及信道的属性配置以及状态信息。

```
cli-uart1# proc_vi
```

-----MODULE PARAM-----									
DetectErrFrame		DropErrFrame							
0		0							
-----VI MODE-----									
DevID	PrerawFE	PrerawBE		Postraw		Scaler			
-----VI DEV ATTR1-----									
DevID	DevEn	BindPipe	Width	Height	IntfM	WkM	ScanM		
-----VI DEV ATTR2-----									
DevID	AD0	AD1	AD2	AD3	Seq	DataType	WDRMode		
-----VI BIND ATTR-----									
DevID	PipeNum	PipeId							
-----VI DEV TIMING ATTR-----									
DevID	DevTimingEn	DevFrmRate		DevWidth		DevHeight			
-----VI CHN ATTR1-----									
DevID	ChnID	Width	Height	Mirror	Flip	SrcRate	DstRate	PixFmt	VideoFmt
-----VI CHN ATTR2-----									
DevID	ChnID	CompressMode	Depth	Align					
-----VI CHN OUTPUT RESOLUTION-----									
DevID	ChnID	Mirror	Flip	Width	Height	PixFmt	VideoFmt	CompressMode	FrameRate
-----VI CHN ROTATE INFO-----									
DevID	ChnID	Rotate							
-----VI CHN EARLY INTERRUPT INFO-----									
DevID	ChnID	Enable	LineCnt						
-----VI CHN CROP INFO-----									
DevID	ChnID	CropEn	CoorType	CoorX	CoorY	Width	Height	TrimX	TrimY
						TrimWid	TrimHgt		
-----VI CHN STATUS-----									
DevID	ChnID	Enable	FrameRate	IntCnt	RecvPic	LostFrame	VbFail	Width	Height

模块参数:

- VI MODE (VI 模式)
- VI DEV ATTR1 (VI 设备特征)
- VI DEV ATTR2 (VI 设备特征)
- VI BIND ATTR (VI 关联特征)
- VI DEV TIMING ATTR (VI 时间特征)
- VI CHN ATTR1 (VI 通道特征)
- VI CHN OUTPUT RESOLUTION (VI 通道输出分辨率)
- VI CHN ROTATE INFO (VI 通道旋转信息)
- VI CHN EARLY INTERRUPT INFO (VI 通道中断信息)
- VI CHN CROP INFO (VI 通道裁剪信息)
- VI CHN STATUS (VI 通道状态)

4.2.1.5 proc_gdc

该指令用于记录 GDC 模块最近完成的若干任务、最近耗时最大的任务及历史累计信息。

```
(cli-uart)# proc_gdc
Module: [GDC]
-----RECENT JOB INFO-----
SeqNo  ModName  TaskNum  State  InSize(pixel)  OutSize(pixel)  CostTime(us)  HwTime(us)
-----MAX WASTE TIME JOB INFO-----
ModName  TaskNum  State  InSize(pixel)  OutSize(pixel)  CostTime(us)  HwTime(us)
-----GDC JOB STATUS-----
Success  Fail  Cancel  BegInNum  BusyNum  ProcInNum
0       0       0       0         0         0         0
-----GDC TASK STATUS-----
Success  Fail  Cancel  BusyNum
0       0       0       0
-----GDC INT STATUS-----
InthNum  IntTm(us)  HwProcTm(us)
0         0         0
-----GDC CALL CORRECTION STATUS-----
TaskSuc  TaskFail  EndSuc  EndFail  CntCnt
0         0         0         0         0
```

4.2.1.6 proc_mipi_rx

该指令用于显示 mipi_rx 进程的信息。

```
(cli-uart)# proc_mipi_rx
-----Module: [MIPI_RX]-----
-----Combo DEV ATTR-----
-----MIPI info-----
```

4.2.1.7 proc_vo

该指令用于记录当前 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。
用于动态获取 VO 的使用状态以调试测试。

```
(cli-uart)# proc_vo
Module: [VO]
-----DEVICE CONFIG-----
DevID  DevEn  IntfType  IntfSync  BkClr  DevFrt
# 0    N     Unknown  PAL       0      0
-----VIDEO LAYER STATUS 1-----
LayerId  VideoEn  PixFmt  ImgW  ImgH  DispX  DispY  DispV  DispH  DispFrt
# 0      N     RGB_888  0     0     0     0     0     0     0
-----VIDEO LAYER STATUS 2 (continue)-----
LayerId  DevId  EnChNum  Luxa  Cont  Hua  Sata  BufLen
# 0      0     0       0    0    0    0    0
-----CHN BASIC INFO-----
LayerId  ChnId  ChnEn  Prio  ChnX  ChnY  ChnW  ChnH  RotAngle
# 0      0     Y     0    0    0    0    0    0
-----CHN PLAY INFO-----
LayerId  ChnId  Show  Pause  Threshd  ChnFrt  ChnGap(us)  DispPts  PreDonePts
# 0      0     N     N     0       0       0         0       0
```


4.2.1.8 proc_rgn

该指令用于记录当前区域资源信息。

```
(cli-uart)# proc_rgn
Module: [RGN]

-----REGION STATUS OF OVERLAY-----
HdL   Type   Used   PIPmc   W   H   BpColor   Phy   Virt
Stride  CnvNum  Cnpr  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   PIPmc   W   H   BpColor   Phy   Virt
Stride  CnvNum
-----REGION CHN STATUS OF OVERLAYEX-----
HdL   Type   Mod   Dev   Chn   bShow   X   Y   Layer
```

4.2.1.9 proc_codec

该指令用于显示编码器与解码器的信息。

```
(cli-uart)# proc_codec
Module: [CodecInst] System Build Time [Jul 27 2023-10:08:45]
```

4.2.1.10 proc_vdec

该指令用于显示视频解码器的信息。

```
(cli-uart)# proc_vdec
Module: [VDEC]
```

4.2.1.11 proc_rc

该指令用于显示当前编码码率控制信息。

```
(cli-uart)# proc_rc
Module: [RC]
```

4.2.1.12 proc_jpege

该指令用于显示 JPEG 编码过程中各通道编码属性状态。

```
(cli-uart)# proc_jpege  
Module: [JPEGE]
```

4.2.1.13 proc_h264e

该指令用于显示 H.264 视频编码属性配置以及状态信息。

```
(cli-uart)# proc_h264e  
Module: [H264E]
```

4.2.1.14 proc_h265e

该指令用于显示 H.265 视频编码属性配置以及状态信息。

```
(cli-uart)# proc_h265e  
Module: [H265E]
```

4.2.1.15 proc_venc

该指令用于显示当前视频编码属性配置以及状态信息。

```
(cli-uart)# proc_venc  
Module: [VENC]
```

4.2.1.16 testMedia_switch_pipeline

该指令用于测试媒体切换 pipeline。

```
(cli-uart)# testMedia_switch_pipeline  
please input 0/1 chose RGB or IR  
testMedia_switch_pipeline 0/1
```

4.2.1.17 testMedia_video_Deinit

该指令用于测试媒体视频资源释放。

```
(cli-uart)# testMedia_video_Deinit  
[ERR] check_chn_handle = 606, Ca[ll VENC Destroy Vbefore Create, fPailed  
[ERR] CVIS_VENC_StopRecvFrSame = 2885, chec-k_chn_handle, -1E073250299  
[RR] cvi_vpss.c:705:CVI_VPSS_DisableChn(): Grp(0) isn't created yet.  
[VPSS-ERR] cvi_vpss.c:371:CVI_VPSS_StopGrp(): Grp(0) isn't created yet.
```

4.2.1.18 testMedia_video_init

该指令用于测试媒体视频初始化。

4.2.1.19 dump_isp_pqgram

该指令用于显示 dump_isp 参数信息。

```
(cli-uart)# dump_isp_pqparm  
cli_dump_isp_param pIspString is null
```

4.2.1.20 proc_log

该指令用于显示当前各个模块的调试级别。

```
(cli-uart)# proc_log
BASE      ( 4)
VB        ( 4)
SYS       ( 4)
RGN       ( 4)
CHNL      ( 4)
VDEC      ( 4)
VPSS      ( 4)
VENC      ( 4)
H264E     ( 4)
JPEGGE    ( 4)
MPEG4E    ( 4)
H265E     ( 4)
JPEGD     ( 4)
VO        ( 4)
VI        ( 4)
DIS       ( 4)
RC        ( 4)
AIO       ( 4)
AI        ( 4)
AO        ( 4)
AENC      ( 4)
ADEC      ( 4)
AUD       ( 4)
VPU       ( 4)
ISP       ( 4)
IVE       ( 4)
USER      ( 4)
PROC      ( 4)
LOG       ( 4)
H264D     ( 4)
GDC       ( 4)
PHOTO     ( 4)
FB        ( 4)
SENSOR    ( 4)
```

4.2.2 ipcm 进程

4.2.2.1 ipcm_dbg

该指令用于打印大小核间通信调试信息。

```
(cli-uart)# ipcm_dbg
<info>[ipcm_dbg_print_help:13]ipcm_dbg help:
<info>[ipcm_dbg_print_help:14]ipcm_dbg pool_info : show pool get/rls callstack.
<info>[ipcm_dbg_print_help:15]ipcm_dbg log0_on : open ipcm recv and send log.
<info>[ipcm_dbg_print_help:16]ipcm_dbg log0_off : close ipcm recv and send log.
<info>[ipcm_dbg_print_help:17]ipcm_dbg logall_on : open all log(set log level to debug).
<info>[ipcm_dbg_print_help:18]ipcm_dbg logall_off : reset log level to default.
```

4.2.2.2 ipcm_dbg pool_info

该指令用于显示大小核通信 buffer 池使用情况。

```
(cli-uart)# ipcm_dbg pool_info
pool get trace:
id:0 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:1 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:2 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:3 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:4 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:5 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:6 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:7 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:8 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:9 <- 0x0 <- 0x0 <- 0x0 <- 0x0
pool free trace:
id:0 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:1 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:2 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:3 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:4 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:5 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:6 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:7 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:8 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:9 <- 0x0 <- 0x0 <- 0x0 <- 0x0

(cli-uart)# ipcm_dbg pool_info
pool get trace:
id:0 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:1 <- 0x822b706a <- 0x82228274 <- 0x822288a4 <- 0x8222e64c
id:2 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:3 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:4 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:5 <- 0x822b706a <- 0x82228274 <- 0x822288a4 <- 0x8223220e
id:6 <- 0x822b706a <- 0x82228274 <- 0x822288a4 <- 0x8223220e
id:7 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:8 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:9 <- 0x0 <- 0x0 <- 0x0 <- 0x0
pool free trace:
id:0 <- 0x82228b9e <- 0x8222e1de <- 0x82226f5a <- 0x0
id:1 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:2 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:3 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:4 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:5 <- 0x82228b9e <- 0x8222e1de <- 0x82226f5a <- 0x0
id:6 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:7 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:8 <- 0x0 <- 0x0 <- 0x0 <- 0x0
id:9 <- 0x0 <- 0x0 <- 0x0 <- 0x0
```

4.2.2.3 ipcm_dbg log0_on

该指令用于打开 ipcm 的第一个调试日志。

4.2.2.4 ipcm_dbg log0_off

该指令用于关闭 ipcm 的第一个调试日志。

4.2.2.5 ipcm_dbg logall_on

该指令用于打开 ipcm 的所有调试日志。

4.2.2.6 ipcm_dbg logall_off

该指令用于关闭 ipcm 的所有调试日志。

4.2.3 进程整体信息

4.2.3.1 tasklist

该指令用于打印进程列表。

::

```

-> tasklist ->
-> Name ID State Prio StackSize MinFreesize Runtime Candi-
date ->
> dyn_mem_proc_task 1 PEND 6 4096 3368 1031 N -> idle_task 2 RDY
61 4288 3720 141756526 N -> DEFAULT-WORKQUEUE 3 PEND 20 4096
3432 153 N -> timer_task 4 PEND 30 3840 3112 58990 N -> cpu_stats
5 SLP 60 2640 2016 50717 N -> app_task 6 SLP 32 8192 4168 206587 N
-> cli-uart 7 RDY 60 8192 6392 444838 Y -> heartbeat 8 SLP 27 1024
400 62195 N -> _isp_post_thread 9 PEND 17 4096 3344 164 N -> cvi-
task_isp_pre 10 PEND 17 4096 2896 117 N -> cvitask_isp_err 11 PEND
17 4096 3344 118 N -> gdc_job_worker 12 PEND 17 4096 3200 58340 N ->
MediaMsg 13 PEND 16 103424 101416 45291 N -> workq_thread 14 PEND
22 4096 3344 56578 N -> workq_thread 15 PEND 22 4096 3344 35171 N
-> workq_thread 16 PEND 22 4096 3344 24530 N -> workq_thread 17
PEND 22 4096 3344 38924 N -> anon_proc 18 PEND 32 8192 7504 14625
N ->

```

- Name 是 task 名称
- ID 是 taskID
- State 是 task 现在状态
- Prio 指的是优先级
- StackSize 指的是栈空间 M
- inFreesize 指的是最小自由块大小
- Runtime 指的是运行时间

4.2.3.2 taskbt

该指令用于通过 task ID 查看任务的调用栈信息。

::

```

-> taskbt -> task_btn <taskname>

-> taskbt 6 -> task name : app_task -> ===== Call stack
===== -> backtrace : 0x8220150A -> backtrace : 0x822EFD16 -
> backtrace : 0x82220822 -> backtrace : ^task entry^ -> =====
End =====

-> taskbt 7 -> task name : cli-uart -> ===== Call stack
===== -> backtrace : 0x823FC1C0 -> Status of task is
‘Running’, Can not backtrace! -> ===== End =====

```

4.2.3.3 taskbtn

该指令用于通过 task Name 查看任务的调用栈信息。

```
--> taskbtn
--> task_btn <taskname>

--> taskbtn cli-uart
--> ===== Call stack =====
--> backtrace : 0x8221CD40
--> Status of task is 'Running', Can not backtrace!
--> ===== End =====

--> taskbtn app_task
--> ===== Call stack =====
--> backtrace : 0x8220150A
--> backtrace : 0x822EFD16
--> backtrace : 0x82220822
--> backtrace : ^task entry^
--> ===== End =====
```

参数:

- Name: 任务名称
- ID: 任务 ID
- State: 任务现在状态
- Priorio: 优先级
- StackSize: 栈空间
- MinFreesize: 最小自由块大小
- Runtime: 运行时间

4.2.4 CPU 信息

4.2.4.1 cpuusage

该指令用于查看 CPU 使用率，会进行实时更新。

```
--> cpuusage
--> -----
--> CPU usage : 0.17%
--> -----
--> Name          %CPU
--> -----
--> dyn_mem_proc_task 0.00
--> idle_task        99.83
--> DEFAULT-WORKQUEUE 0.00
--> timer_task       0.00
--> cpu_stats        0.00
--> app_task         0.00
```

(下页继续)

(续上页)

```

--> cli-uart          0.00
--> heartbeat          0.00
--> _isp_post_thread   0.00
--> cvitask_isp_pre     0.00
--> cvitask_isp_err     0.00
--> gdc_job_worker     0.00
--> MediaMsg           0.00
--> workq_thread        0.00
--> workq_thread        0.00
--> workq_thread        0.00
--> workq_thread        0.00
--> anon_proc           0.00
--> cpuusage            0.16
--> -----

```

4.3 内存信息

4.3.1 查看内存信息

4.3.1.1 p

该指令用于打印物理内存内容。

::

-> **p** -> p <addr> <nunits> <width> -> addr : address to display -> nunits:
number of units to display (default is 16) -> width : width of unit, 1/2/4 (default
is 4)

p<addr> <nunits> <width>

```

--> p 0x80000000 64 2
--> 0x80000000: 0433 0005 84b3 0005 0933 0006 00ef 59c0
--> 0x80000010: 0833 0005 0533 0004 85b3 0004 0633 0009
--> 0x80000020: 58fd 0463 0118 1d63 0b05 0817 0000 0813
--> 0x80000030: 3c68 4885 282f 0118 1463 0a08 0297 0000
--> 0x80000040: 8293 3c42 0317 0000 0313 fbc3 b023 0062
--> 0x80000050: 0297 0000 8293 3b82 b283 0002 03b3 4053
--> 0x80000060: 0e17 0000 0e13 388e 3023 007e 6297 0001
--> 0x80000070: 8293 4342 7317 0001 0313 4943 8363 0662

```


4.3.1.2 p _offcheck

该指令用于打印物理内存内容, 但是不进行内存地址合理性检查

::

```
-> p 0x800000 -> p <addr> <nunits> <width> -> addr : address to display
-> nunits: number of units to display (default is 16) -> width : width
of unit, 1/2/4 (default is 4) -> -> # 使用 p 打印非法地址, 会提示打
印 p 打印物理地址的用法, 使用 p_offcheck 打印非法地址会陷入 dump
状态。-> (0x82402410): 0x20202020 0x20202020 0x19191919 0x19191919
-> (0x82402420): 0x18181818 0x18181818 0x09090909 0x09090909 ->
(0x82402430): 0x08080808 0x08080808 0x8221FCCC 0x00000000 -> !!!!!!!!
dump end !!!!!!!!
```

4.3.2 修改内存信息

4.3.2.1 m

该指令用于修改物理内存内容。

::

```
-> m -> m <addr> <value> <width> -> addr : address to modify -> value
: new value (default is 0) -> width : width of unit, 1/2/4 (default is 4)
```

```
(cli-uart)# p 0x80000000
0x80000000: 00050433 000584b3 00060933 5a4000ef
0x80000010: 00050833 00040533 000485b3 00090633
0x80000020: 046358fd 1d630118 08170b05 08130000
0x80000030: 48853c68 0118282f 0a081463 00000297

(cli-uart)# m 0x80000000 1 4
value on 0x80000000 change from 0x50433 to 0x1.

(cli-uart)# p 0x80000000
0x80000000: 00000001 000584b3 00060933 5a4000ef
0x80000010: 00050833 00040533 000485b3 00090633
0x80000020: 046358fd 1d630118 08170b05 08130000
0x80000030: 48853c68 0118282f 0a081463 00000297
```

- 第一个参数: 物理地址
- 第二个参数: 要修改的新数值 (不填写, 默认修改为 0)
- 第三个参数: 修改的字节个数 (可选项为 1/2/4, 不填写, 默认修改 4 个字节)
- 左侧 m 0x80000000 1 4 指的是将 0x80000000 地址内容修改 4Byte 修改的新数值为 1,
- 修改后该地址处内容由 00050433 修改为 00000001

4.3.3 内存调试信息

4.3.3.1 ddumpsys

该指令用于显示系统调试信息。

4.3.3.2 dumpsys mm

该指令用于显示系统调试内存信息。

```
--> dumpsys mm
--> -----
--> [HEAP]| TotalSz | FreeSz | UsedSz | MinFreeSz | MaxFreeBlkSz |
-->      | 0x005066C0 | 0x004AE580 | 0x00058140 | 0x004AE100 | 0x004AE580 |
--> -----
--> -----
--> [HEAP]| TotalSz | FreeSz | UsedSz | MinFreeSz | MaxFreeBlkSz |
-->      | 0x015A0000 | 0x00E3BCC0 | 0x00764340 | 0x00E3BCC0 | 0x00E3BCC0 |
--> -----
```

4.3.3.3 dumpsys mm_info

该指令用于显示 alios 内存分区信息。

```

----- all memory blocks -----
g_kmm_head = 8243c640
ALL BLOCKS
Blk Addr      Stat      Len  Chk      Caller      Point
0x8243c800    used        64   OK      0x0          (0x0 <- 0x0 <- 0x0 <- 0x0)
0x8243c880    used      65536   OK      0x82343678   (0x0 <- 0x0 <- 0x0 <- 0x0)
0x8244c8c0    used      65536   OK      0x8234368c   (0x0 <- 0x0 <- 0x0 <- 0x0)
0x8245c900    used      8192    OK      0x8221f51c   (0x0 <- 0x0 <- 0x0 <- 0x0)
0x8245e940    used       320   OK      0x8221f52c   (0x0 <- 0x0 <- 0x0 <- 0x0)
0x8245eac0    used        64   OK      0x822395a0   (0x8223959c <- 0x823436ac <- 0x823436c8 <- 0x82220840)
0x8245eb40    used       128   OK      0x822213ac   (0x822213a8 <- 0x822216da <- 0x8234374e <- 0x823436cc)
0x8245ec00    used       128   OK      0x8221cb4e   (0x8221cb4a <- 0x82220f48 <- 0x822266c0 <- 0x82221700)
0x8245ecc0    used       256   OK      0x822265d2   (0x822266ce <- 0x82221700 <- 0x8234374e <- 0x823436cc)
0x8245ee00    used       128   OK      0x822213ac   (0x822213a8 <- 0x82226702 <- 0x82221700 <- 0x8234374e)
0x8245eec0    used       128   OK      0x822213ac   (0x822213a8 <- 0x82226716 <- 0x82221700 <- 0x8234374e)
0x8245ef80    used        64   OK      0x82238d8e   (0x82238d8a <- 0x82226728 <- 0x82221700 <- 0x8234374e)
0x8245f000    used      8192   OK      0x82238d90   (0x82238d94 <- 0x82226728 <- 0x82221700 <- 0x8234374e)
0x82461040    used      4160   OK      0x8222172a   (0x82221726 <- 0x8234374e <- 0x823436cc <- 0x82220840)
0x824620c0    used      1024   OK      0x822067c0   (0x822067bc <- 0x8220f68e <- 0x8220b644 <- 0x0)
0x82462500    used      1024   OK      0x8223a230   (0x8223a22c <- 0x823436cc <- 0x82220840 <- 0x0)
----- all free memory blocks -----
freelist bitmap: 0x10000
Address Stat Len Chk Caller
0x824a9cc0 free 3498688 OK 0x0 (0x82221e12 <- 0x82221ec6 <- 0x82221f16 <- 0x8222108)

----- memory allocation statistic -----
[2^06] bytes: 27 | [2^07] bytes: 111 | [2^08] bytes: 17 | [2^09] bytes: 16 |
[2^10] bytes: 616 | [2^11] bytes: 3 | [2^12] bytes: 2 | [2^13] bytes: 10 |
[2^14] bytes: 5 | [2^15] bytes: 1 | [2^16] bytes: 2 | [2^17] bytes: 3 |
[2^18] bytes: 0 | [2^19] bytes: 0 | [2^20] bytes: 0 | [2^21] bytes: 0 |
[2^22] bytes: 0 | [2^23] bytes: 0 | [2^24] bytes: 0 | [2^25] bytes: 0 |
[2^26] bytes: 0 | [2^27] bytes: 0 | [2^28] bytes: 0 | [2^29] bytes: 0 |

----- [kernel] heap size overview -----
[HEAP] TotalSz | FreeSz | UsedSz | MinFreeSz | MaxFreeBlkSz |
        0x003c39c0 | 0x003562c0 | 0x0006d700 | 0x00355e40 | 0x003562c0 |

----- task allocation statistic -----
TaskName      Prio StackSize HeapAllocSize
app_task      32  1024    294656
cli_uart      60  1024     576
Allocated in ISR 139584, Allocated by deleted task 0

```

4.3.3.4 dumpsys mm_info_resv

该指令用于显示 alios resv 分区内存信息。

```

g_kmm_head_resv = 82800000
ALL BLOCKS
Blk Addr   Stat   Len  Chk   Caller   Point
0x828001c0 used    64   OK   0x0      (0x0 <- 0x0 <- 0x0 <- 0x0)
0x82800240 free  25034048 OK   0x0      (0x0 <- 0x0 <- 0x0 <- 0x0)
0x83fdffc0 used  sentinel OK   0x11224433 (0x0 <- 0x0 <- 0x0 <- 0x0)

----- all free memory blocks -----
freelist bitmap: 0x800000
Address Stat   Len  Chk   Caller   Point
0x82800240 free  25034048 OK   0x0      (0x0 <- 0x0 <- 0x0 <- 0x0)

----- memory allocation statistic -----
[2^06] bytes: 0 | [2^07] bytes: 0 | [2^08] bytes: 0 | [2^09] bytes: 0 |
[2^10] bytes: 0 | [2^11] bytes: 0 | [2^12] bytes: 0 | [2^13] bytes: 0 |
[2^14] bytes: 0 | [2^15] bytes: 0 | [2^16] bytes: 0 | [2^17] bytes: 0 |
[2^18] bytes: 0 | [2^19] bytes: 0 | [2^20] bytes: 0 | [2^21] bytes: 0 |
[2^22] bytes: 0 | [2^23] bytes: 0 | [2^24] bytes: 0 | [2^25] bytes: 0 |
[2^26] bytes: 0 | [2^27] bytes: 0 | [2^28] bytes: 0 | [2^29] bytes: 0 |

-----[kernel] heap size overview -----
[HEAP] | TotalSz | FreeSz | UsedSz | MinFreeSz | MaxFreeBlkSz |
        | 0x017E0000 | 0x017DFD40 | 0x000002C0 | 0x017DFD40 | 0x017DFD40 |

----- task allocation statistic -----
TaskName      Prio StackSize HeapAllocSize
Allocated in ISR 0, Allocated by deleted task 0

```

4.3.3.5 mmlk

该指令用于显示内存泄漏信息。

::

```

-> mmlk -> memory leak chec-----All new alloted blocks:----- ->
Blk_Addr Stat Len Chk Caller Point -> -----New alloted blocks info ends.
----- -> -----All new alloted blocks:----- -> Blk_Addr Stat
Len Chk Caller Point -> -----New alloted blocks info ends.----- -> k
start. -> Add tag 1 when malloc

```

4.4 调试信息

4.4.1 debug

该指令用于显示系统 debug 信息，包括堆栈信息、任务列表、信号量以及锁的等待情况。


```
(cli-uart)# debug
===== Heap Info =====
[HEAP]| TotalSz | FreeSz | UsedSz | MinFreeSz | MaxFreeBlkSz |
      | 0x003C39C0 | 0x003562C0 | 0x0006D700 | 0x00355E40 | 0x003562C0 |
-----
[HEAP]| TotalSz | FreeSz | UsedSz | MinFreeSz | MaxFreeBlkSz |
      | 0x017E0000 | 0x017DFD40 | 0x000002C0 | 0x017DFD40 | 0x017DFD40 |
-----
===== Task Info =====
TaskName      State   Prio    Stack      StackSize (MinFree)
-----
dyn_mem_proc_task  PEND   0x00000006 0x823C2C70 0x00001000(0x00000D30)
idle_task        RDY    0x00000003D 0x823C3DC0 0x000010C0(0x00000E90)
DEFAULT-WORKQUEUE  PEND   0x00000014 0x823C6470 0x00001000(0x00000D70)
timer_task       PEND   0x0000001E 0x823C53B0 0x00000F00(0x00000C30)
cpu_stats        SLP    0x0000003C 0x823C1F98 0x00000A50(0x000007D8)
app_task         SLP    0x00000020 0x8245C940 0x00002000(0x00001A58)
cli-uart         RDY    0x0000003C 0x82463240 0x00002000(0x00001968)
_isp_post_thread  PEND   0x00000011 0x8246BF40 0x00001000(0x00000D00)
cvitask_isp_pre   PEND   0x00000011 0x8246D680 0x00001000(0x00000B20)
cvitask_isp_err   PEND   0x00000011 0x8246EDC0 0x00001000(0x00000D00)
gdc_job_worker    PEND   0x00000011 0x82484380 0x00001000(0x00000C90)
MediaMsg          PEND   0x00000010 0x8248AF40 0x00019400(0x00018EC8)
workq_thread      PEND   0x00000010 0x824A49C0 0x00001000(0x00000D10)
workq_thread      PEND   0x00000010 0x824A5EC0 0x00001000(0x00000D10)
workq_thread      PEND   0x00000010 0x824A73C0 0x00001000(0x00000D10)
workq_thread      PEND   0x00000010 0x824A88C0 0x00001000(0x00000D10)
===== Queue Info =====
QueAddr      TotalSize  PeakNum   CurrNum    TaskWaiting
-----
===== Buf Queue Info =====
BufQueAddr  TotalSize  PeakNum   CurrNum    MinFreeSz  TaskWaiting
-----
0x823C51D8  0x00000280 0x00000000 0x00000000 0x00000280 timer_task
===== Sem Waiting =====
SemAddr      Count      PeakCount  TaskWaiting
-----
0x823C5130  0x00000000 0x00000000 dyn_mem_proc_task
0x823C6418  0x00000000 0x00000000 DEFAULT-WORKQUEUE
0x82484040  0x00000000 0x00000000 gdc_job_worker
0x8248A540  0x00000000 0x00000000 MediaMsg
0x824A4740  0x00000000 0x00000000 workq_thread
0x824A5C40  0x00000000 0x00000000 workq_thread
0x824A7140  0x00000000 0x00000000 workq_thread
0x824A8640  0x00000000 0x00000000 workq_thread
Total: 0x00000024
===== Mutex Waiting =====
MutexAddr    TaskOwner      NestCnt    TaskWaiting
-----
Total: 0x00000039
```