MI\_调试手册

**Version 0.1**

© 2018 MStar Semiconductor, Inc. All rights reserved.

MStar Semiconductor makes no representations or warranties including, for example but not limited to, warranties of merchantability, fitness for a particular purpose, non-infringement of any intellectual property right or the accuracy or completeness of this document, and reserves the right to make changes without further notice to any products herein to improve reliability, function or design. No responsibility is assumed by MStar Semiconductor arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others.

MStar is a trademark of MStar Semiconductor, Inc. Other trademarks or names herein are only for identification purposes only and owned by their respective owners.

REVISION HISTORY

| Revision No. | Description | Date |
| --- | --- | --- |
| 0.1 | * Initial release | 12/20/2017 |

目录

[目录 ii](#_Toc505765820)

[1. 概述 4](#_Toc505765821)

[1.1. 简介 4](#_Toc505765822)

[1.2. 文件清单 4](#_Toc505765823)

[2. SYS 5](#_Toc505765824)

[2.1. common 5](#_Toc505765825)

[2.1.1 dump\_mmap 5](#_Toc505765826)

[2.1.2 dump\_config 6](#_Toc505765827)

[2.2. mi\_log\_info 7](#_Toc505765828)

[2.2.1 cat 7](#_Toc505765829)

[2.2.2 echo 8](#_Toc505765830)

[2.3. mi\_global\_info 9](#_Toc505765831)

[2.3.1 cat 9](#_Toc505765832)

[2.4. debug\_level 10](#_Toc505765833)

[2.4.1 cat/echo 10](#_Toc505765834)

[2.5. miu\_protect 11](#_Toc505765835)

[2.5.1 cat 11](#_Toc505765836)

[2.6. mma\_heap\_name 12](#_Toc505765837)

[2.6.1 cat 12](#_Toc505765838)

[2.6.2 echo 14](#_Toc505765839)

[2.7. vb\_pool\_global 14](#_Toc505765840)

[2.7.1 cat 14](#_Toc505765841)

[2.7.2 echo 15](#_Toc505765842)

[2.8. vb\_pool 15](#_Toc505765843)

[2.8.1 cat 15](#_Toc505765844)

[2.8.2 echo 16](#_Toc505765845)

[2.9. meta 16](#_Toc505765846)

[2.9.1 cat 16](#_Toc505765847)

[2.9.2 echo 17](#_Toc505765848)

[2.10. mi\_modulenamedevid 17](#_Toc505765849)

[2.10.1 cat 17](#_Toc505765850)

[2.10.2 echo 20](#_Toc505765851)

[2.11. mi\_dump\_buffer\_delay\_worker 21](#_Toc505765852)

[2.11.1 cat 21](#_Toc505765853)

[2.11.2 echo 21](#_Toc505765854)

[2.12. module version 22](#_Toc505765855)

[2.12.1 cat 22](#_Toc505765856)

[3. AI 23](#_Toc505765857)

[3.1. cat 23](#_Toc505765858)

[4. AO 26](#_Toc505765859)

[4.1. cat 26](#_Toc505765860)

[4.2. echo 28](#_Toc505765861)

[5. VIF 31](#_Toc505765862)

[5.1. cat 31](#_Toc505765863)

[6. VPE 33](#_Toc505765864)

[6.1. cat 33](#_Toc505765865)

[6.2. echo 35](#_Toc505765866)

[7. VENC 37](#_Toc505765867)

[7.1. cat 37](#_Toc505765868)

[7.2. echo 39](#_Toc505765869)

[8. DIVP 40](#_Toc505765870)

[8.1. cat 40](#_Toc505765871)

[8.2. echo 42](#_Toc505765872)

[9. VDISP 45](#_Toc505765873)

[9.1. cat 45](#_Toc505765874)

[10. DISP 47](#_Toc505765875)

[10.1. cat 47](#_Toc505765876)

[10.2. echo 50](#_Toc505765877)

[11. HDMI 54](#_Toc505765878)

[11.1. cat 54](#_Toc505765879)

[12. FB 56](#_Toc505765880)

[12.1. cat 56](#_Toc505765881)

[12.2. echo 60](#_Toc505765882)

[13. GFX 62](#_Toc505765883)

[13.1. cat 62](#_Toc505765884)

[14. REGION 63](#_Toc505765885)

[14.1. cat 63](#_Toc505765886)

[14.2. echo 65](#_Toc505765887)

[15. VDEC 68](#_Toc505765888)

[15.1. cat 68](#_Toc505765889)

[15.2. echo 70](#_Toc505765890)

# 概述

## 简介

【说明】

使用Linux下的porc文件系统，可以打印出各个模块当前的运行状态，便于调试及分析问题。每个模块都自己特定的路径，部分模块有额外的echo命令。

【路径】

路径为 /proc/mi\_modules

## 文件清单

|  |  |
| --- | --- |
| 模块名称 | 说明 |
| SYS | SYS模块的信息 |
| AI | 音频输入的信息 |
| AO | 音频输出的信息 |
| VIF | 视频输入的信息 |
| VPE | 视频处理引擎的信息 |
| VENC | 编码视频的信息 |
| DIVP | 去隔行/图像引擎的信息 |
| VDISP | 虚拟显示模块的信息 |
| DISP | 显示模块的信息 |
| HDMI | HDMI的信息 |
| FB | Graphic图形层的信息 |
| GFX | 图像引擎的信息 |
| REGION | OSD的信息 |
| VDEC | 解码视频的信息 |

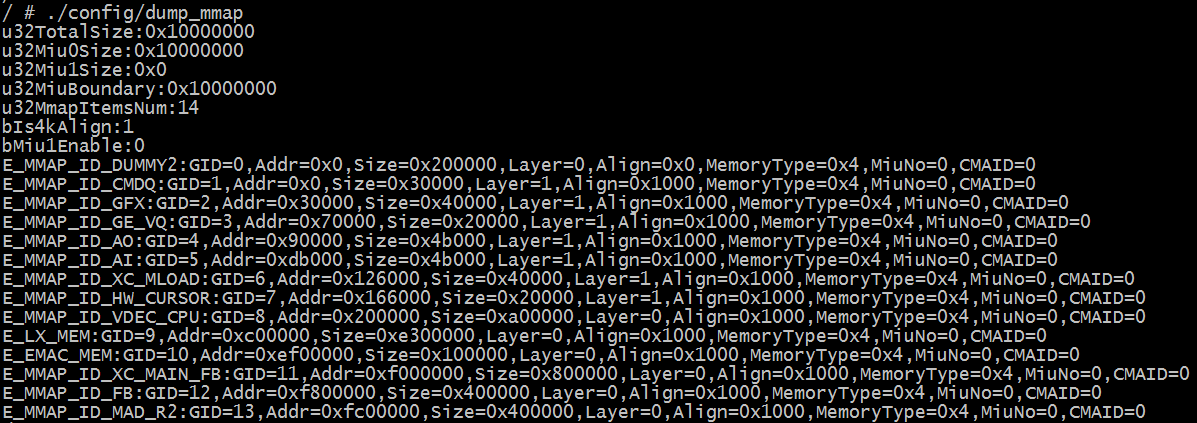
# SYS

## common

### dump\_mmap

【调试信息】

# ./config/dump\_mmap



【调试信息分析】

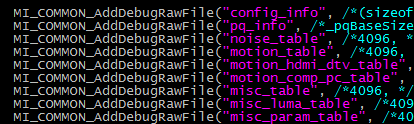
Common下的内容存放的是配置API，之所以会有配置API，有两个目的：

1 把iniparser放在用户态做，iniparser的任何开销不带入内核

2 方便调试时确认配置文件的实际内容，因为在parser时打印的内容可能会与实际设置的内容有偏差

为了达到上面两个目的，导出了一些文件

如下是导入config的配置文件：



Systeminfo.c里面parse好的struct写入config\_info里面

Pqloader.c里面parse好的struct写入pq\_info和\*\_table里面，这里的table跟MISDK里面的那一份不一样的一点是这里的table是连续的内存摆放，MISDK是一个间接数组指向一列一维数组

如下是导入mmap的配置文件：



Mmapinfo.c里面parse好的struct写入mmap\_info和memory\_info，因为这里比较简单，所以从mmap\_info和memory\_info读出来是格式化好的可读信息

### dump\_config

【调试信息】

# ./config/dump\_config

panel size:24

DBC Value=

55,30,10

254,66,0

72,80,72

start dump [motion\_table](6, 8)

{0xe8,0xcd,0xab,0x89,0x67,0x45,0x23,0x00,}

{0xd8,0xbc,0x9a,0x78,0x56,0x34,0x12,0x00,}

{0xc8,0xab,0x89,0x67,0x45,0x23,0x01,0x00,}

{0xb8,0x9a,0x78,0x56,0x34,0x12,0x00,0x00,}

{0xa8,0x89,0x67,0x45,0x23,0x01,0x00,0x00,}

{0x98,0x78,0x56,0x34,0x12,0x00,0x00,0x00,}

end dump

start dump [motion\_hdmi\_dtv\_table](4, 8)

{0x68,0x45,0x23,0x01,0x00,0x00,0x00,0x00,}

{0xa8,0x89,0x67,0x45,0x23,0x01,0x00,0x00,}

{0xc8,0xab,0x89,0x67,0x45,0x23,0x01,0x00,}

{0xe8,0xcd,0xab,0x89,0x67,0x45,0x23,0x00,}

end dump

start dump [motion\_comp\_pc\_table](4, 8)

{0x68,0x45,0x23,0x01,0x00,0x00,0x00,0x00,}

{0x98,0x78,0x56,0x34,0x12,0x01,0x00,0x00,}

{0xa8,0x89,0x67,0x45,0x23,0x12,0x01,0x00,}

{0xb8,0x9a,0x78,0x56,0x34,0x12,0x01,0x00,}

end dump

start dump [misc\_param\_table](4, 4)

{0x02,0xff,0x00,0x00,}

{0x02,0xff,0x01,0x66,}

{0x03,0x88,0x01,0x66,}

{0x03,0xaa,0x01,0x66,}

end dump

start dump [misc\_luma\_table](8, 2)

{0x33,0x08,}

{0x33,0x07,}

{0x22,0x06,}

{0x22,0x05,}

{0x11,0x04,}

{0x11,0x03,}

{0x00,0x02,}

{0x00,0x01,}

end dump

start dump [noise\_table](6, 8)

{0x67,0x45,0x34,0x23,0x12,0x01,0x00,0x00,}

{0xab,0x89,0x67,0x45,0x23,0x12,0x01,0x00,}

{0xbc,0x9a,0x78,0x56,0x34,0x23,0x01,0x00,}

{0xde,0xbc,0x9a,0x78,0x56,0x34,0x12,0x00,}

{0xef,0xcd,0xab,0x89,0x67,0x45,0x23,0x00,}

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,}

end dump

start dump [misc\_table](6, 15)

{0x00,0x00,0x00,0x00,0x00,0x02,0x00,0x00,0x02,0x00,0xc0,0x9f,0x00,0x01,0x66,}

{0x10,0x01,0xc1,0xc1,0xc1,0x04,0x09,0x1f,0x03,0x88,0xc0,0x9f,0x00,0x01,0x66,}

{0x21,0x02,0xc2,0xc2,0xc2,0x08,0x09,0x1f,0x03,0x99,0x00,0x00,0x00,0x01,0x88,}

{0x31,0x04,0xc2,0xc2,0xc2,0x0a,0x09,0x1f,0x03,0xaa,0x00,0x00,0x00,0x01,0xaa,}

{0x41,0x06,0xc3,0xc3,0xc3,0x10,0x19,0x1f,0x03,0xbb,0x00,0x00,0x14,0x01,0xbb,}

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,}

end dump

【调试信息分析】

./config/dump\_config得到的是/proc/mi\_modules/common/里各文件的信息，含panel size、DBC Value、motion\_table、motion\_hdmi\_dtv\_table、motion\_comp\_pc\_table、misc\_param\_table、misc\_luma\_table、noise\_table、misc\_table等

## mi\_log\_info

### cat

【调试信息】

# cat /proc/mi\_modules/mi\_log\_info

---------------- Log Path ------------------------

log path:

---------------- Store Path ----------------------

store path: /mnt

---------------- Module Log Level ----------------

Log module Level

--------------------------

mi\_ive 2(WRN)

mi\_vdf 2(WRN)

mi\_venc 2(WRN)

mi\_rgn 2(WRN)

mi\_ai 2(WRN)

mi\_ao 2(WRN)

mi\_vif 2(WRN)

mi\_vpe 2(WRN)

mi\_vdec 2(WRN)

mi\_sys 2(WRN)

mi\_fb 2(WRN)

mi\_hdmi 2(WRN)

mi\_divp 2(WRN)

mi\_gfx 2(WRN)

mi\_vdisp 2(WRN)

mi\_disp 2(WRN)

mi\_os 2(WRN)

mi\_iae 2(WRN)

mi\_md 2(WRN)

mi\_od 2(WRN)

mi\_shadow 2(WRN)

【调试信息分析】

示意了默认的log path 或store path，同时示意了各module的debug\_level的值。

【参数分析】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| mi\_log\_info | log path | 暂时无效果 |
| store path | 暂时无效果 |
| Log module | 各个模块的名称 |
| Level | 打印等级 |

### echo

|  |  |
| --- | --- |
| 功能 | 修改模块debug\_level |
| 命令 | echo [ModID]=[Level] > /proc/mi\_modules/mi\_log\_info |
|  | [ModID] 模块的名字  mi\_ive mi\_vdf mi\_venc mi\_rgn  mi\_ai mi\_ao mi\_vif mi\_vpe  mi\_vdec mi\_sys mi\_fb mi\_hdmi  mi\_divp mi\_gfx mi\_vdisp mi\_disp  mi\_os mi\_iae mi\_md mi\_od  mi\_shadow |
| 参数说明 | [Level] 0 无Debug信息  1 只显示error的信息 (MI\_DBG\_ERR)  2 只显示waring的信息 (MI\_DBG\_WRN)  3 只显示 info 的信息 (MI\_DBG\_INFO)  4 显示所有信息 |
| 举例 | echo mi\_sys=2 > /proc/mi\_modules/mi\_log\_info  将mi\_sys 的debug\_level修改为2 |

|  |  |
| --- | --- |
| 功能 | 修改log的路径 |
| 命令 | echo log=[Path] > /proc/mi\_modules/mi\_log\_info |
| 参数说明 | [Path] 路径 |
| 举例 | echo log=/mnt > /proc/mi\_modules/mi\_log\_info |

|  |  |
| --- | --- |
| 功能 | 修改存储log的路径 |
| 命令 | echo storepath=[Path] > /proc/mi\_modules/mi\_log\_info |
| 参数说明 | [Path] 路径 |
| 举例 | echo storepath=/mnt > /proc/mi\_modules/mi\_log\_info |

## mi\_global\_info

### cat

【调试信息】

# cat /proc/mi\_modules/mi\_global\_info

miu\_and\_lx\_info:

ARM\_MIU0\_BUS\_BASE 0x20000000 ARM\_MIU0\_BASE\_ADDR 0x0

ARM\_MIU1\_BUS\_BASE 0x60000000 ARM\_MIU1\_BASE\_ADDR 0x80000000

ARM\_MIU2\_BUS\_BASE 0xffffffff ARM\_MIU2\_BASE\_ADDR 0xffffffff

lx\_mem\_addr 0x20c00000 lx\_mem\_size 0xe300000

lx\_mem2\_addr 0xffffffff lx\_mem2\_size 0xffffffff

lx\_mem3\_addr 0xffffffff lx\_mem3\_size 0xffffffff

KernelProtect IP white list:

clientId name

43 MIU\_CLIENT\_MIPS\_RW

50 MIU\_CLIENT\_NAND\_RW

82 MIU\_CLIENT\_USB\_UHC0\_RW

83 MIU\_CLIENT\_USB\_UHC1\_RW

84 MIU\_CLIENT\_USB\_UHC2\_RW

18 MIU\_CLIENT\_G3D\_RW

140 MIU\_CLIENT\_USB3\_RW

129 MIU\_CLIENT\_SDIO\_RW

165 MIU\_CLIENT\_SATA\_RW

133 MIU\_CLIENT\_USB\_UHC3\_RW

225 MIU\_CLIENT\_USB30\_1\_RW

226 MIU\_CLIENT\_USB30\_2\_RW

5 MIU\_CLIENT\_BDMA\_RW

14 MIU\_CLIENT\_EMAC\_RW

PAGE\_OFFSET - the virtual address of the start of the kernel image

PAGE\_OFFSET=0xc0000000

TASK\_SIZE - the maximum size of a user space task

TASK\_SIZE=0xbf000000

MStar SDK version: commit.build\_time 0be783c.2017121315

CHIP\_VERSION U02

【调试信息分析】

该调试信息提供了全局global的一些信息

【参数分析】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| miu\_and\_lx\_info(以只有一个MIU为例) | ARM\_MIU0\_BUS\_BASE | Miu0 bus base |
| ARM\_MIU0\_BASE\_ADDR | Miu0 base addr |
| lx\_mem\_addr | Linux镜像占的memory的起始地址(属于cpu bus address) |
| lx\_mem\_size | Linux镜像占的memory的size |
| kernelProtect IP white list | clientId | Miu protect的IP白名单里的IP的id（从未分group的角度看的全局的id） |
| name | 与clientId对应的该IP的实际的名字 |
| MStar SDK version: commit.build\_time | commit | sdk对应的commit |
| build\_time | sdk的build时间 |
| CHIP\_VERSION | CHIP\_VERSION | 当前chip的版本，版本号是U01,U02,U03… |

## debug\_level

### cat/echo

【调试信息】

# cat /proc/mi\_modules/mi\_sys/debug\_level

2

【调试信息分析】

每个module（包括sys这个module）都有各自的debug level,是为了控制打印级别,各自的打印级别分别在/proc/mi\_modules/mi\_modulename/debug\_level控制，其中modulename形如disp,divp,rgn等等。上面只是以/proc/mi\_modules/mi\_sys/debug\_level为例

|  |  |
| --- | --- |
| 功能 | 打印警告级别 |
| 命令 | cat /proc/mi\_modules/[ModuleName]/debug\_level |
| 参数说明 | [ModuleName] 模块的名字  mi\_disp mi\_gfx mi\_rgn mi\_vdec mi\_vpe  mi\_ai mi\_divp mi\_shadow mi\_vdisp mi\_ao mi\_hdmi mi\_sys mi\_venc mi\_bar mi\_vif |
| 举例 | cat /proc/mi\_modules/mi\_sys/debug\_level  2  mi\_sys 的警告级别为2(只显示waring的信息) |

|  |  |
| --- | --- |
| 功能 | 修改警告级别 |
| 命令 | echo [Level] > /proc/mi\_modules/[ModuleName]/debug\_level |
| 参数说明 | [Level] 0 无Debug信息  1 只显示error的信息 (MI\_DBG\_ERR)  2 只显示waring的信息 (MI\_DBG\_WRN)  3 只显示 info 的信息 (MI\_DBG\_INFO)  4 显示所有信息 |
| [ModuleName] 模块的名字  mi\_disp mi\_gfx mi\_rgn mi\_vdec mi\_vpe  mi\_ai mi\_divp mi\_shadow mi\_vdisp mi\_ao mi\_hdmi mi\_sys mi\_venc mi\_bar mi\_vif |
| 举例 | echo 1 > /proc/mi\_modules/mi\_vdec/debug\_level  将mi\_vdec 模块的警告级别修改为只显示error的信息 |

## miu\_protect

### cat

【调试信息】

cat /proc/mi\_modules/mi\_sys\_mma/miu\_protect

=================== start miu\_protect\_info ================================

kernel protect enabled

LX :

cpu\_start\_addr:0x20c00000 size:0xe300000

miu\_index miuBlockIndex start\_cpu\_bus\_pa length

0x0 0x00 0x20c00000 0x460000

KernelProtect IP white list:

clientId name

43 MIU\_CLIENT\_MIPS\_RW

50 MIU\_CLIENT\_NAND\_RW

82 MIU\_CLIENT\_USB\_UHC0\_RW

83 MIU\_CLIENT\_USB\_UHC1\_RW

84 MIU\_CLIENT\_USB\_UHC2\_RW

18 MIU\_CLIENT\_G3D\_RW

140 MIU\_CLIENT\_USB3\_RW

129 MIU\_CLIENT\_SDIO\_RW

165 MIU\_CLIENT\_SATA\_RW

133 MIU\_CLIENT\_USB\_UHC3\_RW

225 MIU\_CLIENT\_USB30\_1\_RW

226 MIU\_CLIENT\_USB30\_2\_RW

5 MIU\_CLIENT\_BDMA\_RW

14 MIU\_CLIENT\_EMAC\_RW

【调试信息分析】

该命令显示了miu protect相关的信息

【参数分析】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| kernel protect |  | 值是enabled或者disabled，表示是否有enable kernel protect，默认是需要enable kernel protect的。 |
| LX(以只有一个LX为例,LX表示linux 镜像对应的memory) | cpu\_start\_addr | 该LX对应的起始CPU addr。 |
| size | 该LX对应的size。 |
| 某个kernel protect 的range的相关信息 | miu\_index | 编号。 |
| miuBlockIndex | 总共4个miu范围的编号信息。 |
| start\_cpu\_bus\_pa | 该range的起始cpu bus addr。 |
| length | 该range的length。 |
| KernelProtect IP white list | clientId | Miu protect的IP白名单里的IP的id（从未分group的角度看的全局id）。 |
| name | 与clientId对应的该IP的实际的名字。 |

## mma\_heap\_name

### cat

【调试信息】

# cat /proc/mi\_modules/mi\_sys\_mma/mma\_heap\_name0

mma heap name heap\_base\_cpu\_bus\_addr length chunk\_mgr\_avail

mma\_heap\_name0 25200000 9d00000 79c6000

chunk\_mgr info:

pst\_chunk\_mgr offset length avail

c3545618 0 9d00000 79c6000

each chunk info:

offset length used\_flag task\_name

0 1000 1 insmod

1000 100000 1 insmod

101000 200000 1 proc\_rtsp

301000 3f5000 1 vdisp-dev0

6f6000 3f5000 1 vdisp-dev0

aeb000 3f5000 1 vdisp-dev0

ee0000 1f5000 0 NA

10d5000 1000 1 proc\_rtsp

10d6000 64000 1 proc\_rtsp

113a000 200000 1 proc\_rtsp

133a000 700000 1 proc\_rtsp

1a3a000 3f5000 0 NA

1e2f000 3f5000 1 vdisp-dev0

2224000 700000 1 proc\_rtsp

2924000 73dc000 NA

【调试信息分析】

该cat信息对应的是该mma heap的基本信息和当前的一些状态。

【参数分析】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| heap basic info | mma heap name | mma heap的name。 |
| heap\_base\_cpu\_bus\_addr | heap的起始cpu bus addr。 |
| length | heap的length。 |
| chunk\_mgr\_avail | heap里的剩余的未用的memory的总量。 |
| chunk\_mgr info | pst\_chunk\_mgr | chunk mgr的指针的值。 |
| offset | chunk mgr的offset,由于整个mma heap做为一个chunk mgr,因此该值永远为0。 |
| length | chunk mgr的length,由于整个mma heap做为一个chunk mgr,因此该值永远为mma heap length。 |
| avail | chunk mgr(即heap)里的剩余的未用的memory的总量。 |
| each chunk info  (chunk mgr里各chunk的信息和使用情况) | offset | 该chunk在chunk mgr里的offset。 |
| length | 该chunk的length。 |
| used\_flag | 该chunk是否被alloc出去使用了。是的话该值为1；否则free状态的话该值为0。 |
| task\_name | 如果used flag为1，则task\_name里存的是哪个task使用的它；否则该值为无效值NA。 |

### echo

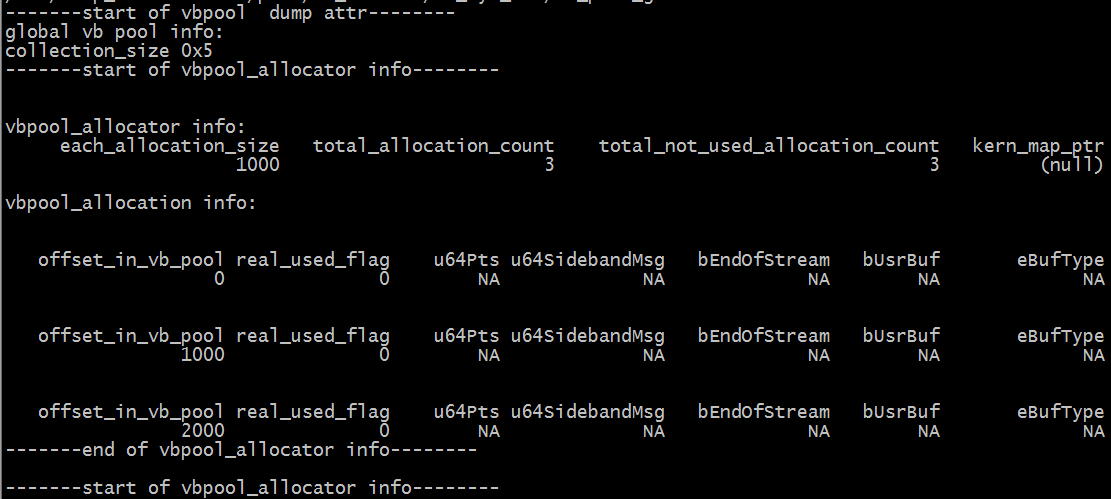
|  |  |
| --- | --- |
| 功能 | 即dump 指定offset和length的data到指定的路径 |
| 命令 | echo [Path] [Offset] [Length] > /proc/mi\_modules/mi\_sys\_mma/mma\_heap\_name[Num] |
| 参数说明 | [Path] 文件要保存的路径,只需提供路径，不要提供具体的文件名，系统会根据参数自动生成对应的文件名。 |
| [Offset] 从该mma heap的哪个offset开始dump data,必须4KB对齐 |
| [Length] 在该mma heap里dump的总的数据量的大小,必须4KB对齐 |
| [Num] mma head 对应的数字，目前 mma\_heap\_name0  可通过 cat /proc/cmdline 查看 |
| 举例 | echo /mnt/ 0 4096 > /proc/mi\_modules/mi\_sys\_mma/mma\_heap\_name0  在 /mnt下产生 mma\_\_mma\_heap\_name0\_\_0\_\_4096.bin 文件 |

## vb\_pool\_global

### cat

【调试信息】

# cat /proc/mi\_modules/mi\_sys\_mma/vb\_pool\_global



【调试信息分析】

如果有通过MI\_SYS\_IMPL\_ConfGloPubPools API或者MI\_SYS\_ConfGloPubPools API配置了global public pools的话，会有/proc/mi\_modules/mi\_sys\_mma/vb\_pool\_global文件；如果没有配置的话，那么没有该文件。

【调试信息分析】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
|  | collection\_size | 该pools的allocator个数 |
| 其中某个vbpool\_allocator 的info | each\_allocation\_size | 该allocator里的allocation的size（同一个vbpool allocator的不同allocation的size是相同的，同一个pools里的不同的allocator的不同的allocations的size可能不同）。 |
| total\_allocation\_count | 该allocator里的allocations的数目。 |
| total\_not\_used\_allocation\_count | 该allocator里的allocations中未被使用的数目。 |
| kern\_map\_ptr | 该allocator的offset0位置的kernel space va；如果没有的话，该值为NULL。 |
| 指定的vbpool\_allocator的某个vbpool\_allocation的 info | offset\_in\_vb\_pool | 该allocation在对应的allocator里的逻辑offset。 |
| real\_used\_flag | 是否被使用中，否的话则后续参数为无效值NA。 |
| u64Pts | 如果有效，该值为pts。 |
| u64SidebandMsg | 如果有效，该值为SidebandMsg。 |
| bEndOfStream | 如果有效，该值为是否是end stream的flag。 |
| bUsrBuf | 如果有效，该值为表示是否是UsrBuf。 |
| eBufType | 如果有效，该值表示buf的type。 |

### echo

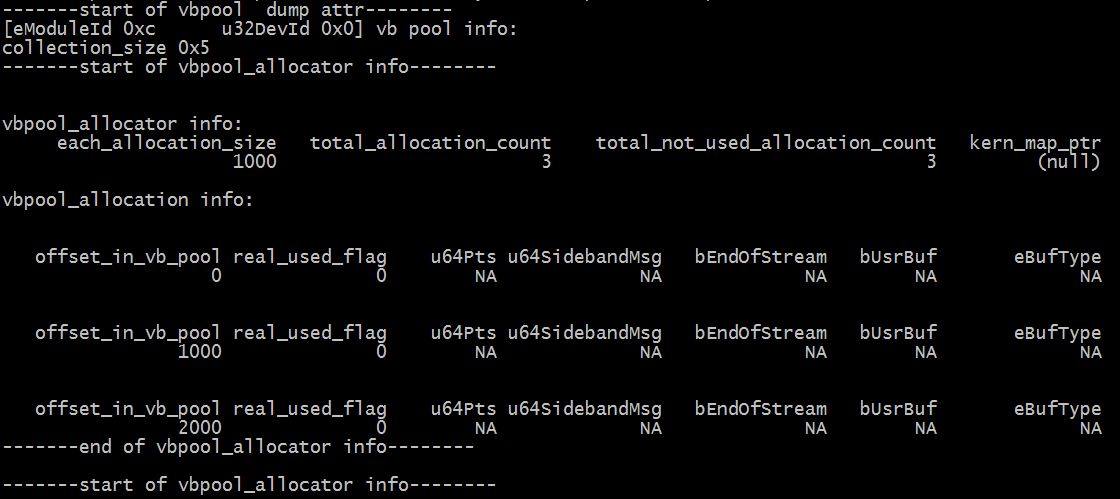
|  |  |
| --- | --- |
| 功能 | 即dump 指定offset和length的data到指定的路径 |
| 命令 | echo [Path] [Offset] [Length] > /proc/mi\_modules/mi\_sys\_mma/ vb\_pool\_global |
| 参数说明 | [Path] 文件要保存的路径,只需提供路径，不要提供具体的文件名，系统会根据参数自动生成对应的文件名。 |
| [Offset] 从该pools的哪个offset开始dump data,是pools的逻辑offset。 |
| [Length] 在该pools里dump的总的数据量的大小。 |
| 举例 | / # echo /mnt/ 0 1024 > /proc/mi\_modules/mi\_sys\_mma/vb\_pool\_global  在 /mnt 下产生 vb\_pool\_\_global\_\_0\_\_1024.bin |

## vb\_pool

### cat

【调试信息】

# cat /proc/mi\_modules/mi\_sys\_mma/vb\_pool\_mi\_divp0



【调试信息分析】

如果有config 某个device的vb pool, 那么会有/proc/mi\_modules/mi\_sys\_mma/vb\_pool\_modnamedevid文件,其中modname形如divp,vpe等，devid形如0,1等。其和vb\_pool\_global的差别在于其对应的是该device,而vb\_pool\_global则是全局共享的vb pool。

上面是以vb\_pool\_mi\_divp0为例，其也有cat命令显示各种信息，也是有且只有”echo path offset length >”这个echo命令来dump数据，这2个都和vb\_pool\_global的类似。

### echo

|  |  |
| --- | --- |
| 功能 | 即dump 指定offset和length的data到指定的路径 |
| 命令 | echo [Path] [Offset] [Length] > /proc/mi\_modules/mi\_sys\_mma/[ModID] |
| 参数说明 | [Path] 文件要保存的路径,只需提供路径，不要提供具体的文件名，系统会根据参数自动生成对应的文件名。 |
| [Offset] 从该pools的哪个offset开始dump data,是pools的逻辑offset。 |
| [Length] 在该pools里dump的总的数据量的大小。 |
| [ModID] vb\_pool的文件名， 如 vb\_pool\_mi\_divp0 |
| 举例 | echo /mnt/ 0 4096 >  /proc/mi\_modules/mi\_sys\_mma/vb\_pool\_mi\_divp0  在/mnt/下 产生vb\_pool\_\_mi\_divp0\_\_0\_\_4096.bin |

## meta

### cat

【调试信息】

# cat /proc/mi\_modules/mi\_sys\_mma/meta

=================== start meta\_info ================================

basic info:

total page count:16 each meta data size:256 total\_count\_with\_metadatasize 256 total\_free\_count\_with\_metadatasize:254

meta info :

ref\_cnt=3

each allocation info:

offset\_in\_pool length phy\_addr va\_in\_kern real\_used\_flag

0x0 0x100 0x2336000 c1736000 0

0x100 0x100 0x2336100 c1736100 0

0x200 0x100 0x2336200 c1736200 0

0x300 0x100 0x2336300 c1736300 0

0x400 0x100 0x2336400 c1736400 0

0x500 0x100 0x2336500 c1736500 0

0x600 0x100 0x2336600 c1736600 0

0x700 0x100 0x2336700 c1736700 0

0x800 0x100 0x2336800 c1736800 0

0x900 0x100 0x2336900 c1736900 0

0xa00 0x100 0x2336a00 c1736a00 0

……

=================== end meta\_info ================================

【调试信息分析】

只要一旦有人用过了meta allocator(有且只有一个meta allocator),那么就会生成/proc/mi\_modules/mi\_sys\_mma/meta文件，且其对应的memory就会完全alloc了出来，等待被使用。

### echo

|  |  |
| --- | --- |
| 功能 | 即dump 指定offset和length的data到指定的路径 |
| 命令 | echo [Path] [Offset] [Length] > /proc/mi\_modules/mi\_sys\_mma/ vb\_pool\_global |
| 参数说明 | [Path] 文件要保存的路径,只需提供路径，不要提供具体的文件名，系统会根据参数自动生成对应的文件名。 |
| [Offset] 从该allocator的哪个offset开始dump data,是allocator的逻辑offset。。 |
| 在该allocator里dump的总的数据量的大小。 |
| 举例 | echo /mt 0 1024 > /proc/mi\_modules/mi\_sys\_mma/meta  在 /mnt 下产生 meta\_\_0\_\_1024.bin |

## mi\_modulenamedevid

### cat

【调试信息】

cat /proc/mi\_modules/mi\_disp/mi\_disp0



【调试信息分析】

cat操作的结果是分为2部分的，第一部分是MI\_SYS提供的针对该device的通用的调试信息，从device、channel、input port、output port四个角度提供；第二部分是该device私有的信息。私有的信息各module的内容别处会阐述，这里SYS角度只阐述通用的信息。

以cat /proc/mi\_modules/mi\_vdisp/mi\_vdisp0为例，得到上面的结果。

【参数说明】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Common info for device | ChnNum | 该device的总的channel 的数目。 |
| EnChnNum | 该device的enabled了的channel的数目。 |
| InPortNum | 该device的input port的数目。 |
| OutPortNum | 该device的output port的数目。 |
| CollectSize | 该dev对应的AllocatorCollection含有的allocator的数目总和。 |
| Common info for channel(only dump enabled channel) | ChnId | Channel 的id值. |
| EnInPNum | 该channel的enabled input port数量。 |
| EnOutPNum | 该channel的enabled output port数量。 |
| MMAHeapName | 如果有SetChnMMAConf的话，该值为对应的mma heap name;如果没有设置的话，该值为NULL。 |
| Input port common info(only dump enabled Input port) | ChnId | 该input port所在的channel id。 |
| PortId | 该input port的id。 |
| SrcFrmrate | Src帧率。 |
| DstFrmrate | Dst帧率。 |
| user\_buf\_quota | 该InputPort的buff数目的Quota。 |
| UsrInjectQ\_cnt | 该InputPort 里的UsrInjectBufQueue里的buff数目。 |
| UsrInjectQ\_size | 该InputPort 里的UsrInjectBufQueue里的buff的总的size。 |
| BindInQ\_cnt | 该InputPort 里的BindInputBufQueue里的buff数目。 |
| BindInQ\_size | 该InputPort 里的BindInputBufQueue里的buff的总的size。 |
| WorkingQ\_cnt | 该InputPort 里的WorkingQueue里的buff数目。 |
| WorkingQ\_size | 该InputPort 里的WorkingQueue里的buff的总的size。 |
| usrLockedInjectCnt | 用户当前拿到了多少个buf。 |
| Input port bind info(only dump enabled Input port) | ChnId | 该input port所在的channel id。 |
| PortId | 该input port的id。 |
| bind\_module\_id | 与该input port 进行了binded的output port所在的module的id。 |
| bind\_module\_name | 与该input port 进行了binded的output port所在的module的name。 |
| bind\_ChnId | 与该input port 进行了binded的output port所在的channel的id。 |
| bind\_PortId | 与该input port 进行了binded的output port的id。 |
| Output port common info (only dump enabled Output port) | ChnId | 该output port所在的channel的id。 |
| PortId | 该output port的id。 |
| BufCntQuota | 该OutputPort的 buff数目的Quota。 |
| usrLockedCnt | 从UsrGetFifoBufQueue里用户实际拿走了多少个buffer。 |
| totalOutPortInUsed | totalOutputPortInUsedBuf数目。 |
| DrvBkRefFifoQ\_cnt | 该OutputPort的DrvBkRefFifoQueue里的buffer数目。 |
| DrvBkRefFifoQ\_size | 该OutputPort的DrvBkRefFifoQueue里的buffer的总size。 |
| UsrGetFifoQ\_cnt | 该OutputPort的UsrGetFifoBufQueue里的buffer数目。 |
| UsrGetFifoQ\_size | 该OutputPort的UsrGetFifoBufQueue里的buffer的总size。 |
| WorkingQ\_cnt | 该OutputPort的WorkingQueue里的buffer数目。 |
| WorkingQ\_size | 该OutputPort的WorkingQueue里的buffer的总size。 |
| Output port BindPeerInputPortList info(only dump enabled Output port) | ChnId | 该output port所在的channel的id。 |
| PortId | 该output port的id。 |
| bind\_module\_id | 与该output port 进行了binded的input port list中的其中一个input port（以下简称该binded input port）所在的module的id。 |
| bind\_module\_name | 该binded input port所在的module的name。 |
| bind\_ChnId | 该binded input port所在的channel id。 |
| bind\_PortId | 该binded input port的id。 |

### echo

|  |  |
| --- | --- |
| 功能 | 获得模块的help信息 |
| 命令 | echo help > /proc/mi\_modules/[ModName]/[ModID] |
| 参数说明 | [ModName] 模块的名字  mi\_disp mi\_gfx mi\_rgn mi\_vdec mi\_vpe  mi\_ai mi\_divp mi\_shadow mi\_vdisp mi\_ao mi\_hdmi mi\_sys mi\_venc mi\_bar mi\_vif |
| [ModID] 模块中对应的文件一般为模块的名字+数字  如mi\_disp0 mi\_gfx0 mi\_rgn0 |
| 举例 | echo help > /proc/mi\_modules/mi\_disp/mi\_disp0  得到mi\_disp0 节点文件的帮助信息 |

|  |  |
| --- | --- |
| 功能 | Dump Port 的信息 |
| 命令 | echo dump\_buffer [ChnID] [PortType] [PortID] [QueueName][Path] [EndMethod] > /proc/mi\_modules/[ModName]/[ModID] |
| 参数说明 | [ChnID] 通道的ID |
| [PortType] iport 或 oport 分别代表input port output port |
| [PortID] 该port的id |
| [QueueName] 如果是input port的话，值只能是"UsrInject" 或者"BindInput"  如果是output port的话，值只能是"UsrGetFifo" |
| [Path] 数据要保存的文件所在的路径。注意是绝对路径，不要提供数据要保存的文件的名字，系统会根据参数自动生成文件名，不同的buffer保存在不同的文件 |
| [EndMethod] dump buffer的结束方法，目前仅支持三类：  (a). "bufnum=xxx":dump指定数目的buffer  (b) "time=xxx"(here unit is ms):dump过程持续time对应的时间  (c). "start/end" pair:用start来开始dump,用end来结束对应的dump,要求这2个命令时其他的5个参数完全一样 |
| [ModName] 模块的名字  mi\_disp mi\_gfx mi\_rgn mi\_vdec mi\_vpe  mi\_ai mi\_divp mi\_shadow mi\_vdisp mi\_ao mi\_hdmi mi\_sys mi\_venc mi\_bar mi\_vif |
| [ModID] 模块中对应的文件 一般为 一般为模块的名字+数字 |
| 举例 | echo dump\_buffer 0 iport 0 BindInput /mnt bufnum=1 > /proc/mi\_modules/mi\_disp/mi\_disp0  dump disp模块 inport 的数据 |

## mi\_dump\_buffer\_delay\_worker

### cat

【调试信息】

cat /proc/mi\_modules/mi\_dump\_buffer\_delay\_worker

delay\_worker\_id module\_name force\_stop dev\_id chn\_id port\_type port\_id

0 mi\_disp 0 0 0 inport 0

Queue\_name stored\_dir dump\_method dump\_method\_value

BindInput /mnt bunfnum 2

【调试信息分析】

/proc/mi\_modules/mi\_dump\_buffer\_delay\_worker文件是和mi\_modulenamedevid相连的一个概念，dump device里的Queue里的buffer采用的是delay worker的方式实现的，可以通过对mi\_dump\_buffer\_delay\_worker的cat操作查看还有哪些delay worker正在进行中，echo force\_stop\_dump delay\_worker\_id强制结束指定的一个delay worker过程。

【参数说明】

|  |  |
| --- | --- |
| 参数 | 描述 |
| delay\_worker\_id | 该delay worker 的id。如果delay worker完成的话，该id会回收，给后续其他的delay worker作为id使用。 |
| module\_name | 该delay worker所对应的module的id。 |
| force\_stop | 该delay worker当前是否处于强制结束阶段，1表示已经被设置了强制结束，0表示未被设置。 |
| dev\_id | 该delay worker所对应的device id。 |
| chn\_id | 该delay worker所对应的channel id。 |
| port\_type | 该delay worker所对应的port的type,是input port,还是output port |
| port\_id | 该delay worker所对应的port的id |
| Queue\_name | 该delay worker所对应的Queue的name |
| stored\_dir | 该delay worker所对应的要dump的data所要存放的绝对路径（不含最终存放的文件名） |
| dump\_method | 该delay worker所对应的dump方法:bufnum，还是time,还是start |
| dump\_method\_value | Dump方法所对应的值. |

### echo

|  |  |
| --- | --- |
| 功能 | 强制结束delay worker |
| 命令 | echo force\_stop\_dump [WorkID] >/proc/mi\_modules/mi\_dump\_buffer\_delay\_worker |
| 参数说明 | [WorkID] delay\_work 的ID |
| 举例 | echo force\_stop\_dump 0 > /proc/mi\_modules/mi\_dump\_buffer\_delay\_worker  强制停止 delay\_work 0 |

## module version

### cat

【调试信息】

# cat /proc/mi\_modules/mi\_ai/module\_version\_file

MStar Module version: project\_commit.sdk\_commit.build\_time c1799df.fd2d52b.20171225180033

# cat /proc/mi\_modules/mi\_global\_info

…..

MStar Module version: project\_commit.sdk\_commit.build\_time cb68bfd.44aca45.20171226100257

….

【调试信息分析】

xxxx\_version\_file提供了份version信息， /proc/mi\_modules/mi\_global\_info里我们也提供了份version信息，不过其本质上指的是mi\_sys模块的version信息, 上面只是以mi\_vi 和 mi\_global\_info 为例子，每个模块都有自己对应的version file.

【参数说明】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Version Info | project\_commit | 模块编译ko时整包的project commit 信息，如果单独替换ko时，模块基于的commit有变化，那么该ko的version里得到的commit也会跟着变更 |
| sdk\_commit | 模块编译ko时整包的sdk commit 信息，如果单独替换ko时，模块基于的commit有变化，那么该ko的version里得到的commit也会跟着变更 |
| build\_time | 时间指的是实际的build时间，精确到秒，即使make clean;make image整体来build的话，各个ko的时间也会有差别。 |

# AI

## cat

【调试信息】

# cat proc/mi\_modules/mi\_ai/mi\_ai0

========================= Private AI0 Info =======================

----Start AIDev0Attr--------------------------------------

AiDev WorkMode SampR BitWidth SondMod PtNumPerFrm

0 i2s-mas 16000 16bit mono 1536

----End AI Dev0 Attr---------------------------------------------

----Start AI Chn0 STATUS-------------------------------------------

AiDev AiChn bReSmp InSampR OutSampR

0 0 0 16000 0

----AI CHN0 VQE STATUS-----------------------------------------------

AiDev AiChn bVqe WorkRate PoiNum

0 0 0 0 0

-----Ai CHN0 ANR of VQE STATUS---------------------------------------

AiDev AiChn bAnr NrIntensity

0 0 0 0

-----AI CHN0 AGC of VQE STATUS----------------------------------------

AiDev AiChn bAgc TargetLevel NoiseFloor

0 0 0 0 0

-----AI CHN0 EQ of VQE STATUS------------------------------------------

AiDev AiChn bEq

0 0 0

-----AI CHN0 HPF of VQE STATUS------------------------------------------

AiDev AiChn bHpf HpfFreq

0 0 0 0

-----AI CHN0 AEC of VQE STATUS------------------------------------------

AiDev AiChn bAec AecIntensity

0 0 0 0

-----End AI Chn0 STATUS--------------------------------------------------

【调试信息分析】

记录当前AI的使用状况以及device属性、channel属性，可以动态地获取到这些信息，方便调试和测试。

【参数说明】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| AI Device Attr  (AI 设备属性) | AiDev | AI设备号:  取值范围：[0~1] |
| WorkMode | AI 工作模式  i2s-mas: I2S master  i2s-sla: I2S slave  tdm-mas: TDM master |
| SamR | Sample Rate(采样率):  8000, 16000, 32000, 48000 |
| BitWidth | 采样精度:  16bit, 24bit |
| SondMod | 声音模式  mono：单声道  stereo：立体声 |
| PtNumPerFrm | 每帧的采样点个数:  128倍数 |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| AI Chn Status  (AI 通道信息) | AiDev | AI设备号:  取值范围：[0~1] |
| AiChn | AI 通道号:  取值范围：[0~1] |
| bReSmp | 该信道是否启用了重采样功能  1: Enable  0: Disable |
| InSampR | 该通道重采样时，输入给重采样的数据帧的采样率 |
| OutSampR | 该通道重采样时，重采样后的音频帧采样率 |
| AI CHN VQE STATUS  (AI 通道VQE信息 ) | AiDev | AI设备号:  取值范围：[0~1] |
| AiChn | AI 通道号:  取值范围：[0~1] |
| bVqe | 该通道是否启用VQE  1: Enable  0: Disable |
| WorkRate | 工作采样率:  8000, 16000 |
| PoiNum | VQE处理的数据帧长中的采样点数目: 128 |
| AI CHN ANR of VQE STATUS  (AI通道ANR 信息) | AiDev | AI设备号:  取值范围：[0~1] |
| AiChn | AI 通道号:  取值范围：[0~1] |
| bAnr | 该通道是否启用ANR  1: Enable  0: Disable |
| NrIntensity | 降噪力度配置:  取值为[0,30] |
| AI CHN AGC of VQE STATUS  (AI 通道AGC信息) | AiDev | AI设备号:  取值范围：[0~1] |
| AiChn | AI 通道号:  取值范围：[0~1] |
| bAgc | 该通道是否启用AGC  1: Enable  0: Disable |
| TargetLevel | 目标电平: 范围：[-80 ~ 0]dB ；调整步长 1dB |
| NoiseFloor | 噪声底线: 范围：[-65 ~ -20]dB ；调整步长 1dB |
| AI CHN EQ of VQE STATUS  (AI 通道EQ 信息) | AiDev | AI设备号:  取值范围：[0~1] |
| AiChn | AI 通道号:  取值范围：[0~1] |
| bEq | 该通道是否启用EQ  1: Enable  0: Disable |
| AI CHN0 HPF of VQE STATUS | AiDev | AI设备号:  取值范围：[0~1] |
| AiChn | AI 通道号:  取值范围：[0~1] |
| bHpf | 该通道是否启用HPF  1: Enable  0: Disable |
| HpfFreq | 截止频率（Hz）:  80, 120, 150 |
| AI CHN AEC of VQE STATUS  (AI 通道AEC信息) | AiDev | AI设备号:  取值范围：[0~1] |
| AiChn | AI 通道号:  取值范围：[0~1] |
| bAec | 该通道是否启用HPF  1: Enable  0: Disable |
| AecIntensity | 回声消除力度配置，取值范围：[0,3] |

# AO

## cat

【调试信息】

# cat proc/mi\_modules/mi\_ao/mi\_ao0

=============================Private AO0 Info =====================

-----Start AO Dev0 Attr-------------------------------------------------

AoDev WorkMode SampR BitWidth SondMod PtNumPerFrm

0 i2s-mas 32000 16bit mono 768

bMute VolumedB TotalWrFrmCnt

0 0 583

-----End AO Dev0 Attr-----------------------------------------------------

-----Start AO CHN0 STATUS---------------------------------------------------

AoDev AoChn bReSmp InSampR OutSampR

0 0 0 0 32000

-----AO CHN0 VQE STATUS--------------------------------------------------

AoDev AoChn bVqe WorkRate PoiNum

0 0 0 0 0

-----AO CHN0 ANR of VQE STATUS-------------------------------------------

AoDev AoChn bAnr NrIntensity

0 0 0 0

-----AO CHN0 AGC of VQE STATUS-------------------------------------------

AoDev AoChn bAgc TargetLevel NoiseFloor

0 0 0 0 0

-----AO CHN0 EQ of VQE STATUS---------------------------------------------

AoDev AoChn bEq

0 0 0

-----AO CHN0 HPF of VQE STATUS--------------------------------------------

AoDev AoChn bHpf HpfFreq

0 0 0 0

-----End AO Chn0 STATUS---------------------------------------------------

【调试信息分析】

记录当前AO的使用状况以及device属性、channel属性，可以动态地获取到这些信息，方便调试和测试。

【参数说明】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| AO Device Attr  (AO设备属性) | AoDev | AO设备号:  取值范围：[0~2] |
| WorkMode | AO 工作模式  i2s-mas: I2S master  i2s-sla: I2S slave  tdm-mas: TDM master |
| SamR | Sample Rate(采样率):  8000, 16000, 32000, 48000 |
| BitWidth | 采样精度:  16bit, 24bit |
| SondMod | 声音模式  mono：单声道  stereo：立体声 |
| PtNumPerFrm | 每帧的采样点个数:  128倍数 |
|  | bMute | AO 设备静音功能是否开  1: Enable  0: Disable |
| VolumedB | AO 设备的输出音量大, 取值范围: [-114 ~ +12] dB |
| TotalWrFrmCnt | AO设备目前输出的Frame数目 |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| AO CHN Status  (AO 通道信息) | AoDev | AO设备号:  取值范围：[0~2] |
| AoChn | AO 通道号:  取值范围：[0~2] |
| bReSmp | 该信道是否启用了重采样功能  1: Enable  0: Disable |
| InSampR | 该通道重采样时，输入给重采样的数据帧的采样率 |
| OutSampR | 该通道重采样时，重采样后的音频帧采样率 |
| AO CHN VQE STATUS  (AO 通道VQE信息 ) | AoDev | AO设备号:  取值范围：[0~2] |
| AoChn | AO 通道号:  取值范围：[0~2] |
| bVqe | 该通道是否启用VQE  1: Enable  0: Disable |
| WorkRate | 工作采样率:  8000, 16000 |
| PoiNum | VQE处理的数据帧长中的采样点数目: 128 |
| AO CHN ANR of VQE STATUS  (AO通道ANR 信息) | AoDev | AO设备号:  取值范围：[0~2] |
| AoChn | AO 通道号:  取值范围：[0~2] |
| bAnr | 该通道是否启用ANR  1: Enable  0: Disable |
| NrIntensity | 降噪力度配置:  取值为[0,30] |
| AO CHN AGC of VQE STATUS  (AO 通道AGC信息) | AoDev | AO设备号:  取值范围：[0~2] |
| AoChn | AO通道号:  取值范围：[0~2] |
| bAgc | 该通道是否启用AGC  1: Enable  0: Disable |
| TargetLevel | 目标电平: 范围：[-80 ~ 0]dB ；调整步长 1dB |
| NoiseFloor | 噪声底线: 范围：[-65 ~ -20]dB ；调整步长 1dB |
| AO CHN EQ of VQE STATUS  (AO 通道EQ 信息) | AoDev | AO设备号:  取值范围：[0~2] |
| AoChn | AO通道号:  取值范围：[0~2] |
| bEq | 该通道是否启用EQ  1: Enable  0: Disable |
| AO CHN0 HPF of VQE STATUS | AoDev | AO设备号:  取值范围：[0~2] |
| AoChn | AO通道号:  取值范围：[0~2] |
| bHpf | 该通道是否启用HPF  1: Enable  0: Disable |
| HpfFreq | 截止频率（Hz）:  80, 120, 150 |

## echo

|  |  |
| --- | --- |
| 功能 | 动态启用/关闭AO设备静音模式 |
| 命令 | echo setaomute [Status] > / proc/mi\_modules/mi\_ao/mi\_ao[ID] |
| 参数说明 | [Status] ON 开启静音模式 OFF 关闭静音模式 |
| [ID] 设备号 [0~2]  0 LINEOUT  1 I2S  2 HDMI  3 ALL 所有设备 |
| 举例 | echo setaomute ON > /proc/mi\_modules/mi\_ao/mi\_ao0  Audio out 设备0 开启静音(LINEOUT 静音) |

|  |  |
| --- | --- |
| 功能 | 动态修改AO volume大小 |
| 命令 | echo setaovolume [Steps] > /proc/mi\_modules/mi\_ao/mi\_ao[ID] |
| 参数说明 | [Steps] 调整步长 [-114~12]dB |
| [ID] 设备号 [0~3]  0 LINEOUT  1 I2S  2 HDMI  3 ALL 所有设备 |
| 举例 | echo setaovolume 10 > proc/mi\_modules/mi\_ao/mi\_ao2  Audio out 设备 2 音量设置为10dB(HDMI 音量为10dB) |

|  |  |
| --- | --- |
| 功能 | 打印每秒钟接收到的数据大小 |
| 命令 | echo checkszie [Status] > /proc/mi\_modules/mi\_ao/mi\_ao[ID] |
| 参数说明 | [Status] ON 开启 OFF 关闭 |
| [ID] 设备号 [0~3]  0 LINEOUT  1 I2S  2 HDMI  3 ALL 所有设备 |
| 举例 | echo checksize ON > /proc/mi\_modules/mi\_ao/mi\_ao0  MI\_AO Dev0 check size ON success.  MI\_AO Dev0 Size Per Second: 1560576  MI\_AO Dev0 Size Per Second: 190464  MI\_AO Dev0 Size Per Second: 193536  MI\_AO Dev0 Size Per Second: 190464  MI\_AO Dev0 Size Per Second: 190464  LINEOUT 打印每秒接收到的数据大小  第一次的数据不准确，后面的数据是正常的, 数据的单位是Byte |

|  |  |
| --- | --- |
| 功能 | dump pcm 数据 |
| 命令 | echo dump [Path] [Count] > /proc/mi\_modules/mi\_ao/mi\_ao[ID] |
| 参数说明 | [Path] 路径 文件名会自动生成 |
| [Count] dump 的次数 |
| [ID] 设备号 [0~3]  0 LINEOUT  1 I2S  2 HDMI  3 ALL 所有设备 |
| 举例 | echo dump /mnt 100 > /proc/mi\_modules/mi\_ao/mi\_ao0  printk in \_MI\_SYS\_IMPL\_Common\_WriteProc  MaxCount:100  Path:/mnt/Dev0\_48000\_STERO.pcm  Count:1, Size: 3072, TotalSize:3072  Count:2, Size: 3072, TotalSize:6144  Count:3, Size: 3072, TotalSize:9216  ….  生成的文件名为 设备ID+采样率+声音模式 |

# VIF

## cat

【调试信息】

#cat /proc/mi\_modules/mi\_vif/mi\_vif0

------------------------- start dump vif dev 0 info --------------------------------

u32VifDevIdx: 0

eIntfMode: 0

eWorkMode:0

eClkEdge:0

au32CompMask [0]: ff000000 [1]:00000000

as32AdChnId [0]: -1 [1]:-1 [2]: -1 [3]:-1

bDataRev:0

u32VifDevIdx: 1

eIntfMode: 0 eWorkMode:0 eClkEdge:0

au32CompMask [0]: ff000000 [1]:00000000

as32AdChnId [0]: -1 [1]:-1 [2]: -1 [3]:-1

bDataRev:0

------- start dump vif dev:0 chn:0 port 0 : --------------------------------

s32X: 0 s32Y: 0 u32Width: 720 u32Height: 240

u32HalBufferHold: 2 eFormat: 9 u32VbFail: 0

【调试信息分析】

记录当前VIF的使用状况以及device属性、OutPort属性、可以动态地获取到这些信息，方便调试和测试。

【参数说明】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Dev info | u32VifDevIdx | Dev号 |
| eIntfMode | 输入数据的协议 |
| eWorkMode | 工作模式 |
| eClkEdge | 时钟边沿触发模式 |
| au32CompMask | 分量掩码配置 |
| as32AdChnId | 通道映射关系，一般设置为-1 |
| bDataRev | 反向连接 |

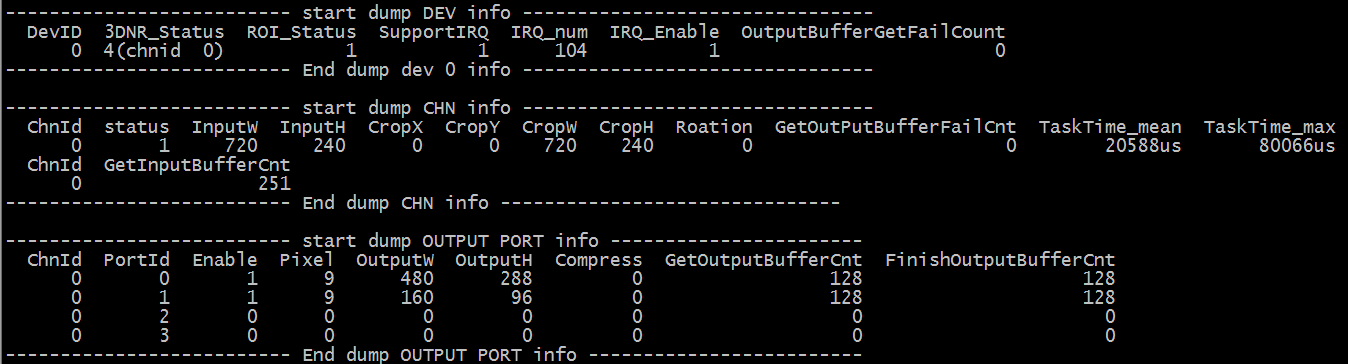
|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| OutPort Info | s32X | X坐标 |
| s32Y | Y坐标 |
| u32Width | 宽度 |
| u32Height | 高度 |
| u32HalBufferHold | Mhal层保存的Buffer数量 |
| eFormat | 格式 |
| u32VbFail | Get Buffer 错误的此数 |

# VPE

## cat

【调试信息】

# cat /proc/mi\_modules/mi\_vpe/mi\_vpe0



【调试信息分析】

记录当前VPE的使用状况以及device属性、OutPort属性、可以动态地获取到这些信息，方便调试和测试。

【参数说明】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Device  Info | DevID | 0, 只有一个Device, ID为0 |
| 3DNR\_Status | E\_MI\_VPE\_3DNR\_STATUS\_INVALID = 0,  E\_MI\_VPE\_3DNR\_STATUS\_IDLE,  E\_MI\_VPE\_3DNR\_STATUS\_NEED\_UPDATE,  E\_MI\_VPE\_3DNR\_STATUS\_RUNNING,  E\_MI\_VPE\_3DNR\_STATUS\_UPDATED,  E\_MI\_VPE\_3DNR\_STATUS\_NUM,  chnid 为当前3DNR正在处理的channelID |
| ROI\_Status | E\_MI\_VPE\_ROI\_STATUS\_INVALID = 0,  E\_MI\_VPE\_ROI\_STATUS\_IDLE, //---> free  E\_MI\_VPE\_ROI\_STATUS\_NEED\_UPDATE, // user call ROI  E\_MI\_VPE\_ROI\_STATUS\_RUNNING, // ROI in cmdQ  E\_MI\_VPE\_ROI\_STATUS\_UPDATED, // ISR ROI finish  E\_MI\_VPE\_ROI\_STATUS\_NUM, |
| SupportIRQ | 0: 当前不支持IRQ  1: 当前支持IRQ.  (在procfs中可以控制该值开关中断) |
| IRQ\_num | 中断号 |
| IRQ\_Enable | 0：IRQ失能  1：IRQ使能 |
| OutputBufferGetFailCnt | 获取outputbuffer总过失败的次数 |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Channel Info | ChnId | Channel ID 0~63 |
| status | E\_MI\_VPE\_CHANNEL\_STATUS\_INITED = 0,  E\_MI\_VPE\_CHANNEL\_STATUS\_START,  E\_MI\_VPE\_CHANNEL\_STATUS\_STOP,  E\_MI\_VPE\_CHANNEL\_STATUS\_DESTROYED,  E\_MI\_VPE\_CHANNEL\_STATUS\_NUM, |
| InputW  InputH | 输入的宽高 |
| CropX CropY  CropW CropH | Crop的位置 |
| Roation | E\_MI\_SYS\_ROTATE\_NONE, //Rotate 0 degrees  E\_MI\_SYS\_ROTATE\_90, //Rotate 90 degrees  E\_MI\_SYS\_ROTATE\_180, //Rotate 180 degrees  E\_MI\_SYS\_ROTATE\_270, //Rotate 270 degrees  E\_MI\_SYS\_ROTATE\_NUM, |
| GetOutPutBufferFailCnt | Channel获取outputbuffer失败的次数 |
| TaskTime\_mean | 获取inputbuffer到finishoutputbuffer的平均时间 |
| TaskTime\_max | 获取inputbuffer到finishoutputbuffer的最大时间 |
| GetInputBufferCnt | channel拿到inputbuffer的数量 |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Outputport Info | ChnId | Channel ID 0~63 |
| PortId | 0~3 |
| Enable | 0:port disable  1:port enable |
| Pixel | E\_MI\_SYS\_PIXEL\_FRAME\_YUV422\_YUYV = 0,  E\_MI\_SYS\_PIXEL\_FRAME\_ARGB8888,  E\_MI\_SYS\_PIXEL\_FRAME\_ABGR8888,  E\_MI\_SYS\_PIXEL\_FRAME\_RGB565,  E\_MI\_SYS\_PIXEL\_FRAME\_ARGB1555,  E\_MI\_SYS\_PIXEL\_FRAME\_I2,  E\_MI\_SYS\_PIXEL\_FRAME\_I4,  E\_MI\_SYS\_PIXEL\_FRAME\_I8,  E\_MI\_SYS\_PIXEL\_FRAME\_YUV\_SEMIPLANAR\_422,  E\_MI\_SYS\_PIXEL\_FRAME\_YUV\_SEMIPLANAR\_420,  E\_MI\_SYS\_PIXEL\_FRAME\_YUV\_MST\_420,  //vdec mstar private video format  E\_MI\_SYS\_PIXEL\_FRAME\_YC420\_MSTTILE1\_H264,  E\_MI\_SYS\_PIXEL\_FRAME\_YC420\_MSTTILE2\_H265,  E\_MI\_SYS\_PIXEL\_FRAME\_YC420\_MSTTILE3\_H265,  E\_MI\_SYS\_PIXEL\_FRAME\_FORMAT\_MAX, |
| OutputW  OutputH | 输出宽  输出高 |
| Compress | E\_MI\_SYS\_COMPRESS\_MODE\_NONE,//no compress  E\_MI\_SYS\_COMPRESS\_MODE\_SEG,//compress unit is 256 bytes as a segment  E\_MI\_SYS\_COMPRESS\_MODE\_LINE,//compress unit is the whole line  E\_MI\_SYS\_COMPRESS\_MODE\_FRAME,//compress unit is the whole frame  E\_MI\_SYS\_COMPRESS\_MODE\_BUTT, //number |
| GetOutputBufferCnt | 获取outputbuffer的数量 |
| FinishOutputBufferCnt | 处理完outputbuffer的数量 |

## echo

|  |  |
| --- | --- |
| 功能 | 控制使用cmdq方式，设置下次启动VPE的时候生效。 |
| 命令 | echo disable\_cmdq [Status] > /proc/mi\_modules/mi\_vpe0 |
| 参数说明 | [Status] ON 下寄存器指令来一个下一个 OFF 使用造剧本的方式下寄存器指令 |
| 举例 | 无 |

|  |  |
| --- | --- |
| 功能 | 控制是否使用IRQ，设置下次启动VPE的时候生效 |
| 命令 | echo disable\_irq [Status] > /proc/mi\_modules/mi\_vpe0 |
| 参数说明 | [Status] ON 不使用IRQ中断 OFF 使用IRQ中断 |
| 举例 | 无 |

|  |  |
| --- | --- |
| 功能 | 打印接收到每一帧的PTS，Debug Frame 顺序问题，正常情况下PTS是递增的 |
| 命令 | echo checkframepts [ChnID] [Status] > /proc/mi\_modules/mi\_vpe/mi\_vpe0 |
| 参数说明 | [ChnID] 通道号 [0~63] |
| [Status] ON 开始打印 OFF 停止打印 |
| 举例 | echo checkframepts 1 ON > /proc/mi\_modules/mi\_vpe/mi\_vpe0  [MI VPE PROCFS]:ChnID 1, receive buffer ID = 111, PTS = 31461  [MI VPE PROCFS]:ChnID 1, receive buffer ID = 112, PTS = 31467  [MI VPE PROCFS]:ChnID 1, receive buffer ID = 113, PTS = 31471  [MI VPE PROCFS]:ChnID 1, receive buffer ID = 114, PTS = 31475  [MI VPE PROCFS]:ChnID 1, receive buffer ID = 115, PTS = 31479  [MI VPE PROCFS]:ChnID 1, receive buffer ID = 116, PTS = 31485  [MI VPE PROCFS]:ChnID 1, receive buffer ID = 117, PTS = 31493  receive buffer ID：每个channel接收到buffer的ID。  PTS: FrameBuffer的PTS。 |

|  |  |
| --- | --- |
| 功能 | 统计1秒时间内GetBuffer的数量，以及每拿到两个inputbuffer之间的间隔时间的平均值，最大值，最小值，Debug丢帧问题。 |
| 命令 | echo statintervaltime [ChnID] [Status] > /proc/mi\_modules/mi\_vpe/mi\_vpe0 |
| 参数说明 | [ChnID] 通道号 [0~63] |
| [Status] ON 开始统计和打印 OFF 停止统计和打印 |
| 举例 | echo statintervaltime [chnID] ON/OFF > /proc/mi\_modules/mi\_vpe/mi\_vpe0  [MI VPE PROCFS]:ChnlID 1,1s FrameCnt:15,Average time: 62666 us,MAX 120006us,100030us,100024us,MIN 19974 us,19993 us,19999 us  [MI VPE PROCFS]:ChnlID 1,1s FrameCnt:14,Average time: 62857 us,MAX 139973us,100024us,80005 us,MIN 19956 us,20039 us,40008 us  [MI VPE PROCFS]:ChnlID 1,1s FrameCnt:11,Average time: 74546 us,MAX 200001us,199984us,100020us,MIN 19973 us,19984 us,19994 us  [MI VPE PROCFS]:ChnlID 1,1s FrameCnt:12,Average time: 69998 us,MAX 119952us,99984 us,80059 us,MIN 19966 us,39962 us,40005 us  [MI VPE PROCFS]:ChnlID 1,1s FrameCnt:11,Average time: 74546 us,MAX 200009us,119988us,100017us,MIN 19964 us,20018 us,39991 us  FrameCnt: 1秒时间GetBuffer的数量  Average time： 拿到两个buffer之间间隔时间的平均时间  MAX：间隔时间最大的三个值  MIN: 间隔时间最小的三个值 |

# VENC

## cat

【调试信息】

#cat /proc/mi\_modules/mi\_venc/mi\_venc2

------------------------- Start dump VENC 2 Dev info --------------------------

DevId CmdqId CmdqSize CmdqCnt IrqNum FromIsr

2 0 0 0 100 0

------------------------- Start dump VENC 2 CHN info --------------------------

ChnId RefMemPA RefMemVA RefMemBufSize AlMemPA AlMemVA AlMemBufSize

0 87040000 (null) 200 87035904 cebc9000 100

ChnId bStart bSelected FrameCnt

0 1 0 0

----------------------------- InputPort of dev: 2 -----------------------------

ChnId Width Height SrcFrmRate MaxW MaxH FrameCnt DropFrameCnt

0 352 288 30 352 288 0 0

---------------------------- OutputPort of dev: 2 -----------------------------

ChnId CODEC Profile BufSize RefNum bByFrame FrameCnt DropFrameCnt

0 H264 0 0 0 0 0 0

ChnId RateCtl GOP Bitrate QP.I QP.P

0 FixQP 30 0 45 45

【调试信息分析】

记录当前某VENC device的使用状况，以及device属性、layer属性、inputport属性，可以动态地获取到这些信息，方便调试和测试。

【参数说明】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Dev info | DevId | 设备号，作为Dev ID使用 (对映内核MI\_VENC\_Dev\_e) |
| CmdqId | Command Queue号 |
| CmdqSize | Command Queue单帧大小 (bytes) |
| CmdqCnt | Command Queue数量 (number of frames) |
| IrqNum | 中断号 |
| FromIsr | 是否有未处理ISR信号：0：无 1：中 |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| CHN info | ChnId | Channel ID 通道号 |
| RefMemPA | 底层driver要求的硬件物理位址 |
| RefMemVA | 底层driver要求的硬件虚拟位址 |
| RefMemBufSize | 底层driver要求的硬件内存大小 |
| AlMemPA | 底层driver要求的CPU物理位址 |
| AlMemVA | 底层driver要求的CPU虚拟位址 |
| AlMemBufSize | 底层driver要求的CPU内存大小 |
| bStart | 是否已开始接收数据 |
| bSelected | 是否有处理中的数据 |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Inputport info | ChnId | Channel ID 通道号 |
| Width | 画面宽度 (pixels) |
| Height | 画面高度 (pixels) |
| SrcFrmRate | 来源每秒帧数，通常以此作为rate control的参数 |
| MaxW | 最大画面宽度 (pixels) |
| MaxH | 最大画面高度 (pixels) |
| FrameCnt | 已处理的帧数 |
| DropFrameCnt | 已丢弃的帧数 |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Outputport info | ChnId | Channel ID 通道号 |
| CODEC | 使用何种CODEC: H264,H265,MJPG |
| Profile | Profile值 |
| BufSize | 由user设定的每帧最大缓存 (bytes) |
| RefNum | 最大参考帧数 |
| bByFrame | 后级是否一次取走一帧？ 0：否，通常代表按包取数据 1：是 |
| FrameCnt | 已处理的帧数 |
| DropFrameCnt | 已丢弃的帧数 |
| Outputport Rate Control Info | RateCtl | Rate Control的算法：CBR, VBR, FixQp等 |
| GOP | Group of Picture大小 |
| Bitrate | CBR, VBR时，最大的目标码率。(bits per second) |
| QP.I | FixQp 时，I帧的QP |
| QP.P | FixQp 时，P帧的QP |
| StatTime | CBR, VBR时，算法操作的时间参数 |
| FluctuateLevel | CBR时，允许的摆荡程度 |
| MaxQp | VBR时，允许的最大QP |
| MinQp | VBR时，允许的最小QP |

## echo

|  |  |
| --- | --- |
| 功能 | 开启或者关闭command queue 开机初始设定或者Demo退出后，下次生效 |
| 命令 | echo enable\_cmdq [Status] > /proc/mi\_modules/mi\_venc/mi\_venc[DevID] |
| 参数说明 | [Status] 1 开启 0 关闭 |
| [DevID] 设备ID [0~3] |
| 举例 | echo enable\_cmdq 0 –h > /proc/mi\_modules/mi\_venc/mi\_venc0  关闭devcie 0的cmdq |

|  |  |
| --- | --- |
| 功能 | 开启或者关闭中断 开机初始设定或者Demo退出后，下次生效 |
| 命令 | echo enable\_irq [Status] > /proc/mi\_modules/mi\_venc/mi\_venc[DevID] |
| 参数说明 | [Status] 1 开启 0 关闭 |
| [DevID] 设备ID [0~3] |
| 举例 | echo enable\_irq 0 –h > /proc/mi\_modules/mi\_venc/mi\_venc0  关闭 devide 0 的中断 |

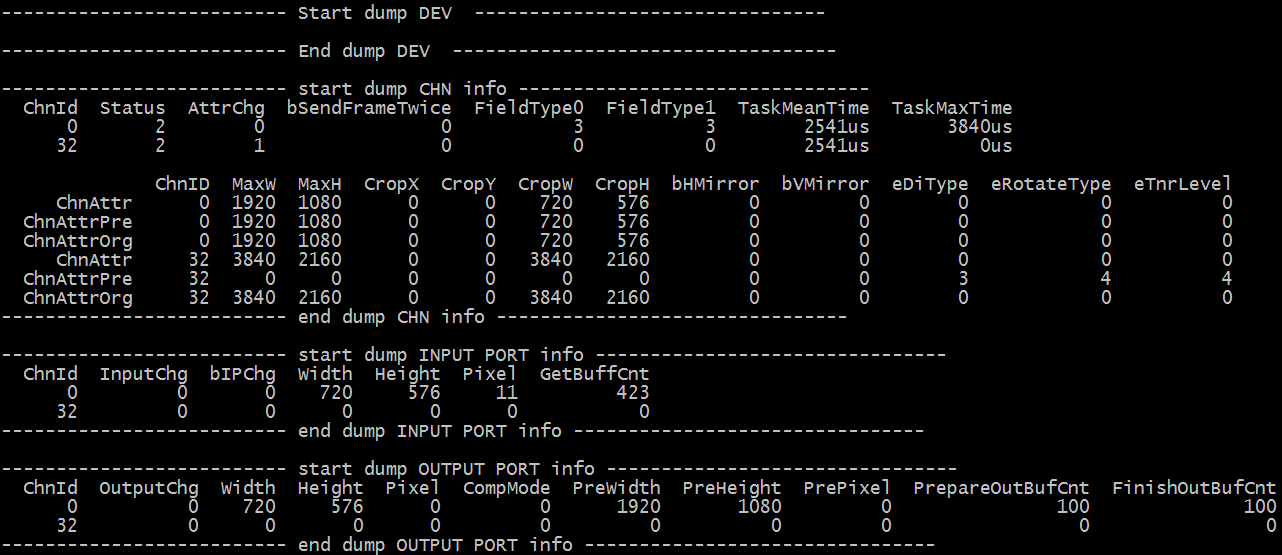
|  |  |
| --- | --- |
| 功能 | 设定模块工程Log等级 直接修改输出Log等级，立即生效 |
| 命令 | echo dbg\_level [Level] > /proc/mi\_modules/mi\_venc/mi\_venc[DevID] |
| 参数说明 | [Level] [0~3]  0: None  1: Error  2: option 1+ warning  3: option 2+ information  [DevID] 设备号 [0~3] |
| 举例 | 无 |

# DIVP

## cat

【调试信息】

# cat /proc/mi\_modules/mi\_divp/mi\_divp0



【调试信息分析】

记录当前某DIVP device的使用状况，以及device属性、layer属性、inputport属性，可以动态地获取到这些信息，方便调试和测试。

【参数说明】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| channelinfo | ChnId | 0~31  32 Capture Channel 不对上层开放 |
| Status | 0 INITED  1 CREATED  2 STARTED  3 STOPED  4 DISTROYED |
| AttrChg | 0：not change  1：changed |
| bSendFrameTwice | 0: not sendFrameTwice  1:SendFrameTwice  \*if the channel's TNR is opend, when I/P mode change or the first frame is comming in, the frame need to be send twice to avoid showwing garbage. |
| FieldType0,  FieldType1  (last two fields's type) | 0 NONE, //< no field.  1 TOP, //< Top field only.  2 BOTTOM, //< Bottom field only.  3 BOTH, //< Both fields.  4 NUM |
| TaskMeanTime | GetBuffer到Releasebuffer的平均时间 |
| TaskMaxTime | GetBuffer到Releasebuffer的最大时间 |
| ChnAttr | 当前channel属性 |
| ChnAttrPre | channel的上一次设置属性 |
| ChnAttrOrg | channel的原始设置属性 |
| MaxW，MaxH | 设置最大宽高 |
| CropX，CropY，CropW，CropH | 上层设置的crop位置 |
| bHMirror bVMirror | 水平翻转使能  垂直翻转使能  0:disable  1:enable |
| eDiType | 0 OFF,//off  1 2D,///2.5D DI  2 3D,///3D DI  3 NUM, |
| eRotateType | 0 NONE, //Rotate 0 degrees  1 90, //Rotate 90 degrees  2 180, //Rotate 180 degrees  3 270, //Rotate 270 degrees  4 NUM, |
| eTnrLevel | 0 OFF,  1 LOW,  2 MIDDLE,  3 HIGH,  4 NUM, |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| inputport info | ChnId | Channel ID 0~31 |
| InputChg | input port 属性是否发生变化 |
| bIPChg | 0 隔行和逐行之间没有发生变化  1 隔行切换到逐行，or 逐行到隔行 |
| Width | input port width |
| Height | input port height |
| Pixel | E\_MI\_SYS\_PIXEL\_FRAME\_YUV422\_YUYV = 0,  E\_MI\_SYS\_PIXEL\_FRAME\_ARGB8888,  E\_MI\_SYS\_PIXEL\_FRAME\_ABGR8888,  E\_MI\_SYS\_PIXEL\_FRAME\_RGB565,  E\_MI\_SYS\_PIXEL\_FRAME\_ARGB1555,  E\_MI\_SYS\_PIXEL\_FRAME\_I2,  E\_MI\_SYS\_PIXEL\_FRAME\_I4,  E\_MI\_SYS\_PIXEL\_FRAME\_I8,  E\_MI\_SYS\_PIXEL\_FRAME\_YUV\_SEMIPLANAR\_422,  E\_MI\_SYS\_PIXEL\_FRAME\_YUV\_SEMIPLANAR\_420,  E\_MI\_SYS\_PIXEL\_FRAME\_YUV\_MST\_420,  //vdec mstar private video format  E\_MI\_SYS\_PIXEL\_FRAME\_YC420\_MSTTILE1\_H264,  E\_MI\_SYS\_PIXEL\_FRAME\_YC420\_MSTTILE2\_H265,  E\_MI\_SYS\_PIXEL\_FRAME\_YC420\_MSTTILE3\_H265,  E\_MI\_SYS\_PIXEL\_FRAME\_FORMAT\_MAX |
| GetBuffCnt | 获取buffer的数量 |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Outputport Info | ChnId | Channel ID 0~31 |
| OutputChg | Output 属性是否发生变化  0: not change  1: changed |
| Width Height | output port宽高 |
| Pixel | 同input pixel |
| CompMode | 0 //no compress  1 //compress unit is 256 bytes as a segment  2 //compress unit is the whole line  3 //compress unit is the whole frame  4 //number |
| PreWidth PreHeight PrePixel | 上一次设置的output port 宽高和像素 |
| PrepareOutBufCnt | 从sys拿到的outputbuffer 数量 |
| FinishOutBufCnt | divp处理的buffer数量 |

## echo

|  |  |
| --- | --- |
| 功能 | 打印接收到每一帧的PTS，如果前端是VDEC,会打印VDEC送过来的FrameID，Debug丢帧问题,正常情况下FrameID和PTS的值都是递增的。 |
| 命令 | echo checkframeid [ChnID] [Status] > /proc/mi\_modules/mi\_divp/mi\_divp0 |
| 参数说明 | [ChnID] 通道号 [0~32] |
|  | [Status] ON 开始打印 OFF 停止打印 |
| 举例 | echo checkframeid 0 ON/OFF > /proc/mi\_modules/mi\_divp/mi\_divp0  [MI DIVP PROCFS]:ChnID 0, receive buffer ID = 63 , H = 0, L = 196615, PTS = 43956000  [MI DIVP PROCFS]:ChnID 0, receive buffer ID = 64 , H = 0, L = 196616, PTS = 43989000  [MI DIVP PROCFS]:ChnID 0, receive buffer ID = 65 , H = 0, L = 196615, PTS = 44022000  [MI DIVP PROCFS]:ChnID 0, receive buffer ID = 66 , H = 0, L = 196615, PTS = 44055000  [MI DIVP PROCFS]:ChnID 0, receive buffer ID = 67 , H = 0, L = 196615, PTS = 44088000  [MI DIVP PROCFS]:ChnID 0, receive buffer ID = 68 , H = 0, L = 196615, PTS = 44121000  [MI DIVP PROCFS]:ChnID 0, receive buffer ID = 69 , H = 0, L = 196614, PTS = 44154000  [MI DIVP PROCFS]:ChnID 0, receive buffer ID = 70 , H = 0, L = 196615, PTS = 44187000  receive buffer ID：每个channel接收到buffer的ID。  H,L:VDEC 送来的FrameID,H高32位，L低32位，前端不是VDEC则无该打印。  PTS:FrameBuffer的PTS。 |

|  |  |
| --- | --- |
| 功能 | 统计1秒时间内GetBuffer的数量，以及每拿到两个inputbuffer之间的间隔时间的平均值，最大值，最小值，Debug丢帧问题。 |
| 命令 | echo statintervaltime [ChnID] ON/OFF > /proc/mi\_modules/mi\_divp/mi\_divp0 |
| 参数说明 | [ChnID] 通道号 [0~32] |
| [Status] ON 开始统计并打印 OFF 停止统计和打印 |
| 举例 | echo statintervaltime 0 ON > /proc/mi\_modules/mi\_divp/mi\_divp0  [MI DIVP PROCFS]:ChnlID 0,1s FrameCnt:19,Average time: 50393 us,MAX 59601 us,57652 us,57534 us,MIN 41401 us,42415 us,42896 us  [MI DIVP PROCFS]:ChnlID 0,1s FrameCnt:19,Average time: 50247 us,MAX 63004 us,62889 us,62845 us,MIN 36896 us,37120 us,37130 us  [MI DIVP PROCFS]:ChnlID 0,1s FrameCnt:18,Average time: 51049 us,MAX 70431 us,59763 us,59736 us,MIN 38873 us,40109 us,40283 us  [MI DIVP PROCFS]:ChnlID 0,1s FrameCnt:20,Average time: 48537 us,MAX 67168 us,67117 us,66953 us,MIN 18513 us,31242 us,32681 us  [MI DIVP PROCFS]:ChnlID 0,1s FrameCnt:19,Average time: 49962 us,MAX 61416 us,61416 us,61338 us,MIN 38564 us,38736 us,38762 us  FrameCnt: 1秒时间GetBuffer的数量  Average time： 拿到两个buffer之间间隔时间的平均时间  MAX：间隔时间最大的三个值。  MIN: 间隔时间最小的三个值 |

|  |  |
| --- | --- |
| 功能 | DIVP不再处理某一个channel，以节省频宽，验证频宽相关问题 |
| 命令 | echo stoponechannel [ChnID] [Status] > /proc/mi\_modules/mi\_divp/mi\_divp0 |
| 参数说明 | [ChnID] 通道号 [0~32] |
| [Status] ON 停止处理设置的channel OFF继续处理设置的channel |
| 举例 | echo stoponechannel 0 ON > /proc/mi\_modules/mi\_divp/mi\_divp0  通道0的画面将会停住 |

|  |  |
| --- | --- |
| 功能 | DIVP不处理所有的channel，DIVP硬件的属性为设置的Channel的属性。可以通过Debug Register来调试指定的channel。 |
| 命令 | echo processonechannel [ChnID] [Status] > /proc/mi\_modules/mi\_divp/mi\_divp0 |
| 参数说明 | [ChnID] 通道号 [0~32] |
| [Status] ON 停止所有的通道，硬件属性为所设置的通道的属性（即此通道会再处理一次）  OFF 所有通道正常运行 |
| 举例 | echo processonechannel 0 ON > /proc/mi\_modules/mi\_divp/mi\_divp0  所有通道全部停止即画面卡住，但通道0会多处理一帧  echo processonechannel 0 OFF > /proc/mi\_modules/mi\_divp/mi\_divp0  所有通道回复正常工作 |

# VDISP

## cat

【调试信息】

# cat proc/mi\_modules/mi\_vdisp/mi\_vdisp0

============================Private Vdisp0 Info ===============================

DevStatus

Start

------------------------------------------------------- Input Port Info ---------------------------------------------

PortID PortStatus ChnX ChnY ChnW ChnH IsFreeRun TryOk RecvOk

0 Enabled 0 0 960 540 1 245356299 313332

1 Enabled 960 0 960 540 1 355670297 313332

2 Enabled 0 540 960 540 1 578760014 313331

3 Enabled 960 540 960 540 1 757579130 313330

------------------------------------------------------ Output Port Info -------------------------------------------------

Inited FrmInterval BgColor PixelFmt FrmRate Width Height SendOk

1 33333 8388736 0 30 1920 1080 471418

【调试信息分析】

记录当前VDISP的使用状况以及device属性、Input port属性、Output port属性，可以动态地获取到这些信息，方便调试和测试。

【参数说明】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| device info | DevStatus | Vdisp设备的工作状态  Uninit：未初始化  Init：初始化  Start：运行状态  Stop：停止状态 |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| layer info | PortID | Inport ID  取值范围：[0~16]，16为overlay通道，即PIP通道。 |
| PortStatus | Port口状态  Uninit：inport未初始化  Init：inport已初始化  Enabled：inport已经使能  Disabled：inport已经禁用 |
| ChnX | 输入inport的起始坐标X地址  取值范围：[0~4094]，需要根据chip的align对齐 |
| ChnY | 输入inport的起始坐标Y地址  取值范围：[0~2159] |
| ChnW | 输入inport的图像宽度，需要根据chip的align对齐  取值范围：[0~4096] |
| ChnH | 输入inport的图像高度  取值范围：[0~4096] |
| IsFreeRun | 是否不做帧率控制  0：按pts控制播放  1：自由播放 |
| TryOk | 尝试取前级绑定端口的数据次数 |
| RecvOk | 从前级绑定端口成功拿到数据的次数 |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Output  port info | Inited | 是否有初始化Output Port  0：未初始化  1：已初始化 |
| FrmInterval | 出帧时间间隔，单位US |
| BgColor | 背景色，此处打印的是10进制 |
| PixelFmt | 色彩空间  取值范围[0~E\_MI\_SYS\_PIXEL\_FRAME\_FORMAT\_MAX-1]  0-YUYV422，9-YUVSP420 |
| FrmRate | 输出帧率 |
| Width | 图像输出宽度  取值范围：[0~4096]，需要根据chip的align对齐 |
| Height | 图像输出高度  取值范围：[0~2160]，需要根据chip的align对齐 |
| SendOk | 成功推往后级的帧数统计 |

# DISP

## cat

【调试信息】

# cat /proc/mi\_modules/mi\_disp/mi\_disp0

------------------------- start dump dev 0 Info -------------------------

DevStatus IrqNum IrqStatus BgColor CvbsStatus

1 211 1 0 0

Interface DevTiming CscMatrix Luma Contrast

5 16 0 50 50

Hue Saturation Gain Sharpness ActiveLayerId

50 50 0 0 0

------------------------- End dump dev 0 Info -------------------------

------------------------- start dump dev 0 Chn 0 Info -------------------------

LayerId LayerStatus BindedDevID ePixFormat Compress

0 1 0 0 0

LayerWidth LayerHeight LayDispWidth LayDispHeight Toleration

1920 1080 1920 1080 100

PendingBufCnt

0

------------------------- End dump dev 0 Chn 0 Info -------------------------

------------------------- start dump dev 0 Chn 0 InputPort 0 Info -------------------------

ParentLayerId PortStatus DispWin\_x DispWin\_y DispWinWidth

0 0 0 0 1920

DispWinHeight ZoomWin\_x ZoomWin\_y ZoomWinWidth ZoomWinHeight

1080 0 0 0 0

SyncMode FlipByGe PtsResetCnt FramePtsBefore FiredDiff

0 0 0 1 25617320

LastFramePts RecvCurPts NextPts

35232981 35232981 35183296

------------------------- End dump dev 0 Chn 0 InputPort 0 Info -------------------------

【调试信息分析】

记录当前DISP的使用状况以及device属性、layer属性、inputport属性，可以动态地获取到这些信息，方便调试和测试。

【参数说明】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| device info | DevStatus | 使能或者禁用  0：禁用  1：使能 |
| IrqNum | 中断号 |
| IrqStatus | 是否开启中断  0：中断禁用  1：中断使能 |
| BgColor | 背景色（RGB format） |
| CvbsStatus | 是否开启cvbs  0：关闭cvbs  1：开启cvbs |
| Interface | 接口类型  取值范围：（CVBS、YPBPR、VGA、BT656、BT1120、HDMI、LCD、BT656\_H、BT656\_L） |
| DevTiming | 输出时序  取值范围：[0~ E\_MI\_DISP\_OUTPUT\_USER] |
| CscMatrix | CSC矩阵选择  取值范围：[0~ E\_MI\_DISP\_CSC\_MATRIX\_RGB\_TO\_BT709\_PC] |
| Luma | 亮度  取值范围：[0 ~ 100] 默认50 |
| Contrast | 对比度  取值范围：[0 ~ 100] 默认50 |
| Hue | 色调  取值范围：[0 ~ 100] 默认50 |
| Saturation | 饱和度  取值范围：[0 ~ 100] 默认40 |
| Active Layers | 关联了那些Layer  取值范围：[0~1] |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| layer info | LayerId | layer ID  取值范围：[0~1] |
| LayerStatus | layer状态  0：关闭layer  1：开启layer |
| BindedDevID | 绑定的device ID  取值范围：[0~1] |
| ePixFormat | 像素格式  取值范围：[0~E\_MI\_SYS\_PIXEL\_FRAME\_YC420\_MSTTILE3\_H265] |
| Compress | 数据压缩  取值范围：[E\_MI\_SYS\_COMPRESS\_MODE\_NONE~E\_MI\_SYS\_COMPRESS\_MODE\_BUTT] |
| LayerWidth | Layer的宽 |
| LayerHeight | Layer的高 |
| LayDispWidth | Layer显示的宽 |
| LayDispHeight | Layer显示的高 |
| Toleration | PTS误差允许阈值，单位毫秒 |
| PendingBufCnt | 当前还有多少待显示的Frame |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Inputport info | ParentLayerId | 关联的layer ID  取值范围：[0~1] |
| PortStatus | port状态  取值范围：[E\_MI\_LAYER\_INPUTPORT\_STATUS\_INVALID~ E\_MI\_LAYER\_INPUTPORT\_STATUS\_HIDE] |
| DispWin\_x | 该port在layer上的起始横坐标 |
| DispWin\_y | 该port在layer上的起始纵坐标 |
| DispWinWidth | 该port输入的宽度 |
| DispWinHeight | 该port输入的高度 |
| ZoomWin\_x | 局部放大区域起始横坐标 |
| ZoomWin\_y | 局部放大区域起始纵坐标 |
| ZoomWinWidth | 局部放到区域宽度 |
| ZoomWinHeight | 局部放大区域高度 |
| Sync Mode | Check PTS/Free Run  0：无效  1：Check PTS  2：Free Run |
| FlipByGe | 是否使用GE  0：不使用GE  1：使用GE |
| PTS reset Cnt | reset pts的统计记录 |
| FramePtsBefore | Frame pts是否超前于系统时间用于计算nextpts  0：滞后  1：超前 |
| FiredDiff | 根据FramePtsBefore计算的系统时间和FramePts的差值（绝对值） |
| LastFramePts | 最近一次Flip的FramePts |
| RecvCurPts | 当前接收到buff的FramePts |
| NextPts | 期望的下一帧FramePts |

## echo

|  |  |
| --- | --- |
| 功能 | 设置开关中断 |
| 命令 | echo setIrqStatus [Status] > /proc/mi\_modules/mi\_disp/mi\_disp[DevID] |
| 参数说明 | [Status] 0 开启 1关闭 |
| [DevID] 设备号 [0~1] |
| 举例 | echo setIrqStatus 0 > /proc/mi\_modules/mi\_disp/mi\_disp0  disp设备0 关闭 中断 |

|  |  |
| --- | --- |
| 功能 | 设置pst检测 |
| 命令 | echo setsyncmode [LayID] [PortID] [SycMode] > /proc/mi\_modules/mi\_disp/mi\_disp[DevID] |
| 参数说明 | [LayID] Layer [0~1] |
| [PortID] Input Port 号 [0~15] |
| [SycMode] 同步模式 [0~2]  0 pts无效  1 检查pts  2 不检查pts |
| [DevID] 设备号 [0~1] |
| 举例 | echo setsyncmode 0 0 2 > /proc/mi\_modules/mi\_disp/mi\_disp0  disp 设备 0 输入端口0 不检查pts |

|  |  |
| --- | --- |
| 功能 | 设置Toleration |
| 命令 | echo settoleration [LayID] [PortID] [Value] > /proc/mi\_modules/mi\_disp/mi\_disp[DevID] |
| 参数说明 | [LayID] Layer 号 [0~1] |
| [PortID] Input Port 号 [0~15] |
| [Value] Toleration的数值，单位为ms |
| [DevID] 设备号 [0~1] |
| 举例 | echo settoleration 0 0 100 > /proc/mi\_modules/mi\_disp/mi\_disp0  disp 设备0 Layer 0 输入端口0 Toleration的值设为100ms |

|  |  |
| --- | --- |
| 功能 | 获取帧数据 |
| 命令 | echo getcapframe [DevID] [LayID] [File] > /proc/mi\_modules/mi\_disp/mi\_disp[DevID] |
| 参数说明 | [DevID] 设备号 [0~1] |
| [LayID] Layer [0~1] |
| [File] 路径+文件名， 产生的文件为YUYV |
| 举例 | echo getcapframe 0 0 /mnt/capture.yuv > /proc/mi\_modules/mi\_disp/mi\_disp0  disp 设备0 获取 设备0 Layer 0 在/mnt下产生一帧YUYV的图像，文件名为capture.yuv |

|  |  |
| --- | --- |
| 功能 | 设置Layer 的属性 |
| 命令 | echo setlayer [LayID][Width][Height][WinX][WinY][WinWidth][WinHeight] > /proc/mi\_modules/mi\_disp/mi\_disp[DevID] |
| 参数说明 | [LayID] Layer [0~1] |
| [Width] Layer 的宽 |
| [Height] Layer 的高 |
| [WinX] Windows 的x坐标 |
| [WinY] Windows 的y坐标 |
| [WinWidth] Windows 的宽 |
| [WinHeight] Windows 的高 |
| [DevID] 设备号 [0~1] |
| 举例 | echo setlayer 0 1920 1080 0 0 1920 1080 > /proc/mi\_modules/mi\_disp/mi\_disp0  disp 设备0 设置 Lyaer 0 宽为1920 高为1080 起始x坐标0 起始y坐标0 Windows的宽为1920 Windows的高为1080 |

|  |  |
| --- | --- |
| 功能 | 设置Layer 的状态 |
| 命令 | echo setlayerstatus [LyaID] [Status] > /proc/mi\_modules/mi\_disp/mi\_disp[DevID] |
| 参数说明 | [LayID] Layer [0~1] |
| [Status] 0 Disable 1 Enable |
| [DevID] 设备号 [0~1] |
| 举例 | echo setlayerstatus 0 0 > /proc/mi\_modules/mi\_disp/mi\_disp0  disp 设备0 Layer 0 Disable |

|  |  |
| --- | --- |
| 功能 | 设置PQ |
| 命令 | echo setcsc [DevID][Matrix] [Contrast] [ Hue] [Luma] [Saturation] > /proc/mi\_modules/mi\_disp/mi\_disp[DevID] |
| 参数说明 | [DevID] 设备号 [0~1] |
| [Matrix] 彩色空间 [0~6] 默认为0 |
| [Contrast] 对比度 [0~100] 默认为50 |
| [ Hue] 色度 [0~100] 默认为 50 |
| [Luma] 亮 度 [0~100] 默认为50 |
| [Saturation] 饱和度 [0~100] 默认为50 |
| 举例 | echo setcsc 0 0 50 50 50 50 > /proc/mi\_modules/mi\_disp/mi\_disp0  disp 设备0 颜色空间为0， 对比度为50， 色度50， 亮度50， 饱和度 50 |

|  |  |
| --- | --- |
| 功能 | 设置背景色 |
| 命令 | echo setbgcolor [DevID] [Value] > /proc/mi\_modules/mi\_disp/mi\_disp[DevID] |
| 参数说明 | [DevID] 设备号 [0~1] |
| [Value] 数值 |
| 举例 | echo setbgcolor 0 100 > /proc/mi\_modules/mi\_disp/mi\_disp0  disp 设备0 背景色数值为100 |

|  |  |
| --- | --- |
| 功能 | 是否打印FramePts |
| 命令 | echo checkframepts [LayID] [Status] > /proc/mi\_modules/mi\_disp/mi\_disp[DevID] |
| 参数说明 | [LayID] Layer [0~1] |
| [Status] ON 开启 OFF 关闭 |
| 举例 | echo checkframepts 0 ON > /proc/mi\_modules/mi\_disp/mi\_disp0  disp 设备0 开启Layer 0 的FramePts 抓log  [MI DISP PROCFS]:ChnID 0, receive buffer ID = 121, PTS = 123365433  ChnID:通道号  receive buffer ID: buff ID，数值递增  PTS:当前pst数值 |

|  |  |
| --- | --- |
| 功能 | 开启统计Get InputBuff时间间隔，该功能开启后，没隔1秒打印在该时间段内GetInputBuff的3个最大时间间隔、3个最小时间间隔和平均时间间隔 |
| 命令 | echo statintervaltime [LayID] [Status] > /proc/mi\_modules/mi\_disp/mi\_disp[DevID] |
| 参数说明 | [LayID] Layer [0~1] |
| [Status] ON 开启 OFF关闭 |
| [DevID] 设备号 [0~1] |
| 举例 | echo statintervaltime 0 ON > /proc/mi\_modules/mi\_disp/mi\_disp0  disp 设备0 开启时间间隔  [MI DISP PROCFS]:ChnlID 0,1s FrameCnt:29,Average time: 33100 us,MAX 40161 us,40123 us,40105 us,MIN 29788 us,29799 us,29838 us  ChnlID:通道号  1s:1秒钟  Average time:平均时间  MAX: 最大时间，获取了3个  MIN:最小时间， 获取了3个 |

# HDMI

## cat

【调试信息】

# cat /proc/mi\_modules/mi\_hdmi/mi\_hdmi0

--------------------------------------- HDMI0 Dev Info -------------------------------------

InitFlag AvMute PowerOn

Y N Y

EnableVideo TimingType OutputMode ColorType DeepColorMode

1 9 0 2 4

EnableAudio IsMultiChannel BitDepth CodeType SampleRate

1 0 16 0 3

【调试信息分析】

记录当前HDMI的使用状况以及device属性可以动态地获取到这些信息，方便调试和测试。

【参数说明】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| device info | InitFlag | HDMI模块是否初始化  Y：已初始化  N：未初始化 |
| AvMute | 音视频是否处于MUTE状态  Y：被MUTE  N：未被MUTE |
| PowerOn | HDMI是否使能电源管理  Y：是  N：否 |
| EnableVideo | 是否使能视频  0：未使能  1：使能 |
| TimingType | 当前HDMI的输出分辨率  取值范围[0~ E\_MI\_HDMI\_TIMING\_MAX] |
| OutputMode | HDMI输出模式  0：HDMI模式  2：DVI模式  1/3为HDCP模式，当前暂不支持 |
| ColorType | 颜色空间  0：RGB44  1：YUV422  2：YUV444  3：YUV420 |
| DeepColorMode | 色彩位深  0：8Bit  1：10Bit  2：12Bit  3：16Bit |
| EnableAudio | 是否使能音频  0：未使能  1：使能 |
| IsMultiChannel | 是否多通道  0：否  1：是 |
| BitDepth | 音频采样位深  8：8Bit  16：16Bit  32：32Bit |
| CodeType | 音频输出的压缩格式  0：PCM  1：非PCM |
| SampleRate | 音频输出采样率  0：未知  1：32K  2：44K  3：48K  4：88K  5：96K  6：176K  7：192K |

# FB

## cat

【调试信息】

# cat /proc/mi\_modules/mi\_fb0

############################OnDumpOSDAttr####################################

Framebuffer id = MStar FB0

xres=1280, yres=720

xres\_virtual=1280, yres\_virtual=1440

xoffset=0,yoffset=0

fix.line\_length=0xa00 Bytes

fix.smem\_start=0xf800000

Memory Size=0x400000 Bytes

Gop ID=0

Gwin ID=0

Open Count=1

Visible State=0

MIU Sel=0

ColorSpace=YUV

ColorFomrmat=ARGB1555

StretchWindow Pos[0,0]

StretchWindow Src[1280,720],StretchWindow Dst[1920,1080]

Gwin Pos[0,0]

Gwin Size[1280,720]

Gwin PhyAddr=0x0

ColorKey Enable=0

ColorKey Val=[0,0,0]

Enable Alpha Blend=1

Enalbe Multi Alpha=0

Global Alpha Val=0x0

Alpha0=0x0,Alpha1=0x0,(Only for ARGB1555)

GOP Hstart=192

Current TimingWidth=1920,TimingWidth=1080,hstar=192

##################################end##########################################

#########################OnDumpHwcursorAttr####################################

Cursor Gop ID=4

Cursor Gwin ID=6

Cursor MIU Sel=0

Cursor PhyAddr=0x166000

Cursor ColorFmt=ARGB8888

Cursor Icon Width=44,Height=56

Cursor HotSpot[18,9]

Cursor request pos[712,207]

Cursor Visible=1

Cursor Gwin Pos[694,198]

Cursor Gwin Size[44,56]

Cursor Gwin Pitch=0x200 Bytes

Curosr StretchWindow pos[0,0]

Cursor StretchWin Src[1280,720],Dst[1920,1080]

Cursor ColorKey Enable=0

Cursor ColorKey Value=[0xff,0xff,0xff]

############################OnDumpHwcursorAttr#################################

【调试信息分析】

记录当前Fbdev && Hwcursor的使用状况以及OSD图层属性、OSD device属性、Hwcursor属性，可以动态地获取到这些信息，方便调试和测试。

【参数说明】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| OSD info | Framebuffer id | Framebuffer设备ID |
| xres | 可见分辨率宽度 |
| yres | 可见分辨率宽高度 |
| xres\_virtual | 虚拟分辨率的宽度 |
| yres\_virtual | 虚拟分辨率的高度 |
| xoffset | 可见分辨率到虚拟分辨率的x偏移量 |
| yoffset | 可见分辨率到虚拟分辨率的y偏移量 |
| fix.line\_length | 虚拟分辨率每行的字节数 |
| fix.smem\_start | 帧缓冲内起始地址 |
| Memory Size | 帧缓冲内存长度，单位字节数 |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| OSD device info | Gop ID | 图层硬件ID |
| Gwin ID | 图层硬件显示窗口ID |
| Open Count | 该图形层打开次数。在用户调用open()时增1；调用close()时减1 |
| Visible State | 该图层显示状态。  取值:{1表示可见，0表示不可见}  用户显示调用FBIOSET\_SHOW去设置图层可见状态。 |
| MIU Sel | 保存帧缓冲数据的物理内存编号。  对于MSR620设备该值为0 |
| ColorSpace | 该图形层输出的颜色空间。  取值:{YUV表示输出YUV color space;RGB表示输出RGB color space }。默认为YUV |
| ColorFomrmat | 图形层格式  取值:{ARGB1555,ARGB4444,ARGB8888,RGB565}。  用户设置可变屏幕信息中的格式项后更新 |
| StretchWindow Pos | 图形层在显示设备上的起始位置。  单位:像素  默认为0,用户可调用FBIOSET\_SCREEN\_LOCATION去设置 |
| StretchWindow Src | 图形层原始分辨率的宽度和高度。  单位:像素  通过硬件放大功能放大到当前timing。用户设置可变屏幕信息中的可见分辨率后更新 |
| StretchWindow Dst | 图形层在显示设备上的宽度和高度。  默认设置为输出timing大小。可以通过FBIOSET\_SCREEN\_LOCATION去设置 |
| Gwin Pos | 图形层显示窗口的起始位置  单位:像素  对于OSD图层来说,该值总是为(0,0) |
| Gwin Size | 图形层显示窗口的宽度、高度。  单位:像素  对于OSD图层来说，该值总是被设置为可见分辨率的宽度、高度。用户设置可变屏幕信息中的可见分辨率后更新。 |
| Gwin PhyAddr | 图形层当前显示数据的物理地址 |
| ColorKey Enable | 图形层ColorKey是否使能  取值:{0表示Disable ColorKey; 1表示Enable ColorKey}用户设置可通过FBIOSET\_COLORKEY去设置 |
| ColorKey Val | 图形层ColorKey的RED,GREEN,BLUE通道数值  取值:每个通道的取值范围从0-255。ColorKey val的数值用RGB888格式表示  用户可通过FBIOSET\_COLORKEY去设置后更新 |
| Enable Alpha Blend | 图形层alpha是否使能。  取值:{0表示否，1表示是},默认为1。  用户可通过FBIOSET\_GLOBAL\_ALPHA去设置后更新。  该项关闭，则像素alpha配置不生效。该项开启且Enable Multi Alpha关闭,仅像素alpha生效(对于ARGB1555格式,Alpha bit为1的像素使用Global Alpha作为像素alpha,Alpha bit为0的像素使用0xff作为像素alpha)。当该项目开启且Enable Multi Alpha生效时,则像素alpha和Global Alpha都生效 |
| Enable Multi Alpha | 全局alpha是否生效开关  取值{0时，1否}默认值为0  Enable Multi Alpha使能，Global Alpha便生效 |
| Global Alpha Val | 全局Alpha |
| Alpha0 | ARGB1555格式时，当最高位为0时,选择Alpha0作为Alpha混合的的Alpha数值  取值: 0~255默认为0 |
| Alpha1 | ARGB1555格式时，当最高位为1时，选择Alpha1作为Alpha混合的Alpha数值  取值: 0~255默认为0 |
| GOP Hstart | 图层硬件向对于显示窗口的偏移。  取值:该值和当前输出timing相关，当输出timing发生变化时，通过FBIOSET\_DISPLAYLAYER\_ATTRIBUTES并将输入参数MI\_FB\_DisplayLayerAttr\_s的u32SetAttrMask数值包含E\_MI\_FB\_DISPLAYLAYER\_ATTR\_MASK\_SCREEN\_SIZE mask位来通知图层驱动输出timing发生变化。图层驱动会根据当前timing设置相应的hstart数值 |
| Current TimingWidth, | 当前输出timing的宽度、高度以及当前timing下应该给图形层硬件设置的像对于显示窗口的偏移。  该项与GOP Hstart配合使用，用于检查设置给硬件图层的hstart数值是否和当前timing一致。 |
| Current TimingHeight, |
| hstar |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Hwcursor info | Cursor Gop ID | 鼠标图层硬件ID |
| Cursor Gwin ID | 鼠标图层硬件显示窗口ID |
| Cursor MIU Sel | 用于保存鼠标图标数据的物理内存编号。  对于MSR620设备该值为0 |
| Cursor PhyAddr | 鼠标层当前显示数据的物理地址 |
| Cursor ColorFmt | 鼠标图层格式  值:{ARGB1555,ARGB4444,ARGB8888,RGB565}。  用户可通过FBIOSET\_CURSOR\_ATTRIBUTE 设置 |
| Cursor Icon Width | 鼠标图标的宽度、高度。  单位：像素  用户可通过FBIOSET\_CURSOR\_ATTRIBUTE设置 |
| Cursor HotSpot | 鼠标图标的热点信息  单位:像素  热点信息通常和鼠标图标相关，用户可通过FBIOSET\_CURSOR\_ATTRIBUTE设置 |
| Cursor request pos | 鼠标图标的位置信息  单位: 像素  用户通过FBIOSET\_CURSOR\_ATTRIBUTE设置 |
| Cursor Visible | 鼠标图层是否可见  取值:{0,表示不可见，1表示可见}  用户通过FBIOSET\_CURSOR\_ATTRIBUTE设置 |
| Cursor Gwin Pos | 鼠标图层显示窗口的起始位置  单位:像素 |
| Cursor Gwin Pitch | 鼠标图层每行的字节数。  对于鼠标图层来说，该值固定为0x200 |
| Curosr StretchWindow pos | 鼠标图层在显示设备上的起始位置。  单位:像素 |
| Cursor StretchWin Src | 鼠标图层原始大小  单位:像素。该值通常和OSD的显示分辨率一致。 |
| Cursor StretchWin Dst | 鼠标图层在显示设备上的宽度、高度  该值和输出timing一致。 |
| Cursor ColorKey Enable | 鼠标图层ColorKey是否使能  取值:{0表示Disable ColorKey; 1表示Enable ColorKey}。  用户通过FBIOSET\_CURSOR\_ATTRIBUTE设置 |
| Cursor ColorKey Value | 鼠标图层ColorKey的RED,GREEN,BLUE通道数值  取值:每个通道的取值范围从0-255。ColorKey val的数值用RGB888格式表示  用户可通过FBIOSET\_CURSOR\_ATTRIBUTE去设置后更新 |

## echo

|  |  |
| --- | --- |
| 功能 | 打开或关闭指定的图层 |
| 命令 | echo GUI\_SHOW [LayerID] [Status] > /proc/mi\_modules/mi\_fb0 |
| 参数说明 | [LayerID] 图层号 |
| [Status] on 打开 off 关闭 |
| 举例 | echo GUI\_SHOW 0 on > /proc/mi\_modules/mi\_fb0  显示图层fb0  echo GUI\_SHOW 0 off > /proc/mi\_modules/mi\_fb0  显示图层fb0 |

|  |  |
| --- | --- |
| 功能 | 打开或关闭硬件鼠标 |
| 命令 | echo CURSOR\_SHOW [Status] > /proc/mi\_modules/mi\_fb0 |
| 参数说明 | [Status] on 打开 off 关闭 |
| 举例 | echo CURSOR\_SHOW on > /proc/mi\_modules/mi\_fb0  显示鼠标图标  echo CURSOR\_SHOW off > /proc/mi\_modules/mi\_fb0  关闭鼠标图标 |

|  |  |
| --- | --- |
| 功能 | 为指定的图层设置Colorkey |
| 命令 | echo GUI\_SET\_CLRKEY [LayerID] [Red] [Green] [Blue] > /proc/mi\_modules/mi\_fb0 |
| 参数说明 | [LayerID] 图层号 |
| [Red] 红色分量 以16进制,ARGB888 格式表示 |
| [Green] 绿色分量 以16进制,ARGB888 格式表示 |
| [Blue] 蓝色分量 以16进制,ARGB888 格式表示 |
| 举例 | echo GUI\_SET\_CLRKEY 0 ff 00 00 > /proc/mi\_modules/mi\_fb0  设置fb0的Colorkey为红色 |

|  |  |
| --- | --- |
| 功能 | Dump当前正在显示的鼠标图标 |
| 命令 | echo CURSOR\_DUMP [Path] >/proc/mi\_modules/mi\_fb0 |
| 参数说明 | [Path] dump 的路径 |
| 举例 | echo CURSOR\_DUMP /mnt > /proc/mi\_modules/mi\_fb0  dump cursor Icon到/mnt路径，文件名为CursorData.raw，用7yuv打开，宽高设置为128,128，格式按照cursor icon的格式设置即可显示Rawdata |

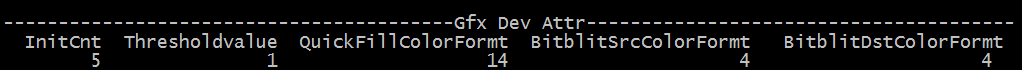
|  |  |
| --- | --- |
| 功能 | 分别为ARGB1555格式的alpha bit0,alpha bi1设置pixel alpha数值。因为目前msr620不支持这个feature，暂时未实现 |
| 命令 | echo GUI\_SETALPHA\_ARGB1555 [LayerID] [Alpha0] [Alpha1]> /proc/mi\_modules/mi\_fb0 |
| 参数说明 | [LayerID] 图层号 |
| [Alpha0] alpha0 的数值 |
| [Alpha1] alpha1 的数值 |
| 举例 | 无 |

# GFX

## cat

【调试信息】

# cat /proc/mi\_modules/mi\_gfx/mi\_gfx0



【调试信息分析】

记录当前GFX的使用状况以及device属性可以动态地获取到这些信息，方便调试和测试。

【参数说明】

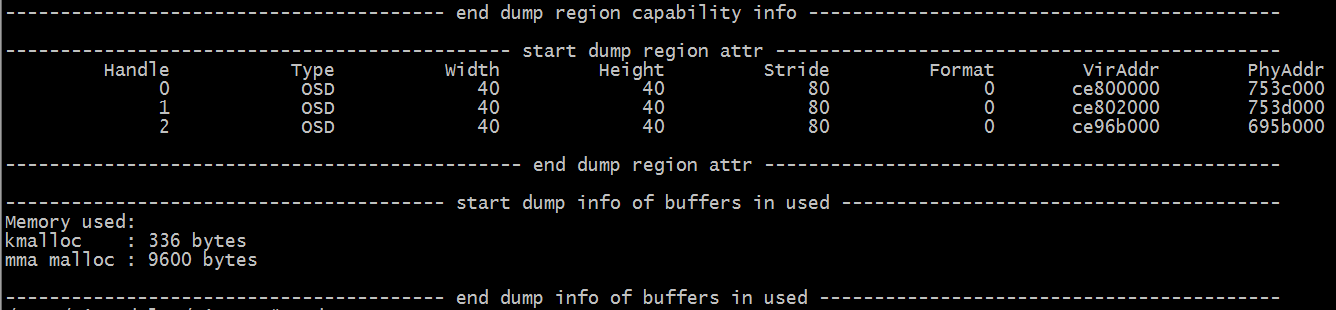
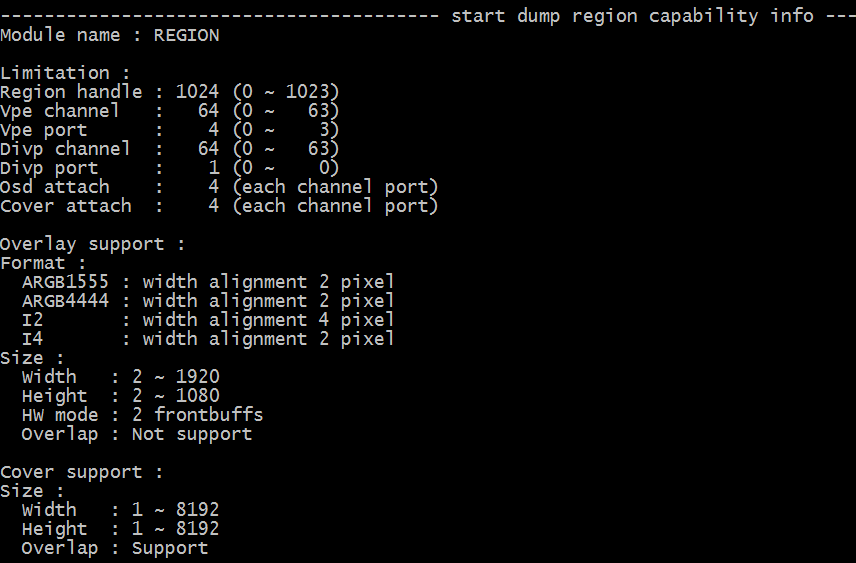
|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| device info | InitCnt | 设备被Open的次数，由于GFX会有多个模块进行调用，并且不能确定是哪一个module会先调用GFX的初始化，所以设备允许多次Open，会记录Open次数来在Close的时候做对应的计数判断是否需要真的Close Device。 |
| Thresholdvalue | ARGB1555时，GFX内部的alpha判定阈值，如大于N则alpha为1，反之alpha为0.  在ARGB8888模式下无效。默认值为1. |
| QuickFillColorFormt | 最后一次做QuickFill的颜色格式  取值参考：GFX\_Buffer\_Format |
| BitblitSrcColorFormt | 最后一次做BitBlit的Src Surface的颜色格式 |
| BitblitDstColorFormt | 最后一次做BitBlit的Dst Surface的颜色格式 |

# REGION

## cat

【调试信息】

# cat /proc/mi\_modules/mi\_rgn/mi\_rgn0



【调试信息分析】

记录当前RGN的使用状况以及device属性可以动态地获取到这些信息，方便调试和测试

【参数说明】

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Region capability | Region handle | 创建region的最大数量为1024个，句柄取值为[0~1023] |
| Vpe channel | Vpe最大通道数量为64个，通道取值为[0~63] |
| Vpe port | Vpe最大输出端口数量为4个，端口取值为[0~3] |
| Divp channel | Divp最大通道数量为64个，通道取值为[0~63] |
| Divp port | Divp最大输出端口数量为1个，端口取值为0 |
| Osd attach | 每个输出端口每个通道能绑定的Osd数量上限为4个 |
| Cover attach | 每个输出端口每个通道能绑定的Cover数量上限为4个 |
| Overlay Format | Osd支持的颜色格式：  ARGB1555，宽度按2个像素对齐  ARGB4444，宽度按2个像素对齐  I2，宽度按4个像素对齐  I4，宽度按2个像素对齐 |
| Overlay Size | Osd宽取值范围为2 ~ 1920，要求为2的整数倍  Osd高取值范围为2 ~ 1080，要求为2的整数倍 |
| Overlay HW mode | Osd的frontbuff数量，目前设置为2 |
| Cover Size | Cover宽取值为[1~8192]  Cover高取值为[1~8192] |
| Overlap | 是否支持范围重叠 |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Region attr | Handle | 句柄 |
| Type | 类型，Osd或是Cover |
| Width | 宽 |
| Height | 高 |
| Stride | 宽补齐 |
| Format | 颜色格式，ARGB1555，ARGB4444，I2，I4 |
| VirAddr | Canvas的虚拟地址 |
| PhyAddr | Canvas的物理地址 |

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | | | 描述 |
| Region attr | Frontbuffer info | Index | Frontbuffer索引 |
| bShow | 是否显示 |
| OffsetX | X偏移 |
| OffsetY | Y偏移 |
| Width | 宽 |
| Height | 高 |
| Stride | 宽补齐 |
| Format | 颜色格式 |
| VirAddr | 虚拟地址 |
| PhyAddr | 物理地址 |
| Attach info | Handle | 绑定Region句柄 |
| Type | 绑定Region类型 |
| bShow | 是否显示 |
| Layer | 绑定Cover的层级 |
| Color | 绑定Cover的颜色值 |
| Width | 宽 |
| Height | 高 |
| Stride | 宽补齐 |
| PositionX | X偏移 |
| PositionY | Y偏移 |
| Format | 绑定Osd的颜色格式 |
| VirAddr | 虚拟地址 |
| PhyAddr | 物理地址 |

|  |  |  |
| --- | --- | --- |
| 参数 | | 描述 |
| Buffer info | kmalloc | Region模块内部申请内存字节数。 |
| mma alloc | Region模块通过Sys模块申请内存字节数。 |

## echo

|  |  |
| --- | --- |
| 功能 | Dump指定Region的buffer |
| 命令 | echo dumpRgnBuf [Handle] [Path] > /proc/mi\_modules/mi\_rgn0 |
| 参数说明 | [Handle] region句柄 |
| [Path] 保存dump数据的路径。保存内容为region的canvas内容 |
| 举例 | echo dumpRgnBuf 0 /mnt > /proc/mi\_modules/mi\_rgn0  在/mnt 下产生文件 Rgn0\_canvasInfo\_fmt0\_64X48  表示dump句柄为0，颜色格式为ARGB1555，Stride为64，高为48的region的canvas数据  文件格式为Rgn[Handle]\_canvasInfo\_fmt[Format]\_[Stride]X[Height]  [Handle]: region句柄。  [Format]：颜色格式，ARGB1555为0，ARGB4444为1，I2为2，I4为3。  [Stride]：宽补齐  [Height]：高 |

|  |  |
| --- | --- |
| 功能 | Dump指定channel和port的frontbuffer |
| 命令 | echo dumpFrontBuf [ModId] [ChnID] [PortID] [Path] > /proc/mi\_modules/mi\_rgn0 |
| 参数说明 | [ModId] 端口类型，Vpe为0，Divp为1。 |
| [ChnID] 通道号 [0 ~ 63] |
| [PortID] 端口号 [0~3] |
| [Path] 保存dump数据的路径。会根据当前实际使用的frontbuffer数量生成0 ~ 2个文件 |
| 举例 | echo dumpFrontBuf 0 0 0 /mnt > /proc/mi\_modules/mi\_rgn0  在/mnt下产生 Vpe\_Chn0\_Port0\_frontbuf0\_fmt0\_64X48, 表示dump通道为0，输出端口为Vpe0号端口，索引为0的frontbuffer的数据，颜色格式为ARGB1555，Stride为64，高为48  文件格式  [Mod]\_Chn[Channel]\_Port[Port]\_frontbuf[Index]\_fmt[Format]\_[Stride]X[Height]  [Mod] 输出端口类型，Vpe时为Vpe，Divp时为Divp。  [Channel 通道号。  [Port] 输出端口。  [Index] frontbuffer索引号。  [Format] 颜色格式，ARGB1555为0，ARGB4444为1，I2为2，I4为3。  [Stride] 宽补齐  [Height] 高 |

|  |  |
| --- | --- |
| 功能 | 获取region能力级信息 |
| 命令 | echo getcap > /proc/mi\_modules/mi\_rgn0 |
| 参数说明 | 无 |
| 举例 | 无 |

|  |  |
| --- | --- |
| 功能 | 获取已创建region信息 |
| 命令 | echo dumprgn > /proc/mi\_modules/mi\_rgn0 |
| 参数说明 | 无 |
| 举例 | 无 |

|  |  |
| --- | --- |
| 功能 | 获取channel和port的信息 |
| 命令 | echo dumpchport > /proc/mi\_modules/mi\_rgn0 |
| 参数说明 | 无 |
| 举例 | 无 |

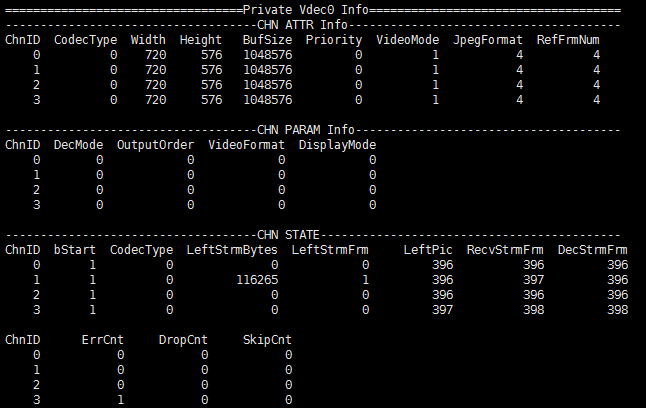
|  |  |
| --- | --- |
| 功能 | 获取内存使用信息 |
| 命令 | echo bufcnt > /proc/mi\_modules/mi\_rgn0 |
| 参数说明 | 无 |
| 举例 | 无 |

# VDEC

## cat

【调试信息】

# cat /proc/mi\_modules/mi\_vdec/mi\_vdec0



【调试信息分析】

打印分为两部分，使用Private Vdec0 Info分隔开。上半部分为common信息，下半部分为vdec模块信息。

主要记录了解码通道的使用情况及配置属性，可用于检查属性配置和当前通道的工作状态，便于debug。

【参数说明】

|  |  |  |
| --- | --- | --- |
| **参数** | | **描述** |
| CHN ATTR Info | ChnID | 通道号。 |
| CodecType | 解码协议类型  0：H264;  1：H265;  2：JPEG。 |
| Width | 解码图像最大宽度。 |
| Height | 解码图像最大高度。 |
| BufSize | VDEC码流buffer大小，单位：byte。 |
| Priority | 解码通道优先级。 |
| VideoMode | 发送码流方式。  0：按流发送；  1：按帧发送。 |
| JpegFormat | Jpeg图片的存储格式  0：YCbCr400；  1：YCbCr420；  2：YCbCr422；  3：YCbCr444。 |
| RefFrmNum | 最大参考帧个数，JPEG通道无效。 |
| CHN PARAM Info | ChnID | 通道号。 |
| DecMode | 解码模式。  0：解码IPB数据帧模式；  1：只解I帧模式；  2：只解IP帧，跳过B帧。 |
| OutputOrder | 解码图像输出顺序。  0：按显示顺序输出数据帧；  1：按解帧顺序输出数据帧。 |
| VideoFormat | 解码图像数据格式。  0：TILE数据格式；  1：数据帧压缩模式，减少数据帧内存使用量。 |
| DisplayMode | 显示模式。  0：预览模式，不参考PTS输出。  1：回放模式，参考PTS数值输出。 |
| CHN STATE | ChnID | 通道号。 |
| bStart | 解码器是否启动接收码流。 |
| CodecType | 解码协议类型  0：H264;  1：H265;  2：JPEG |
| LeftStrmBytes | 码流buffer中待解码的byte数。 |
| LeftStrmFrm | 码流buffer中待解码的帧数。–1 表示无效。  仅按帧发送时有效。 |
| LeftPics | 图像buffer中剩余的pic数目。 |
| RecvStrmFrm | 码流buffer中已接收码流帧数。-1 表示无效。  仅按帧发送时有效。 |
| DecStrmFrm | 码流buffer中已解码帧数。 |
| ErrCnt | 解码错误帧数 |
| DropCnt | 解码丢弃帧数 |
| SkipCnt | 解码跳过帧数 |

## echo

|  |  |
| --- | --- |
| 功能 | 设置解码模式 |
| 命令 | echo setDecMode [ChnID] [ModID] > /proc/mi\_modules/mi\_vdec/mi\_vdec0 |
| 参数说明 | [ChnID] 通道号 [0~32] |
| [ModID] 解码模式 0->H264  1->H265  2->JPEG |
| 举例 | echo setDecMode 0 0 > /proc/mi\_modules/mi\_vdec/mi\_vdec0  通道0，解码模式为H264 |

|  |  |
| --- | --- |
| 功能 | 设置解码图像输出顺序 |
| 命令 | echo setOutputOrder [ChnID] [ModID] > /proc/mi\_modules/mi\_vdec/mi\_vdec0 |
| 参数说明 | ChnID 通道号 [0~32] |
| ModID 0 按显示顺序输出数据帧  1 按解码顺序输出数据帧 |
| 举例 | echo setOutputOrder 0 0 > /proc/mi\_modules/mi\_vdec/mi\_vdec0  通道0，按显示顺序输出 |

|  |  |
| --- | --- |
| 功能 | 设置解码图像数据格式，暂不支持设置 |
| 命令 | echo setVideoFormat [ChnID] [ModID] > /proc/mi\_modules/mi\_vdec/mi\_vdec0 |
| 参数说明 | [ChnID] 通道号 [0~32] |
| [ModID] 解码数据格式 0 TILE格式  1数据帧压缩模式，减少数据帧内存使用量 |
| 举例 | echo setVideoFormat 0 0 > /proc/mi\_modules/mi\_vdec/mi\_vdec0  通道0，TILE格式输出 |

|  |  |
| --- | --- |
| 功能 | 设置显示模式 |
| 命令 | echo setDisplayMode [ChnID] [ModID] > /proc/mi\_modules/mi\_vdec/mi\_vdec0 |
| 参数说明 | [ChnID] 通道号 [0~32] |
| [ModID] 0：预览模式，不参考PTS输出。  1：回放模式，参考PTS数值输出。 |
| 举例 | echo setDisplayMode 0 0 > /proc/mi\_modules/mi\_vdec/mi\_vdec0  通道0 预览模式 |

|  |  |
| --- | --- |
| 功能 | 打开/关闭统计打印码流输入时间间隔，每隔一秒打印一次 |
| 命令 | echo statInputTimeIntvl [ChnID] [Status] >/proc/mi\_modules/mi\_vdec/mi\_vdec0 |
| 参数说明 | [ChnID] 通道号 [0~32] |
| [Status] on 打开 off 关闭 |
| 举例 | echo statInputTimeIntvl 0 on >/proc/mi\_modules/mi\_vdec/mi\_vdec0  [MI VDEC PROCFS]:Input ChnID: 0, CurTime: 5959100469, TotalFrmCnt: 19, AvgTimeIntvl: 49856, MaxTimeIntvl: 50047, MinTimeIntvl: 47252  [Input ChnID] 通道号  [CurTime] 当前时间  [TotalFrmCnt] 输入帧次数  [AvgTimeIntvl 每秒内所有帧输入平均时间间隔  [MaxTimeIntvl] 每秒内两帧输入最大时间间隔  [MinTimeIntvl] 每秒内两帧输入最小时间间隔。 |

|  |  |
| --- | --- |
| 功能 | 打开/关闭统计打印图像输出时间间隔 |
| 命令 | echo statOutputTimeIntvl[ChnID][Status]>/proc/mi\_modules/mi\_vdec/mi\_vdec0 |
| 参数说明 | [ChnID] 通道号 [0~32] |
| [Status] on 打开 off 关闭 |
| 举例 | echo statOutputTimeIntvl 0 on> /proc/mi\_modules/mi\_vdec/mi\_vdec0  MI VDEC PROCFS]:Output ChnID: 0, CurTime: 6413830139, TotalFrmCnt: 19, AvgTimeIntvl: 49999, MaxTimeIntvl: 50021, MinTimeIntvl: 49985  [Output ChnID] 通道号  [CurTime] 当前时间  [TotalFrmCnt] 输入帧次数  [AvgTimeIntvl 每秒内所有帧输入平均时间间隔  [MaxTimeIntvl] 每秒内两帧输入最大时间间隔  [MinTimeIntvl] 每秒内两帧输入最小时间间隔。 |

|  |  |
| --- | --- |
| 功能 | 打开/关闭统计打印输入码流时间戳 |
| 命令 | echo ChkInputFrmPts [ChnID] [Status] > /proc/mi\_modules/mi\_vdec/mi\_vdec0 |
| 参数说明 | [ChnID] 通道号 [0~32] |
| [Status] on 打开 off 关闭 |
| 举例 | echo ChkInputFrmPts 0 on > /proc/mi\_modules/mi\_vdec/mi\_vdec0  [MI VDEC PROCFS]:Input ChnID: 0, TotalFrmCnt: 400, PTS: 3964389  [Input ChnID] 通道号；  [TotalFrmCnt] 输入帧次数；  [PTS] 码流时间戳。 |

|  |  |
| --- | --- |
| 功能 | 打开/关闭统计打印输出码流时间戳 |
| 命令 | echo ChkOutputFrmPts [ChnID] [Status] > /proc/mi\_modules/mi\_vdec/mi\_vdec0 |
| 参数说明 | [ChnID] 通道号 [0~32] |
| [Status] on 打开 off 关闭 |
| 举例 | echo ChkOutputFrmPts 0 on > /proc/mi\_modules/mi\_vdec/mi\_vdec0  [MI VDEC PROCFS]:Output ChnID: 0, TotalFrmCnt: 154, PTS: 4039893000  [Output ChnID] 通道号；  [TotalFrmCnt] 输入帧次数；  [PTS] 码流时间戳。 |

|  |  |
| --- | --- |
| 功能 | 将写入码流buffer的码流同时存文件 |
| 命令 | echo DumpBS [Path/Status] [MoreChnID] > /proc/mi\_modules/mi\_vdec/mi\_vdec0 |
| 参数说明 | [Path/Status] 存储路径；或者输入 off，关闭dump操作 |
| [MoreChnID] 通道号 [0~32] 可以设置多路，如 0 1 2 3 |
| 举例 | echo DumpBS /mnt 0 > /proc/mi\_modules/mi\_vdec/mi\_vdec0  [MI WRN ]: \_MI\_VDEC\_IMPL\_InjectBuffer[737]: Chn(0) Dump Es Buffer, Size:5765, (0x00 0x00 0x00 0x01 0x61)  在/mnt下产生 chn\_0\_h264\_dump\_vdec.es 文件，为es流文件 |

|  |  |
| --- | --- |
| 功能 | 将解码出的图像同时存文件，存3帧 |
| 命令 | echo DumpFB [Path] [MoreChnID] > /proc/mi\_modules/mi\_vdec/mi\_vdec0 |
| 参数说明 | [Path] 存储路径 |
| [MoreChnID] 通道号 [0~32] 可以设置多路，如 0 1 2 3 |
| 举例 | echo DumpFB /mnt 0 > /proc/mi\_modules/mi\_vdec/mi\_vdec0  [MI WRN ]: \_MI\_VDEC\_IMPL\_DebugWriteFile[3494]: dump file(/mnt/chn\_0\_h264\_dump\_vdec[720\_576\_736]\_0.yuv) v1 ok ..............[len:635904]  在 /mnt下产生 chn\_0\_h264\_dump\_vdec[720\_576\_736]\_0.yuv（已detile），  chn\_0\_dump\_vdec[720\_576\_736]\_0\_luma.yuv，  chn\_0\_dump\_vdec[720\_576\_736]\_0\_chrome.yuv  各3帧，共9帧 |

|  |  |
| --- | --- |
| 功能 | 将当前码流buffer中的所有数据存储到文件中 |
| 命令 | echo DumpCurBS [Path] [ChnID] > /proc/mi\_modules/mi\_vdec/mi\_vdec0 |
| 参数说明 | [Path] 存储路径 |
| [ChnID] 通道号 [0~32] |
| 举例 | echo DumpCurBS /mnt 0 > /proc/mi\_modules/mi\_vdec/mi\_vdec0  Dump Chn(0) ES Data Now  在/mnt下 产生 chn\_0\_h264\_dump\_now\_vdec.es |

|  |  |
| --- | --- |
| 功能 | 设置utopia打印等级， 不建议使用 |
| 命令 | echo SetDbgLevel [Level] > /proc/mi\_modules/mi\_vdec/mi\_vdec0 |
| 参数说明 | [Level] 等级 [0~5]  0 关闭所有utopia的信息  1 只输出error信息  2 输出general 信息  3 输出debug信息  4 输出trace 信息  5 输出firmware 信息 |
| 举例 | echo SetDbgLevel 3 > /proc/mi\_modules/mi\_vdec/mi\_vdec0  utopia层输出debug信息 |

|  |  |
| --- | --- |
| 功能 | 统计前1秒内解码帧数，需配合 cat /proc/mi\_modules/mi\_vdec/mi\_vdec0 使用 |
| 命令 | echo StatDecCntLastSec [Static] > /proc/mi\_modules/mi\_vdec/mi\_vdec0 |
| 参数说明 | [Status] on 开启 off 关闭 |
| 举例 | echo StatDecCntLastSec on > /proc/mi\_modules/mi\_vdec/mi\_vdec0  cat /proc/mi\_modules/mi\_vdec/mi\_vdec0  ------------------------------------CHN STATE 2-----------------------------------------  ChnID DecCntLastSec  0 18  1 18    ChnID: 通道号  DecCntLastSec: 前一秒内解码帧数 |