



# CV180X & CV181X 双系统使用手册

Version:

Release date: 2024-03-05

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

# 目录

<b>1</b>	<b>声明</b>	<b>2</b>
<b>2</b>	<b>概述</b>	<b>3</b>
<b>3</b>	<b>双系统使用介绍</b>	<b>4</b>
3.1	双系统特点	4
3.2	双系统编译 Flow	4
3.3	双系统启动流程	5
3.4	软件架构	6
3.4.1	单/双系统软件架构	6
3.4.2	双核通信	7
3.5	单/双系统的差异	9
3.5.1	新增编译选项	9
3.5.2	镜像包差异	10
3.5.3	uboot & sd 卡升级	11
3.5.4	xml 分区	12
3.5.5	memmap	13
3.6	PINMUX 使用	14
3.6.1	单系统中的 PPINMUX 配置	14
3.6.2	双系统中的 PINMUX 配置	15
3.7	常见的外设使用	16
3.8	外设更换 demo	16
3.8.1	Flash	16
3.8.2	Sensor	16
3.8.2.1	非快启	17
3.8.2.2	快启	17
3.9	常用 Debug 方法	19
3.9.1	快启上电到出第一张收敛后的图像各阶段的耗时	19
3.9.2	常见问题	19
3.9.3	Linux Crash SOP	20
3.9.4	小核 Crash	20
3.9.5	小核 Memory 分析	21

修订记录

Revision	Date	Description
v1.0	2024/03/05	初稿

# 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 概述

---

此文件介绍双系统的特点及使用方法，旨在为客户提供快速上手文档，提高开发效率。

# 3 双系统使用介绍

---

## 3.1 双系统特点

- 降低大核 cpu loading: 将多媒体部分挪到小核, 利用小核处理能力进行多媒体处理, 减少大核 cpu loading
- 减少系统启动时间: 大核启动 linux 的同时, 小核已经在进行多媒体、isp 的初始化, 在 linux 启动后, 多媒体码流已经准备好, 减少系统整体启动时间

## 3.2 双系统编译 Flow

通用编译 flow 基本与单 linux 系统一致, 在拿到开发板后, 执行:

```
--> source build/cvsetup.sh
--> defconfig cv181x_XXXX_XXXX
--> build_all
```

即可编译出 evb 板所需要的镜像, 具体可参考 SDKCompilationandUsageGuide\_zh.pdf 2.4 编译

### 3.3 双系统启动流程

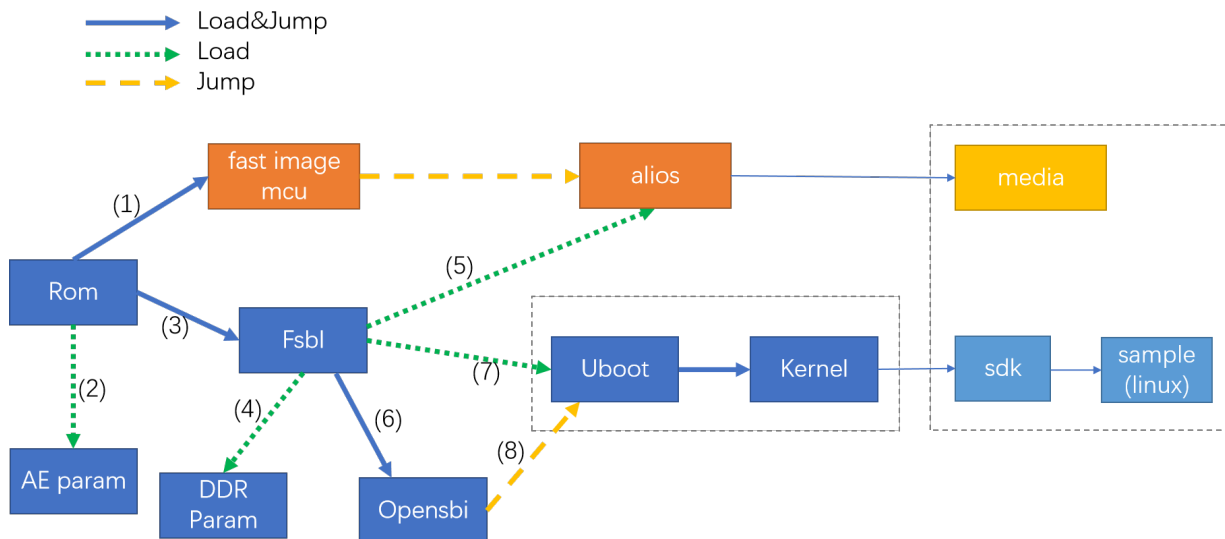


图 3.1: 双系统启动流程

## 3.4 软件架构

### 3.4.1 单/双系统软件架构

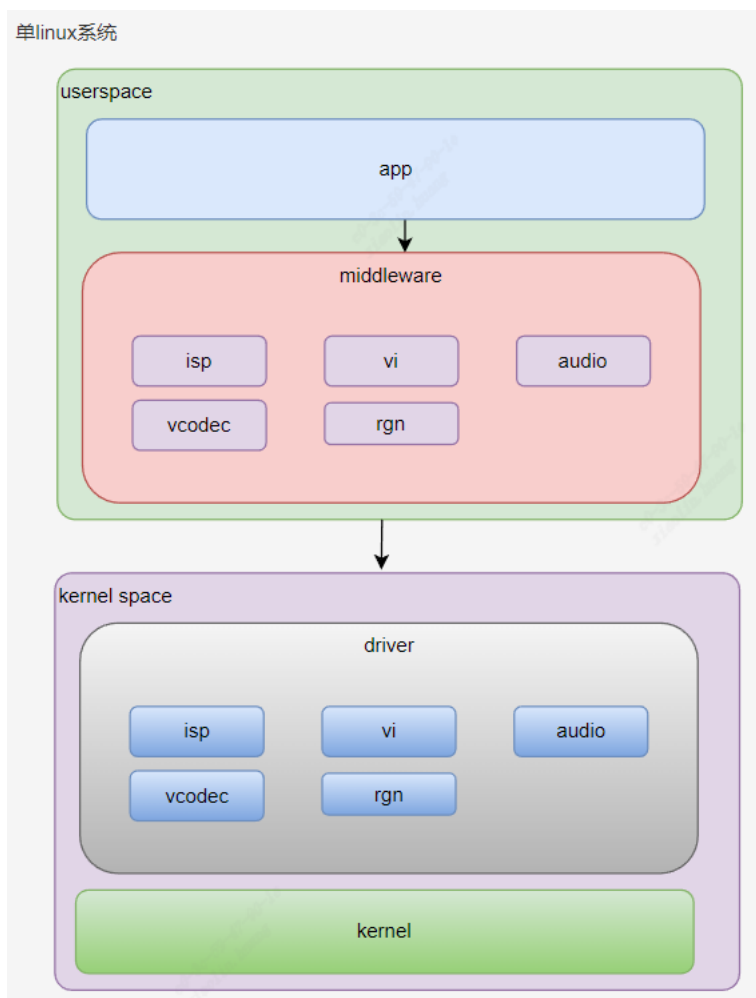


图 3.2: 单系统软件架构



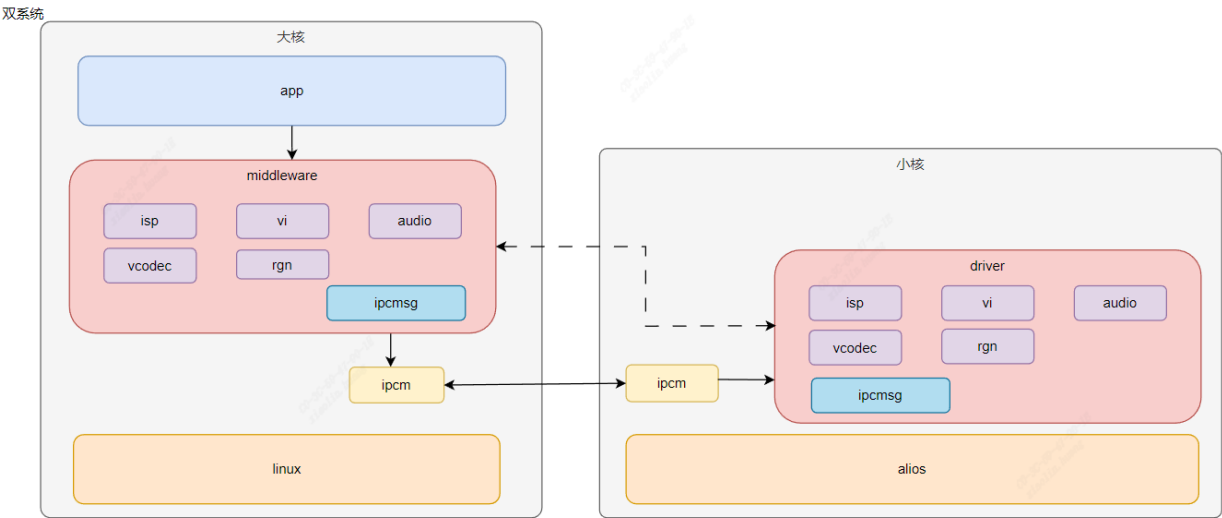


图 3.3: 双系统软件架构

### 3.4.2 双核通信

双核通信主要实现流程如图所示：

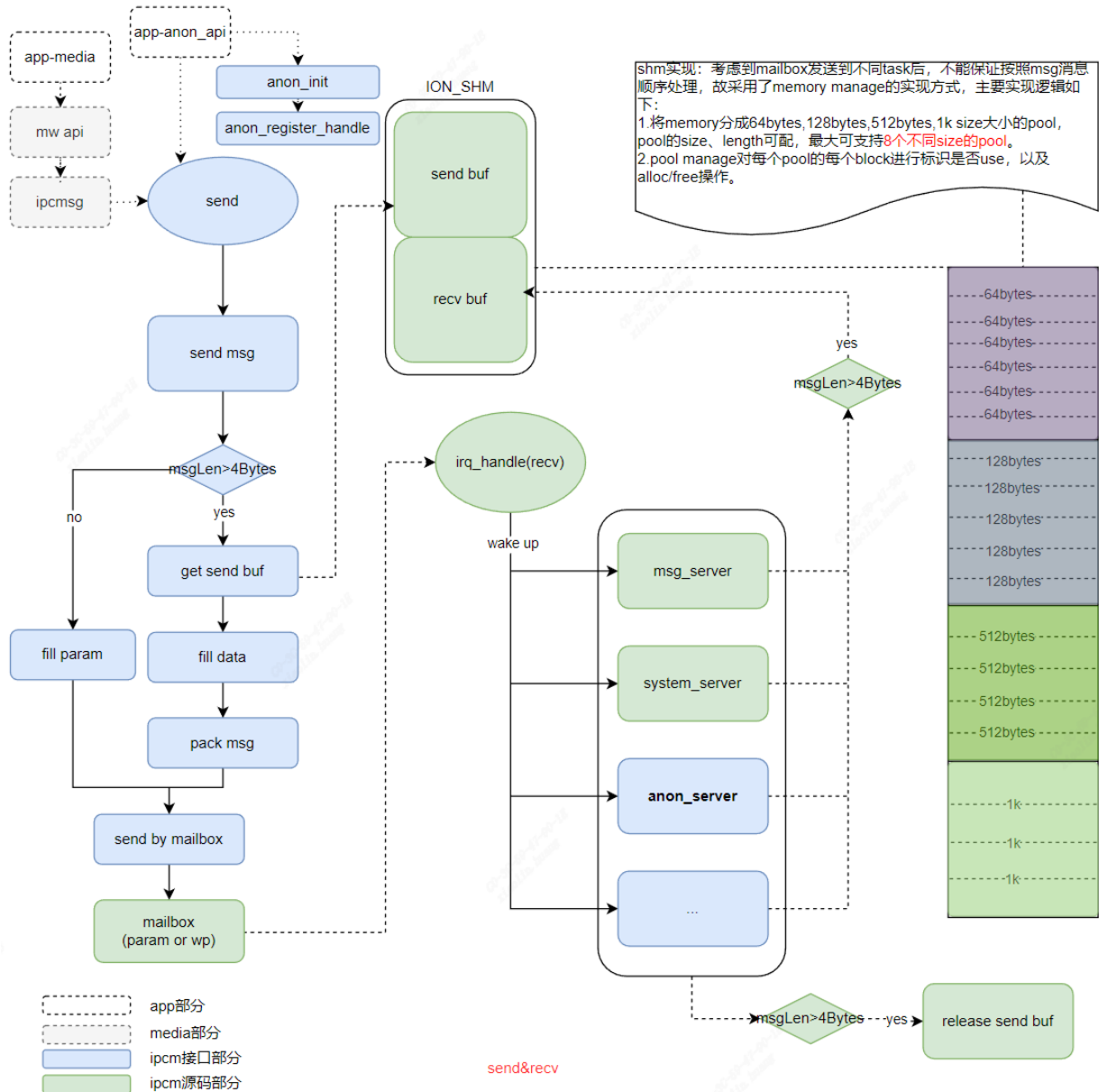


图 3.4: 双核通信流程

- 白色虚框代表 app 部分实现的逻辑
- 浅灰色虚框代表 media 部分实现的逻辑
- 浅蓝色框部代表 ipcm 提供的通信接口
- 浅绿色框代表 ipcm 实现的源码
- 多媒体 api 基本保持与单系统一致，底层调用 ipcmsg、ipcm 提供的相关接口发送 msg 到小核
- ipcm 提供 ipcm\_anon 相关 api 用于支持 app 层大小核通信:ipcm 客制化通信 api 介绍.pdf

## 3.5 单/双系统的差异

主要模块	单系统	双系统
build系统	source build/cvsetup.sh defconfig xxx build_all	基本跟单系统一致，新增部分选项
uboot & sd卡升级	编译生成fip.bin	双系统会额外编译uboot_spl & fip_spl.bin, fip_spl.bin会在升级过程中烧录flash
多媒体	Middleware api通过系统调用下发命令到driver, 在driver中进行多媒体处理; 需要在linux端 insmod vi/codec等ko	<b>API接口基本保持一致</b> ; Middleware api通过ipcmmsg(ipcm)接口发送消息到小核, 在小核进行多媒体处理; 不需要在linux端insmod vi/codec等ko
proc信息	通过/proc/cvitek/下面文件夹查看	在小核串口通过proc_xxx命令查看
memmap.py	主要包括linux sys和ion两部分	主要包括linux sys、linux ion、alios sys、alios resv(ion)等部分
flash 占用	-	开启COMPRESS_RTOS_BIN选项后, 与单系统基本一致
debug	uart0	当前需要uart0和uart1

### 3.5.1 新增编译选项

```
(Top) -> SDK options
C library (musl library for user mode application on riscv64) --->
[ ] Build static binary (no shared libs)
[*] Build SDK with debug config
[ ] Enable SDK sanitizer
[*] Do not install sample and self test application
[ ] Install the osdrv/extdrv/wireless/*.ko
[*] Do not compile frame buffer drivers
[*] Select CONFIG_NO_TP to build osdrv without Touchscreen driver(extdrv/tp)
[*] Boot Time Optimization
[ ] Shrink the rootfs
[*] Optimization of File System Compression Format
[ ] OpenSBI jump to Kernel instead of Uboot jump to Kernel
[ ] osdrv build-in instead of modules
[*] init process optimization
[ ] Select CONFIG_USB_OSDRV_CVITEK_GADGET to build osdrv with usb gadget cvg
[*] Make the boot image only have one dtb
[*] Compile 64MB DDR size project
[ ] enable the C906 DMA functions
```

#### [ ] Shrink the rootfs (NEW)

拿掉一些常用的busybox等相关命令，只保留一些必要的命令

#### [ ] Optimization of File System Compression Format (NEW)

针对文件系统的压缩格式进行优化，将默认压缩方式xz换成gzip，能提速200ms左右；但是会增大文件系统的大小；

#### [ ] OpenSBI jump to Kernel instead of Uboot jump to Kernel (NEW)

优化启动流程，由opensbi直接引导kernel启动，使用的是不压缩Image的方式来启动kernel；能提速200ms以上，但是会增大Kernel Size；如果Kernel Size太大，可以通过设置BUILD\_FOR\_DEBUG=n来裁剪kernel size；

#### [ ] osdrv build-in instead of modules (NEW)

把osdrv中相关代码build-in到kernel中，能提速150ms左右，对Flash Size影响不大；

ps：目前osdrv相关源码未开源，该选项当前主要内部使用

#### [ ] init process optimization (NEW)

把原本放在最后启动的用户程序挪到开机启动后第一个程序运行；



**快启模式：** 小核在启动后，从flash中拿到媒体参数，完成media pipeline

**非快启模式：** 小核启动到系统，只完成媒体部分初始化，然后大核通过middleware api给小核发送msg，完成media pipeline

#### [ \* ]Enable ALIOS

小核跑alios；必须选上；

#### [ \* ]Media In RTOS

媒体在rtos运行；必须选上；

#### [ ]DualOS fastboot Mode

用于配置小核是否启动Media Pipeline流程，初始化sensor/VI/VPSS等多媒体流程；

#### [ ]VCODEC fast boot In Dualos

用于配置venc bin文件是在小核加载还是在大核加载；

#### [ ]Compress rtos bin

用于配置是否对小核固件进行压缩；小核未压缩的情况下yoc.bin的size是1.84m，压缩后yoc.bin的size是791k；开启该选项会增加系统启动时间；

## 3.5.2 镜像包差异

data	2023/9/4 16:38	文件夹	
elf	2023/9/4 16:37	文件夹	
mw_musl_riscv64	2023/9/4 16:37	文件夹	
rawimages	2023/9/4 16:38	文件夹	
rootfs	2023/9/4 16:38	文件夹	
system	2023/9/4 16:38	文件夹	
tools	2023/9/4 16:38	文件夹	
tpu_musl_riscv64	2023/9/4 16:37	文件夹	
boot.spinor	2023/9/4 16:36	SPINOR 文件	3,794 KB
data.spinor	2023/9/4 16:38	SPINOR 文件	513 KB
fip.bin	2023/9/4 16:37	BIN 文件	512 KB
fip_spl.bin	2023/9/4 16:37	BIN 文件	185 KB
fw_jump.bin	2023/9/4 16:36	BIN 文件	100 KB
fw_payload_uboot.bin	2023/9/4 16:37	BIN 文件	2,549 KB
Image	2023/9/4 16:36	文件	6,313 KB
partition_spinor.xml	2023/9/4 16:38	XML 文档	1 KB
ramboot.itb	2023/9/4 16:37	ITB 文件	4,372 KB
rootfs.spinor	2023/9/4 16:38	SPINOR 文件	2,085 KB
upgrade.zip	2023/9/4 16:38	WinRAR ZIP 压缩...	8,191 KB
yoc.bin	2023/9/4 16:37	BIN 文件	790 KB

默认双系统会额外生成 fip\_spl.bin 和 yoc.bin 两个镜像文件：- fip\_spl.bin：主要包括 fsbl、opensbi、uboot\_spl.bin 等内容，通常烧录到 mtd0 分区 - yoc.bin：小核 alios 镜像 - fip.bin：主要包括 fsbl、opensbi、uboot.bin 等内容，在 sd/usb 升级时，通过加载并运行 fip.bin 中镜像完成

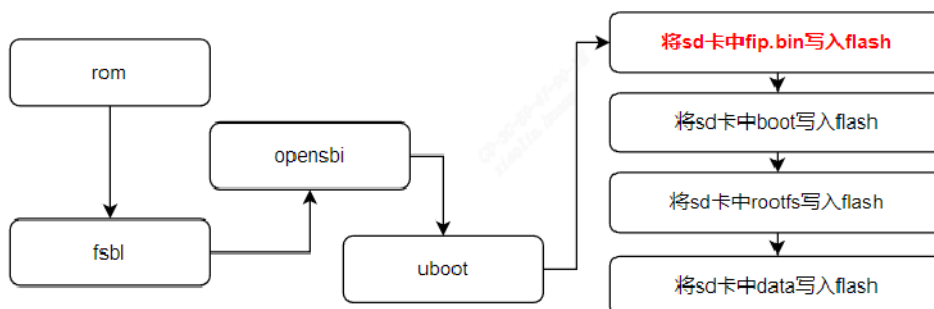
剩下的升级功能 - boot.spinor: linux kernel 镜像 - rootfs.spinor: rootfs 镜像 - data.spinor: data 分区镜像

### 3.5.3 uboot & sd 卡升级

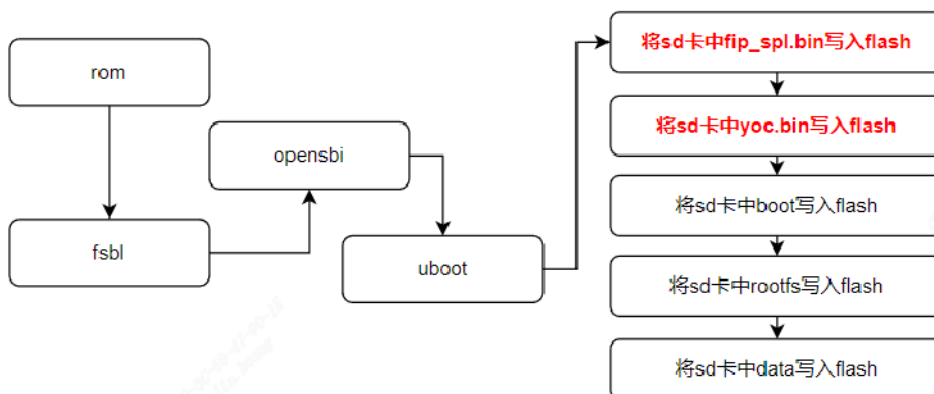
为了快速启动，双系统编译了 spl 版本的 uboot，最终会生成 fip.bin 和 fip\_spl.bin 两个 fip 文件。fip.bin 和 fip\_spl.bin 的差异如下图所示：uboot 内容不同，其他内容一致。



单系统sd卡升级



双系统sd卡升级



单系统和双系统 sd 卡升级流程及差异如下： - 单系统和双系统在插入 sd 卡后，如果 sd 卡存在合法的 fip.bin，都会进入升级流程 - 单系统会将 sd 卡中 fip.bin、boot、rootfs、data 等写入 flash - 双系统则会将 sd 卡中 fip\_spl.bin、yoc.bin、boot、rootfs、data 等写入 flash

在双系统固件中， - fip.bin：主要包括 fsbl、opensbi、uboot - fip\_spl.bin：主要包括 fsbl、opensbi、uboot\_spl - yoc.bin：小核 alios 镜像文件 - boot：大核 kernel 镜像文件 - rootfs：大核文件系统镜像文件 - data：数据分区镜像文件

### 3.5.4 xml 分区

双系统根据产品需求可能新增 2nd、PQ、PARAM、PARAM\_BAK 等分区。

```
<physical_partition type="spinor">
  <partition label="fip" size_in_kb="1024" readonly="false" file="fip.bin"/>
  <partition label="2nd" size_in_kb="3072" readonly="false" file="yoc.bin"/>
  <partition label="PQ" size_in_kb="1024" readonly="false" file="pq_data.spinor"/>
  <partition label="BOOT" size_in_kb="3072" readonly="false" file="boot.spinor"/>
  <partition label="MISC" size_in_kb="128" readonly="false" file="logo.jpg"/>
  <partition label="PARAM" size_in_kb="64" file="" />
  <partition label="PARAM_BAK" size_in_kb="64" file="" />
  <partition label="ENV" size_in_kb="64" file="" />
  <partition label="ENV_BAK" size_in_kb="64" file="" />
  <partition label="ROOTFS" size_in_kb="6800" readonly="false" file="rootfs.spinor" />
  <partition label="DATA" size_in_kb="1024" readonly="false" file="data.spinor" mountpoint="/mnt/data" type="j"
</physical_partition>
```

图 3.5: xml 分区

### 3.5.5 memmap

- 以 cv1811h\_wevb\_0007a\_spinor 为例分别列出了目前单系统和双系统的内存 layout
- 单系统启动后，内存 layout 如图左边所示：包括 linux、ion、freertos 三部分实际项目如果没有跑 freertos 系统，freertos 部分可以拿掉
- 双系统启动后，内存 layout 如图右边所示：包括 linux、ion、alios、alios ion、alios logo、param、param bak、pq param、ipcm shm 等部分

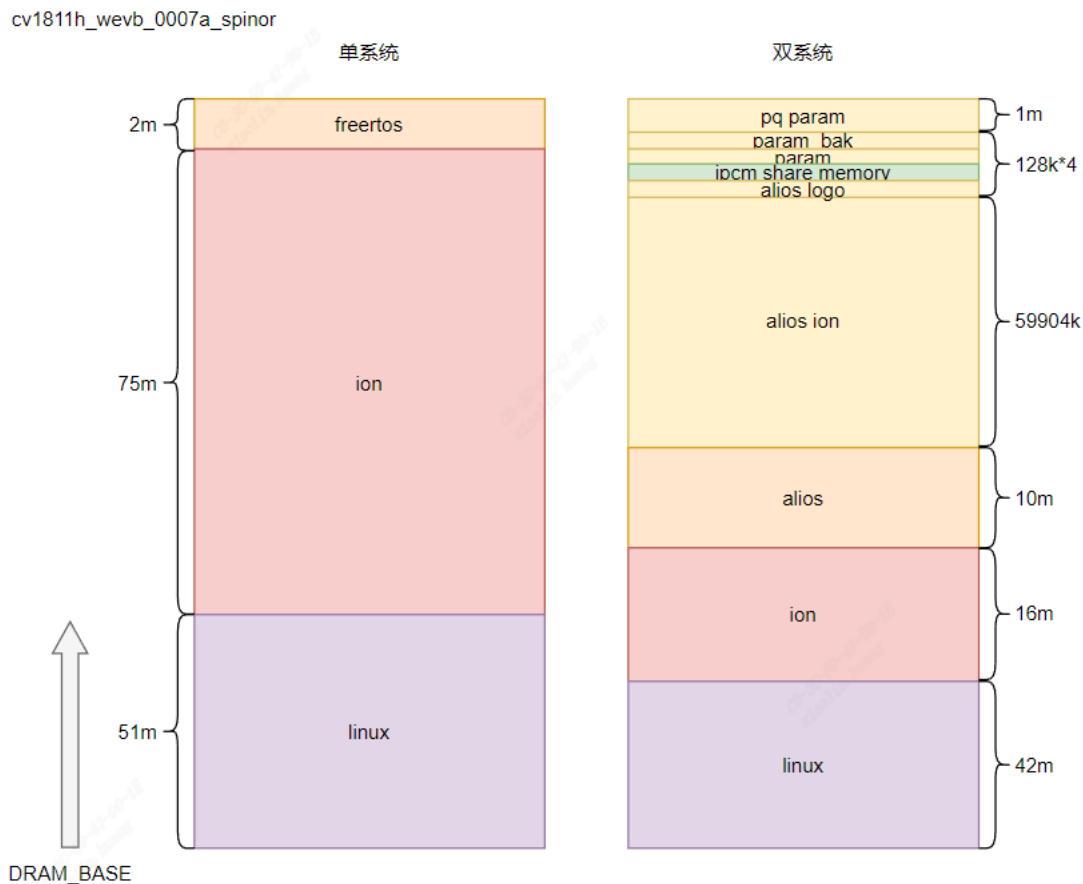


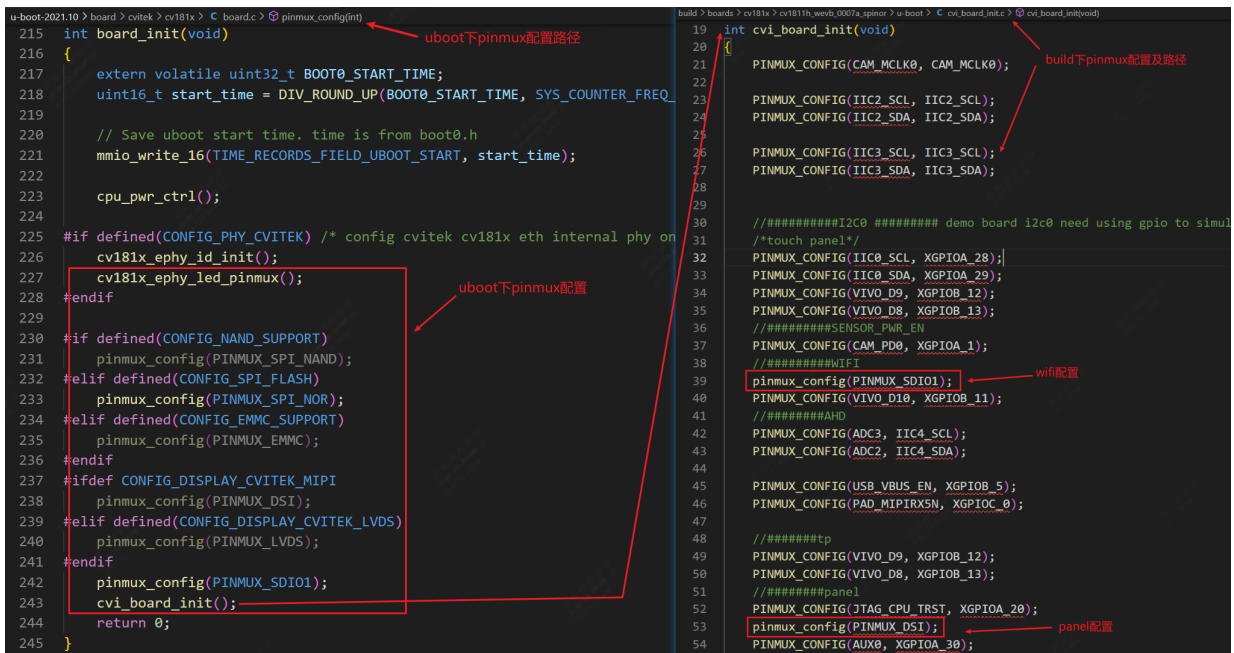
图 3.6: 单/双系统的内存 layout

## 3.6 PINMUX 使用

### 3.6.1 单系统中的 PPINMUX 配置

在单系统情况下, 通常需要在 U-Boot 中进行 PINMUX 配置。对于 cv1811h\_wevb\_0007a\_spinor 板卡, PINMUX 配置路径如下: - 在 u-boot-2021.10/board/cvitek/cv181x/board.c - 在 build/boards/cv181x/cv1811h\_wevb\_0007a\_spinor/u-boot/cvi\_board\_init.c

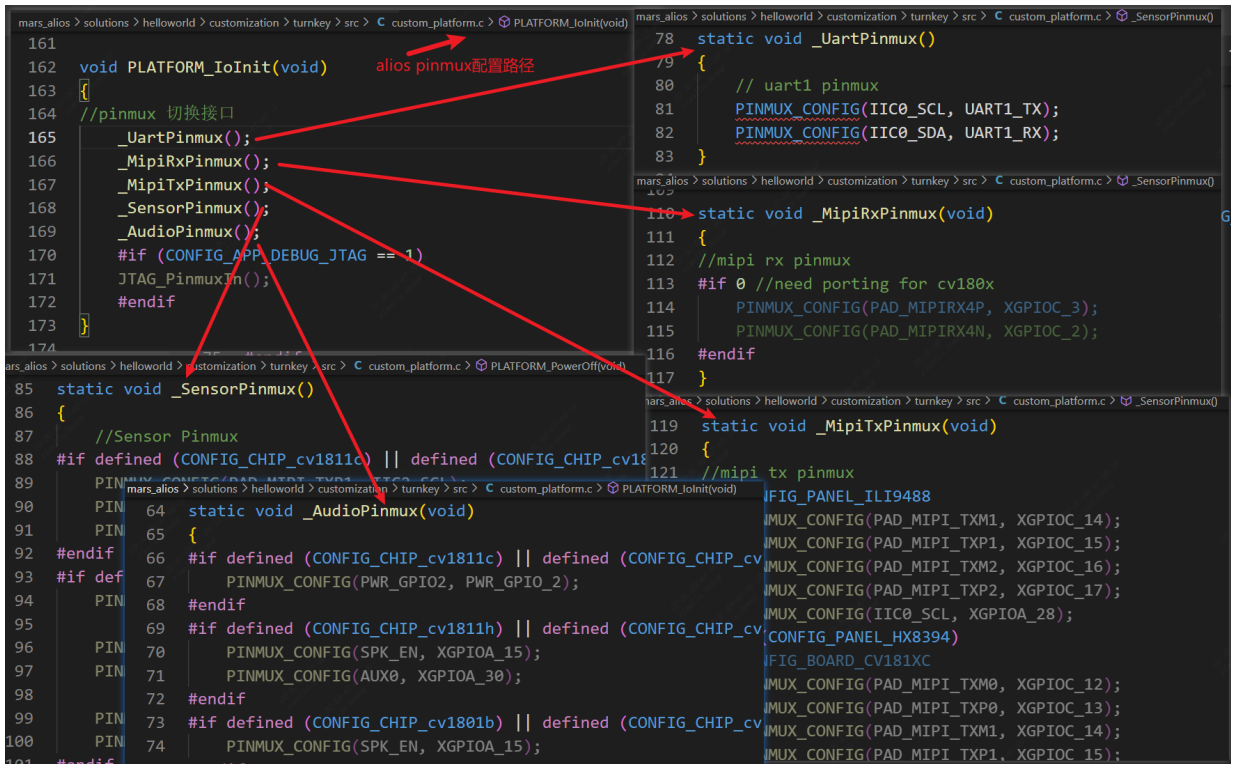




· 上图展示了 PINMUX 配置路径的不同之处。

### 3.6.2 双系统中的 PINMUX 配置

双系统场景 alios 也会进行 pinmux 配置, alios pinmux 配置路径如下: - 在 `cv1_alios/solutions/normboot/customization/turnkey/src/custom_platform.c`



## 3.7 常见的外设使用

GPIO 使用：参考 PeripheralDriver\_zh.pdf 7 GPIO 操作指南

PWM 使用：参考 PeripheralDriver\_zh.pdf 10 PWM 操作指南或参考 sample: middleware/v2/sample/common/sample\_common\_peripheral.c

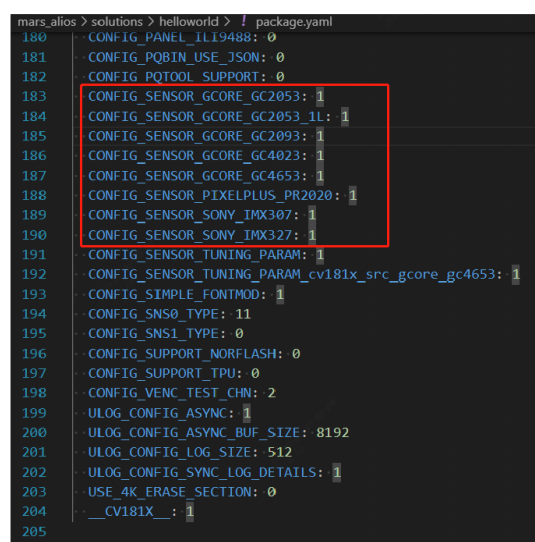
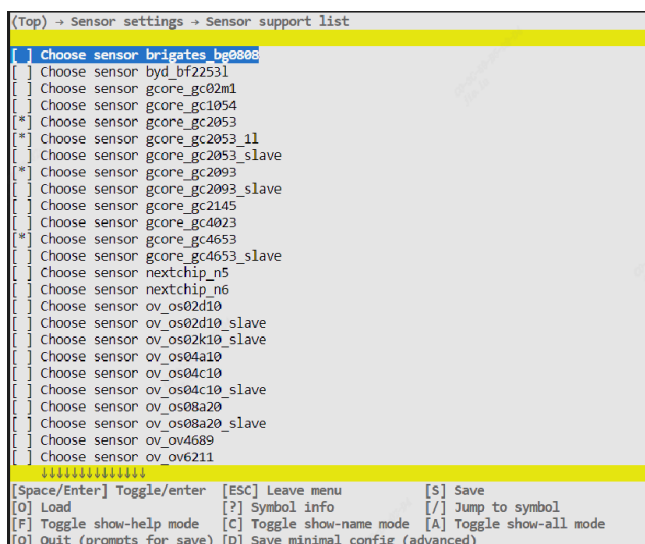
## 3.8 外设更换 demo

### 3.8.1 Flash

- 对于 Support list 中支持的 Flash 型号，无需额外适配。
- 如果所需 Flash 型号未在 Support list 中列出，参考文档 nor\_nand 支持.docx。

### 3.8.2 Sensor

- 无论是快启或非快启，在编译时，需要在 menuconfig -> Sensor settings -> Sensor support list 中勾选需要使用的对应 Sensor。
- 勾选后，会生成相应参数传递给 AliOS 系统，并打开对应的宏，使得 Sensor 对应的对象可以被调用。
- 在右侧的 AliOS 文件中，可以看到相应的 Sensor 被打开。



### 3.8.2.1 非快启

在非快启模式下，操作与单系统保持一致，需要通过 INI 文件进行配置以使用不同的 Sensor。以下是一个示例

```
--> [source]
--> dev_num = 1 -----表示sensor个数
--> [sensor] -----表示sensor编号
--> name = GCORE_GC4653_MIPI_4M_30FPS_10BIT -----表示sensor ID
--> bus_id = 2 -----表示此sensor用的I2C bus ID
--> sns_i2c_addr = 41 -----表示此sensor用的I2C address
--> mipi_dev = 0 -----表示使用的mac number
--> lane_id = 3, 2, 4, -1, -1 -----此sensor使用的lane配置
--> pn_swap = 0, 0, 0, 0, 0 -----此sensor使用的pin swap配置
--> mclk_en = 1 -----是否使用ini中配置的mclk
--> mclk = 1 -----配置的mclk ID
--> port = 2 -----sensor的rst pin的portID
--> pin = 13 -----sensor的rst pin的pin
--> pol = 1 -----sensor的rst pin的有效状态, cvi_
↪ comm_cif.h
```

### 3.8.2.2 快启

在快启模式下，需要配置 Sensor 对应硬件的 RST 配置，否则 Sensor 无法工作，I2C 通信也受阻。在左侧文件中配置相应硬件信息，包括 I2C 地址、I2C 总线 ID、RST 引脚的端口、引脚、有效状态等。所有配置内容可在右侧文件中找到。

```
mars.alios > solutions > helloworld > package.yaml > package.yaml.turnkey
176 ULOG_CONFIG_SYNC_LOG_DETAILS: 1
177 ULOG_CONFIG_ASYNC_BUF_SIZE: 8192
178 ULOG_CONFIG_LOG_SIZE: 512
179 USE_4K_ERASE_SECTION: 0
180 CONFIG_SIMPLE_FONTMOD: 1
181 CONFIG_SUPPORT_NOFLASH: 0
182 CONFIG_SNS0_TYPE: 11
183 CONFIG_SNS1_TYPE: 0
184 CONFIG_PANEL_HX8394: 0
185 CONFIG_PANEL_ILI9488: 0
186 CONFIG_VENC_TEST_CHN: 2
187 CONFIG_PQTOOL_SUPPORT: 0
188 CONFIG_PQBTN_USE_JSON: 0
189 CONFIG_ISP_SUPPORT_PROC: 0
190 CONFIG_ENABLE_FASTBOOT: 0
191
192 CONFIG_DUALOS_NO_CROP: 0 # 1 不裁剪 0 裁剪
193
```

```
mars.alios > components > cvi_mmio_sdk > cvi_sensor > sensor_cfg > C sensor_cfg.h
27 typedef enum _SNS_TYPE_E {
28     SNS_TYPE_NONE = 0,
29     /* ----- LINEAR BEGIN ----- */
30     SONY_IMX327_MIPI_2M_30FPS_12BIT,
31     SONY_IMX307_MIPI_2M_30FPS_12BIT,
32     SONY_IMX307_2L_MIPI_2M_30FPS_12BIT,
33     SONY_IMX307_SLAVE_MIPI_2M_30FPS_12BIT,
34     GCORE_GC0308_MIPI_1M_30FPS_8BIT,
35     GCORE_GC1054_MIPI_1M_30FPS_10BIT,
36     GCORE_GC2053_MIPI_2M_30FPS_10BIT,
37     GCORE_GC2053_1L_MIPI_2M_30FPS_10BIT,
38     GCORE_GC2093_MIPI_2M_30FPS_10BIT,
39     GCORE_GC2145_MIPI_2M_12FPS_8BIT,
40     GCORE_GC02M1_MIPI_2M_30FPS_10BIT,
41     GCORE_GC02M1_SLAVE_MIPI_2M_30FPS_10BIT,
42     GCORE_GC4023_MIPI_4M_30FPS_10BIT,
43     GCORE_GC4653_MIPI_4M_30FPS_10BIT,
44     PIXELPLUS_PR2020_1M_25FPS_8BIT,
45     PIXELPLUS_PR2020_1M_30FPS_8BIT,
46     PIXELPLUS_PR2020_2M_25FPS_8BIT,
47     PIXELPLUS_PR2020_2M_30FPS_8BIT,
48     TECHPOINT_TP9950_1M_30FPS_8BIT,
49     TECHPOINT_TP9950_2M_30FPS_8BIT,
50     TECHPOINT_TP9950_1M_25FPS_8BIT,
```

- 在快启模式下，需要配置 sensor 的对应硬件的 rst 配置，否则 sensor 无法工作，I2C 也是不通的。
- 在如下左图文件中配置对应硬件配置的信息，包括 I2C address, I2C busID, rst pin 的 port, pin, 有效状态等等，配置的全部内容可在图右文件中参考。

```

mars_aliOS > solutions > helloworld > customization > turnkey > cv181xc_evb_qfn_param > C custom_viparam.c
3  /*
4  * File Name: custom_viparam.c
5  * Description:
6  * .....
7  */
8  #include "custom_param.h"
9
10 PARAM_CLASSDEFINE(PARAM_SNS_CFG_S, SENSORCFG, CTX, Sensor)[] = {
11 {
12     .enSnsType = CONFIG_SNS0_TYPE,
13     .s32I2cAddr = -1,
14     .s8I2cDev = 2,
15     .u32Rst_port_idx = 2, //GPIOC_13
16     .u32Rst_pin = 13,
17     .u32Rst_pol = OF_GPIO_ACTIVE_LOW,
18 }
19 };
20

```

```

mars_aliOS > components > cv1_platform > param > include > C param_vih
12 typedef struct _PARAM_SNS_CFG_S {
13     CVI_S32 s32SnsId;
14
15     SNS_TYPE_E enSnsType;
16     HDR_MODE_E enHDRMode;
17     CVI_S32 s32Framerate;
18
19     CVI_S8 s8I2cDev;
20     CVI_S32 s32BusId;
21     CVI_S32 s32I2cAddr;
22     CVI_S32 s32MipiDev;
23     CVI_S16 as16LaneId[5];
24     CVI_S8 as8PNSwap[5];
25
26     CVI_BOOL bMc1kEn;
27     CVI_U8 u8Mc1k;
28     CVI_S32 u8Orien;
29     CVI_BOOL bHwSync;
30     CVI_U8 u8UseDualSns;
31
32     CVI_U32 u32Rst_port_idx;
33     CVI_U32 u32Rst_pin;
34     CVI_U32 u32Rst_pol;
35     CVI_U8 u8Rotation;
36 }PARAM_SNS_CFG_S;
37

```

- 在快启模式下，sensor 通信使用的 I2C 需要在如下地方配置，需要将对应的 pinmux 切对，才能正确使用 I2C 来初始化和控制 sensor。
- 具体控制 pinmux 切换请参考此页靠下图。

```

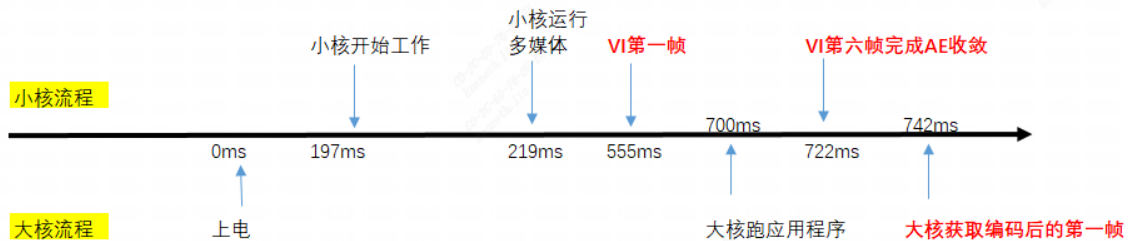
mars_aliOS > solutions > helloworld > customization > turnkey > src > C custom_platform.c
53 static void _SensorPinmux()
54 {
55     PINMUX_CONFIG(PAD_MIPI_TXP1, IIC2_SCL);
56     PINMUX_CONFIG(PAD_MIPI_TXM1, IIC2_SDA);
57     PINMUX_CONFIG(PAD_MIPI_TXM0, CAM_MCLK1);
58 }
59

```

VDD18A_MIPI 1P8V ONLY		VDD18A_MIPI		82	VDD18A_MIPI
CV_4WTDI_CR_SCL0/VIO_CLK/V11_D[13]/XGPIOC[2]/IIC1_SDA/CAM_MCLK0/KEY_ROW0/MUX_SPI1_SCK	[PD]	PAD_MIPIRX4N	72	10	MIPI_RX0_4N
CV_4WTMS_CR_SDA0/VIO_D[0]/V11_D[14]/XGPIOC[3]/IIC1_SCL/CAM_MCLK1/KEY_ROW1/MUX_SPI1_CS	[PD]	PAD_MIPIRX4P	73	10	MIPI_RX0_4P
(CV_4WTDI_CR_2WTMS/VIO_D[1]/V11_D[15]/XGPIOC[4]/CAM_MCLK0/MUX_SPI1_MISO)	[PD]	PAD_MIPIRX3N	74	10	MIPI_RX0_3N
(CV_4WTCK_CR_2WTCK/VIO_D[2]/V11_D[16]/XGPIOC[5]/MUX_SPI1_MOSI)	[PD]	PAD_MIPIRX3P	75	10	MIPI_RX0_3P
((PAD_MIPIRX2N)/VIO_D[3]/VIO_D[10]/XGPIOC[6]/V11_D[17]/IIC4_SCL/DBG[6])	[PD]	PAD_MIPIRX2N	76	10	MIPI_RX0_2N
((PAD_MIPIRX2P)/VIO_D[4]/VIO_D[9]/XGPIOC[7]/V11_D[18]/IIC4_SDA/DBG[7])	[PD]	PAD_MIPIRX2P	77	10	MIPI_RX0_2P
((PAD_MIPIRX1N)/VIO_D[5]/VIO_D[8]/XGPIOC[8]/KEY_ROW3/DBG[8])	[PD]	PAD_MIPIRX1N	78	10	MIPI_RX0_1N
((PAD_MIPIRX1P)/VIO_D[6]/VIO_D[7]/XGPIOC[9]/IIC1_SDA/KEY_ROW2/DBG[9])	[PD]	PAD_MIPIRX1P	79	10	MIPI_RX0_1P
((PAD_MIPIRX0N)/VIO_D[7]/VIO_D[6]/XGPIOC[10]/IIC1_SCL/CAM_MCLK1/DBG[10])	[PD]	PAD_MIPIRX0N	80	10	MIPI_RX0_0N
((PAD_MIPIRX0P)/VIO_D[8]/VIO_D[5]/XGPIOC[11]/CAM_MCLK0/DBG[11])	[PD]	PAD_MIPIRX0P	81	10	MIPI_RX0_0P
(CV_4WTDI_CR_SDA0/VIO_D[13]/VIO_D[0]/XGPIOC[16]/IIC1_SDA/PWM[8]/SPI0_SCK/SD1_D2)	[PD]	PAD_MIPI_TXM2	83	10	MIPI_TX02
(CV_4WTDI_CR_SCL0/VIO_D[14]/VIO_CLK0/XGPIOC[17]/IIC1_SCL/PWM[9]/SPI0_CS_X/SD1_D3)	[PD]	PAD_MIPI_TXP2	84	10	MIPI_TX01
(CV_4WTDI_CR_2WTMS/VIO_D[11]/VIO_D[2]/XGPIOC[14]/IIC2_SDA/PWM[10]/SPI0_SDO/DBG[14])	[PD]	PAD_MIPI_TXM1	85	10	MIPI_TX01
CR_2WTCK/VIO_D[12]/VIO_D[1]/XGPIOC[15]/ANA_TESTOUT_VIVD/IIC2_SCL/PWM[11]/SPI0_SDI/DBG[15])	[PD]	PAD_MIPI_TXP1	86	10	MIPI_TX00
((PAD_MIPI_TXM0)/VIO_D[9]/VIO_D[4]/XGPIOC[12]/CAM_MCLK1/PWM[14]/CAM_VS0/DBG[12])	[PD]	PAD_MIPI_TXM0	87	10	MIPI_TX00
((PAD_MIPI_TXP0)/VIO_D[10]/VIO_D[3]/XGPIOC[13]/CAM_MCLK0/PWM[15]/CAM_HS0/DBG[13])	[PD]	PAD_MIPI_TXP0	88	10	MIPI_TX00

## 3.9 常用 Debug 方法

### 3.9.1 快启上电到出第一张收敛后的图像各阶段的耗时



小核：指的是C906L

大核：指的是C906B

小核开始工作：芯片上电后，小核默认处于复位状态；大核加载小核固件到DDR后，解除小核复位状态；

小核运行多媒体：小核开始初始化sensor、VI、VPSS、VENC；

VI第一帧：VI第一帧，并进行isp处理

VI第六帧完成AE收敛：在标定范围内的环境亮度下，5帧能完成AE收敛，所以这里统计最大值第六帧；VENC会把前面未收敛的5帧丢掉；

大核获取编码后的第一帧：经过VENC的第一帧，为小核AE收敛后的帧数，即第六帧；

### 3.9.2 常见问题

Q：双系统 proc 信息怎么查看？A：通过小核串口，输入 ‘proc\_xxx’ 可以查看对应模块的 proc 信息。- proc 信息说明参考《MMF 媒体软件开发指南》第 12 节：Proc 调试信息说明。

Q：双核通信驱动 log 怎么打开？A：在某些情况下，遇到问题需要打开双核通信驱动 log 进行 debug：

- 小核打开双核通信驱动 log：
- ipcm\_dbg pool\_info：show pool get/rls callstack
- ipcm\_dbg log0\_on：open ipcm recv and send log
- ipcm\_dbg log0\_off：close ipcm recv and send log
- ipcm\_dbg logall\_on：open all log(set log level to debug)
- ipcm\_dbg logall\_off：reset log level to default
- 大核打开双核通信驱动 log：
- echo 1 > /sys/kernel/debug/ipcm/log0 #open msg send and recv log
- echo 0 > /sys/kernel/debug/ipcm/llog0 #close msg send and recv log
- echo 4 > /sys/kernel/debug/lipcm/loglevel #set ipcm log to debug
- echo 3 > /sys/kernel/debug/ipcm/lloglevel #set ipcm log to info
- 双核通信驱动 log 打开后会影响到双核通信的效率，可能会影响的 media 的时序

### 3.9.3 Linux Crash SOP

参考《oops 使用指南.docx》

### 3.9.4 小核 Crash

- 小核 crash 后，在小核串口会打印 crash 信息，将 crash 信息保存到文本，
- 然后通过 coredump\_parser\_mmleak\_riscv.py 进行解析。

```
> ipcm > plat > alios > message > C ipcm_msg_proc.c > ...
23 static int ipcm_info_show(void)
24 {
25     int *p = NULL;
26     ipcm_info("mailbox not valid cnt:%d\n",
27     *p = 123;
28     return 0;
29 }
```

构建crash代码

```
(cli-uart)# ipcm_dbg info
<info>[ipcm_info_show:26]mailbox not valid cnt:0
Exception ++++++ MEPC 0x83ab8b92, MSTATUS 0x8000000a00007980, CPU Exception: NO.0x7

!!!!!!! Exception !!!!!!!!
crash time : 1970-01-01 16:00:18
current task : cli-uart
===== Regs info =====
X1(ra) 0x0000000083AB8B92
X2(sp) 0x0000000083C5C030
X3(gp) 0x0000000083BD7B90
X4(tp) 0x0404040404040404
X5(t0) 0x0000000083BD8AE8
X6(t1) 0x0000000000000030
X7(t2) 0x0000000083B5AAB4
X8(s0) 0x0000000083BD0D41
X9(s1) 0x0000000083BD0D38
```

串口crash信息

- 将上述右图串口 crash 信息保存到 crash\_log 文件，执行 python2 coredump\_parser\_mmleak\_riscv.py crash\_log yoc.elf 可以将 crash 信息解析出来。

```
crash time : 1970-01-01 16:00:18
current task : cli-uart
===== Regs info =====
X1(ra) 0x0000000083AB8B92
X2(sp) 0x0000000083C5C030
X3(gp) 0x0000000083BD7B90
X4(tp) 0x0404040404040404
```

```
===== Call stack =====
backtrace : 0x83A2206E
proc_onecmd at /media/cvitek/allen.huang/code/dualos/mars_alios/components/cli/src/cli.c:151
backtrace : 0x83A221C8
cli_handle_input at /media/cvitek/allen.huang/code/dualos/mars_alios/components/cli/src/cli.c:273 (discriminator 4)
backtrace : 0x83A2280A
cli_main at /media/cvitek/allen.huang/code/dualos/mars_alios/components/cli/src/cli.c:1053
backtrace : "task entry"
```

```
===== Heap Info =====
heap info
[HEAP] TotalSz FreeSz UsedSz MinFreeSz MaxFreeBlkSz
0x007AC980 0x00758E40 0x00053E40 0x007589C0 0x00758E40

[HEAP] TotalSz FreeSz UsedSz MinFreeSz MaxFreeBlkSz
0x03AC0000 0x03ABFD40 0x000002C0 0x03ABFD40 0x03ABFD40
```



### 3.9.5 小核 Memory 分析

- 通过 `dumpsys mm_info` 命令，将内存使用信息 dump 出来，并保存到文本 `mm_log`。（如果分析 reserved 内存，则使用 `dumpsys mm_info_resv` 命令）
- 通过上一页的 `coredump_parser_mmleak_riscv.py` 脚本解析 memory 信息（执行 `python2 coredump_parser_mmleak_riscv.py mm_log yoc.elf`）。
- 解析之后可以看到详细的内存分配和使用情况，如果 Alloc Cnt 明显很大，则有可能存在内存泄漏。

```
(cli-uart)# dumpsys mm_info
```

dump memory信息

```
----- all memory blocks -----
g_kmm_head = 83c53680
ALL BLOCKS
Blk Addr   Stat   Len  Chk   Caller   Point
0x83c53840 used    64  OK    0x0     (0x0 <- 0x0 <- 0x0 <- 0x0)
0x83c538c0 used   8192 OK 0x83a1f63c (0x0 <- 0x0 <- 0x0 <- 0x0)
0x83c55900 used   320  OK 0x83a1f64c (0x0 <- 0x0 <- 0x0 <- 0x0)
0x83c55a80 used   128  OK 0x83a21426 (0x83a21422 <- 0x83a21754 <- 0x83b55340 <- 0x83b552a4)
0x83c55b40 used   128  OK 0x83a1cb76 (0x83a1cb72 <- 0x83a20fc2 <- 0x83a26780 <- 0x83a2177a)
0x83c55c00 used   256  OK 0x83a26792 (0x83a2678e <- 0x83a2177a <- 0x83b55340 <- 0x83b552a4)
0x83c55d40 used   128  OK 0x83a21426 (0x83a21422 <- 0x83a267c2 <- 0x83a2177a <- 0x83b55340)
0x83c55e00 used   128  OK 0x83a21426 (0x83a21422 <- 0x83a267d6 <- 0x83a2177a <- 0x83b55340)
```

parser后的内存信息情况

Caller	Func	Alloc Cnt	Total Size	Line	File
0x83a1f63c	task_dyn_create	12	160768	278	/media/cvitek/allen.huang/code/daalos/mars_alios/components/rhino/k task.c
0x83a1f63c	ipcm_port_init	2	32896	125	/media/cvitek/allen.huang/code/daalos/mars_alios/components/cvi_mmf_sdk/cvi_osdrv/cvi_osdrv_ipcm/ipcm/core/ipcm.c
0x83a74038	vpss_core_init	1	32768	1398	/media/cvitek/allen.huang/code/daalos/mars_alios/components/cvi_mmf_sdk/cvi_osdrv/cvi_osdrv_vpss/chip/cv181x/vpss_core.c
0x83a799c0	ldc_create_instance	1	32768	116	/media/cvitek/allen.huang/code/daalos/mars_alios/components/cvi_mmf_sdk/cvi_osdrv/cvi_osdrv_ldc/common/ldc_platform.c
0x83a799c0	vpss_core_init	1	8896	1375	/media/cvitek/allen.huang/code/daalos/mars_alios/components/cvi_mmf_sdk/cvi_osdrv/cvi_osdrv_vpss/chip/cv181x/vpss_core.c
0x83a39494	csi_uart_init	1	8192	226	/media/cvitek/allen.huang/code/daalos/mars_alios/components/cvi_mmf_sdk/chip/cv181x/src/drivers/uart/uv/v2.0/src/uart.c
0x83a4e0da	vi_create_proc	1	8192	4440	/media/cvitek/allen.huang/code/daalos/mars_alios/components/cvi_mmf_sdk/cvi_osdrv/cvi_osdrv_vi/chip/cv181x/vi.c