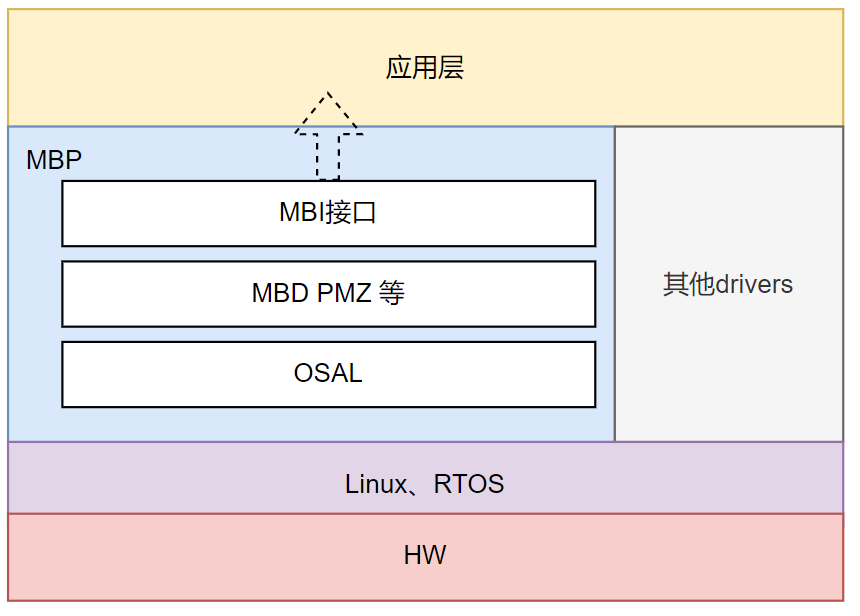
C3-MBP

# 1.概述

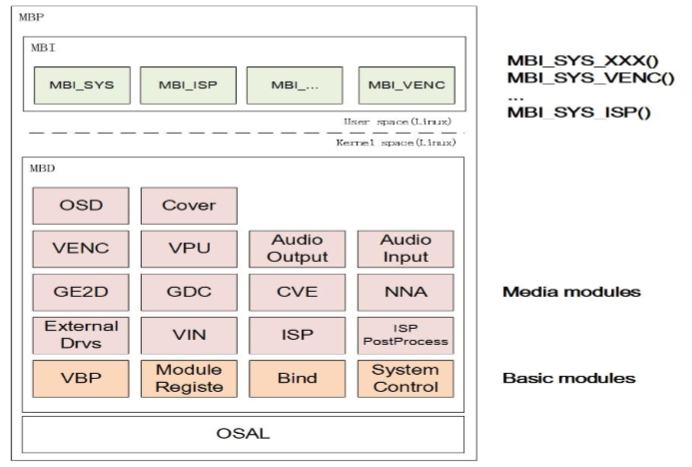
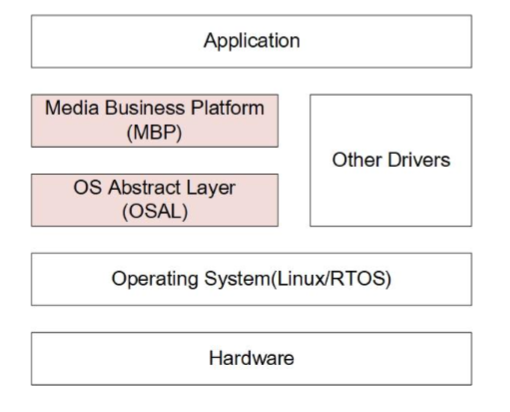
MBP(Media Business Platform)，目的是优化多媒体业务系统的维护成本和方便多人开发媒体业务驱动模块。MBP提供一套基于模块管理组件，当多人进行并行开发媒体业务驱动模块时，只需关注对应模块xxx.h 中的 API即可。

优化C1和C2一些软件问题，跳过传统的alsa，V4L2等接口。使得移植不同的操作系统（Linux/RTOS）很方便。

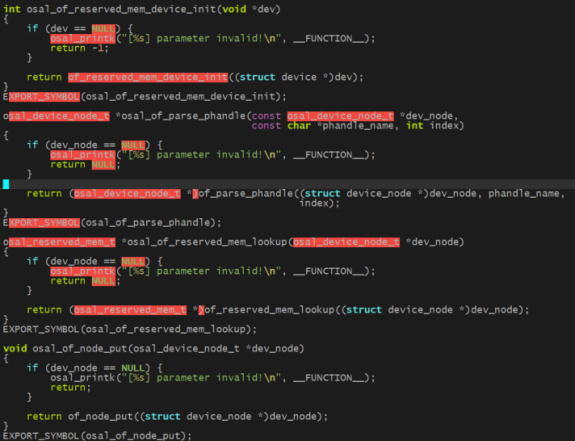
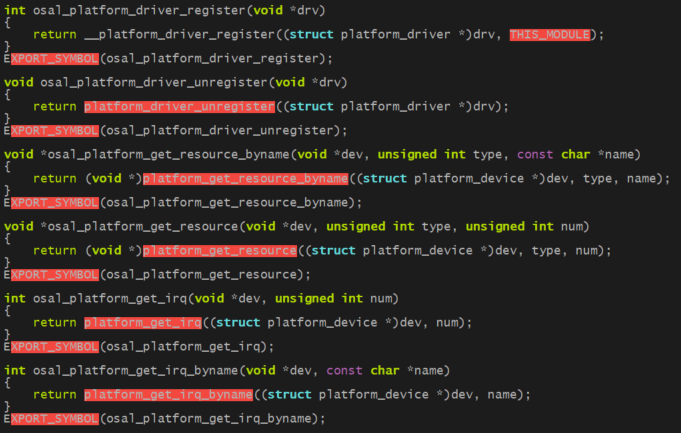


# 2.MBP框架

系统层次图如下所示：



MBP包括MBD（媒体业务驱动程序）、MBI（媒体业务接口），MBD基于OSAL，主要由各种媒体驱动程序组成，MBI是每个媒体驱动程序（ioctl）的封装（实际上是在kernel原始API上套一层，如下代码图）。



## 2.1 mbp工作pipeline

**video:**

sensor -- 视频输入 -- VPU视频处理 -- VENC视频编码-- 视频输出 -- 屏幕

**audio:**

麦克风 -- 音频输入 -- 音频编码(AENC) -- 音频码流 -- 音频解码(ADEC) -- 喇叭

# 3.MBP Module

## 3.1 module的设计

**（1）模块绑定功能的初始化/退出**

初始化部分应该实现在对应的模块驱动的初始化函数中。同理，退出功能应该实现在对应的模块驱动的退出函数中。

**（2）绑定功能的核心结构：binder**

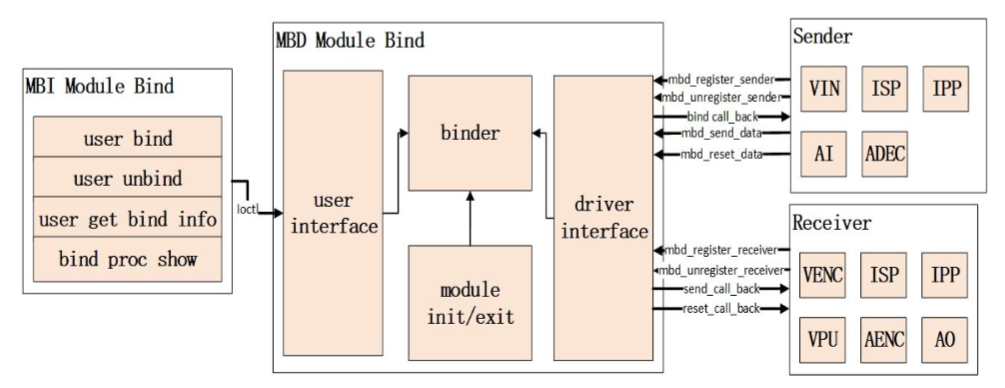
Binder 的核心结构是两个链表

**（3）绑定功能提供给其他驱动调用的接口**

通过 module register 注册为私有的驱动模块，并提供对外的驱动层接口，方便其他驱动模块使用。

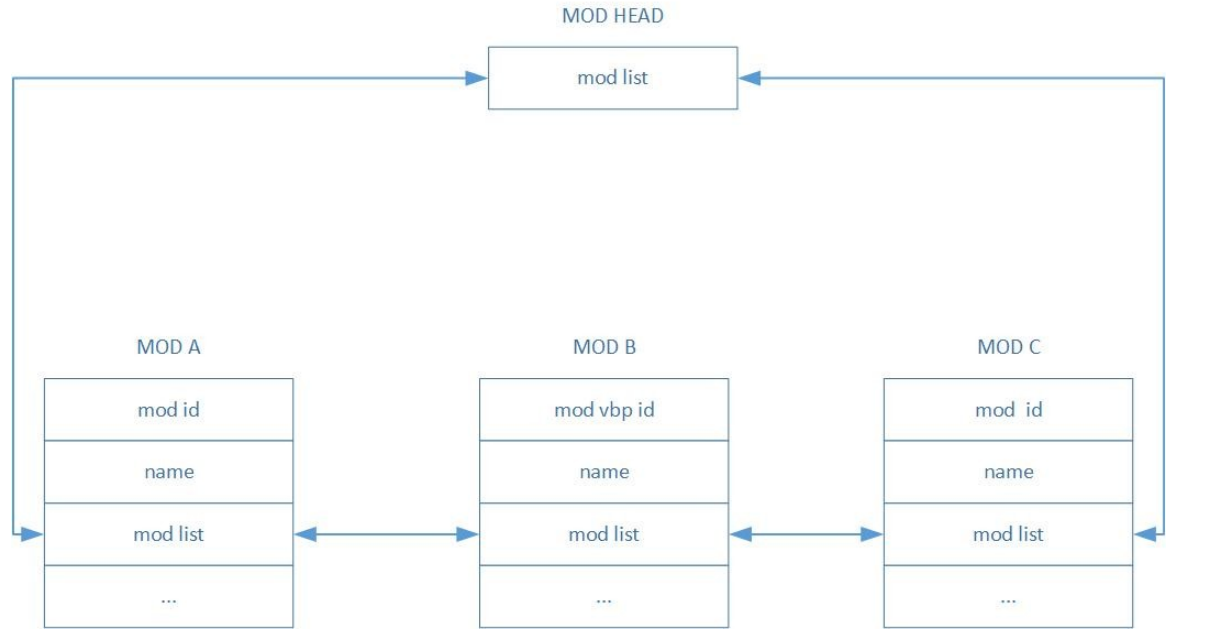
**（4）绑定功能提供给用户层使用的接口**

Linux 上通过一般的 ioctl，提供用户层接口，并将其封装成 MBI 接口。



## 3.2 Module注册

模块管理组件提供一组通用接口，每个驱动模块抽象出与之对应的对外接口，各个模块通过 **模块描述符** 链表进行管理，各个模块以注册的方式加入到 模块描述符 链表中。



## 3.3 Module使用

Module Bind 的使用可分为三个步骤

### 3.3.1注册

- 在业务驱动模块初始化阶段，需要先通过 module register 注册成 MBD (Media Business Driver)

- 在业务驱动模块初始化阶段，需要通过 module bind 注册成为 sender 或者 receiver 或者既是 sender 也是 receiver

- 如果是 sender 即需要提供一个 bind callback 回调函数

- 如果是 receiver 即需要提供 send callback 及 reset callback 两个回调函数

### 3.3.2 绑定

- 用户需要使用用户接口(MBI)来指定两业务模块间的绑定关系

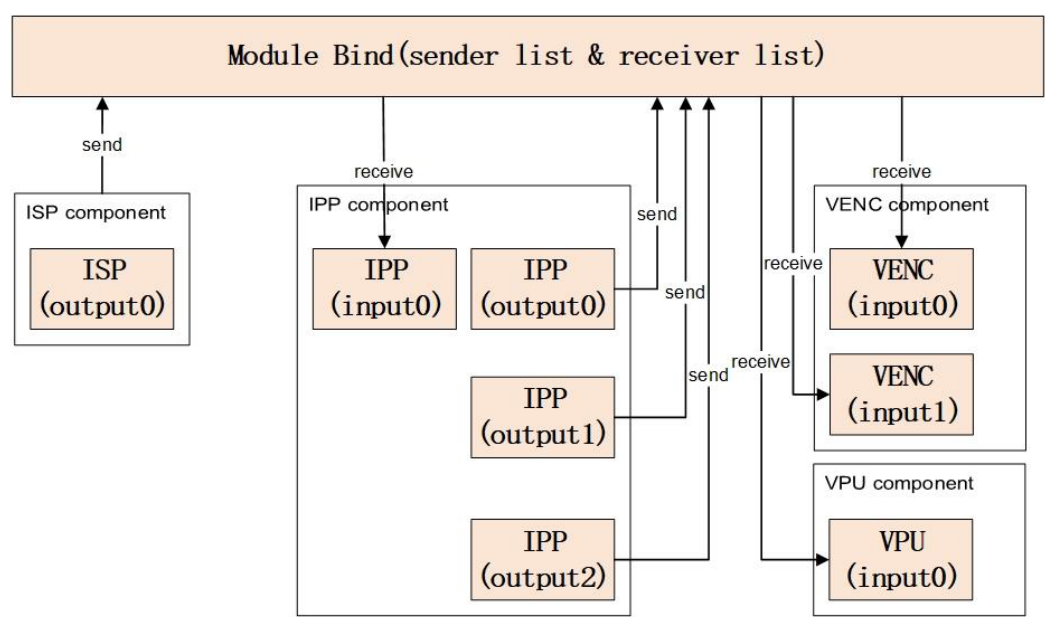
- 绑定的目标对像是两业模块的输入、输出接口 (instance,channel)

### 3.3.3 运行

- 如果某一业务模块的 sender 需要发送数据给它的 receiver ，那么需要调用 module register 提供 send\_data()，这样在链表中找到当初注册的回调函数，并执行

- 同上，如某一业务模块的 sender 需要复位它的 receiver，即需要调用 reset()

- 业务模块的 sender 是主动的，receiver 是被动的。需要注意一个 sender 的同一个 channel/instance 可以绑定多个 receiver channel，但同一个 receiver channel 只能绑定一个 sender channel。



## 3.4 内存管理模块

MBP整套框架都是基于其专有内存进行数据传输、存储的。OS内存可以用free命令查看，而pmz则使用cat /proc/media-mem可以查看到系统总共、已使用的、剩余的pmz，包括每一块已经使用的pmz的使用情况。

|  |
| --- |
| / # free  total used free shared buff/cache available  Mem: 240240 139552 61568 756 39120 91972  Swap: 0 0 0  / # cat /proc/mbp/pmz  +---PMZ: PHYS[0x08000000 - 0x0FFFFFFF], GFP=0, nBytes=131072KB, NAME="anonymous"  |-PMB: PHYS[0x08000000 - 0x0805FFFF], KVIRT=0xD0832000, FLAGS=0x00000001, nBytes=384KB, NAME="isp0-reg"  |-PMB: PHYS[0x08060000 - 0x08065FFF], KVIRT=0xD0893000, FLAGS=0x00000001, nBytes=24KB, NAME="venc job"  |-PMB: PHYS[0x08066000 - 0x08066FFF], KVIRT=0xD0AAB000, FLAGS=0x00000001, nBytes=4KB, AME="ge2d\_cmd\_buffer"  ...... |

MBP 提供了一套PMZ的API接口，包括PMZ申请，释放，物理地址和虚拟地址的映射等。

<[关于视频缓存池更详细介绍](https://confluence.amlogic.com/display/SW/MBP+Memory+Usage+Framework+Design?preview=/122798259/122805441/IPC_C3_SDK_MemoryUsageFramework.pdf)>

## 3.5 Audio Design

Audio 模块包括音频输入（Audio In）、音频输出（Audio Out）、音频编码（Audio encode）、音频解码（Audio decode）、声音质量增强（VQE）。

### 3.5.1音频输入输出API说明

<https://confluence.amlogic.com/display/SW/Audio+Design>

**相关API举例**

|  |
| --- |
| MBP\_S32 MBI\_AO\_SetPubAttr(AUDIO\_DEV AoDevId, AO\_ATTR\_S \*pstAttr )  MBP\_S32 MBI\_AO\_SetDecAttr (AUDIO\_DEV AiDevId, AO\_CHN AiChn, AO\_AdecConfig\_t \*pstAdecConfig);  /////////////////////////////////////////////  MBP\_S32 ret;  AO\_ATTR\_S stAttr;  AO\_AdecConfig\_t stAdecConfig;  AUDIO\_DEV AoDevId = 0; //such as, 0=TDM\_OUT, 1=ACODEC\_DAC, 2=Line out;  AO\_CHN AoChn = 0;    stAttr.eBitwidth = MBP\_AUDIO\_BIT\_WIDTH\_16;  stAttr.eSamplerate = MBP\_AUDIO\_SAMPLE\_RATE\_8000;  stAttr.eSoundmode = MBP\_AUDIO\_SOUND\_MODE\_MONO;  stAttr.eWorkmode = MBP\_AUDIO\_MODE\_I2S\_MASTER;  stAttr.u32PtNumPerFrm = 1024;  stAttr.u32ChnCnt = 1;    ret = MBI\_AO\_SetPubAttr(AoDevId, &stAttr); // 设置AI设备属性  ret = MBI\_AO\_GetPubAttr(AoDevId, &stGetAttr); // 获取AI设备属性  ret = MBI\_AO\_Enable(AoDevId); // 启用AO设备  ret = MBI\_AO\_EnableChn(AoDevId, AiChn ); //启用AO通道    memset(&stAdecConfig, 0x0, sizeof(AO\_AdecConfig\_t));  stAdecConfig.eAdecType = MBP\_AUDIO\_ADEC\_TYPE\_G711A;  stAoSetAdecConfig.stAdecG711Cfg.eSamplerate = MBP\_AUDIO\_SAMPLE\_RATE\_8000;  stAoSetAdecConfig.stAdecG711Cfg.eSoundmode = MBP\_AUDIO\_SOUND\_MODE\_MONO;    ret = MBI\_AO\_SetDecAttr (AoDevId, AoChn, &stAoSetAdecConfig); |

### 3.5.1 audio sample测试

|  |
| --- |
| # 修改sample\_audio.c  make sample-rebuild  找到编译出来的sample\_audio文件  cd c3\_buildroot\_refapp && find ./output/c3\_aw409\_refapp\_a32\_release/ -name "sample\_audio" | xargs ls -l  adb push sample\_audio /data  # 测试  / # chmod 777 /data/sample\_audio  # 录音（10秒）  / # sample\_audio read  # 播放录音  / # ls \*.pcm  dev0\_2ch\_48kHz\_16bit.pcm loopback\_lb.pcm  / # sample\_audio write dev0\_2ch\_48kHz\_16bit.pcm |

## 3.6 venc视频编码

VENC 模块由编码通道子模块（ VENC）和编码协议子模块（ H.264/H.265/JPEG/MJPEG）组成

通道接收到图像之后，比较图像尺寸和编码通道尺寸：

* 如果输入图像比编码通道尺寸大， VENC 将按照编码通道尺寸大小，调用 VGS 对源图像进行缩小，然后对缩小之后的图像进行编码。
* 如果输入图像比编码通道尺寸小， VENC 丢弃源图像。 VENC 不支持放大输入图像编码。
* 如果输入图像与编码通道尺寸相当， VENC 直接接受源图像，进行编码。

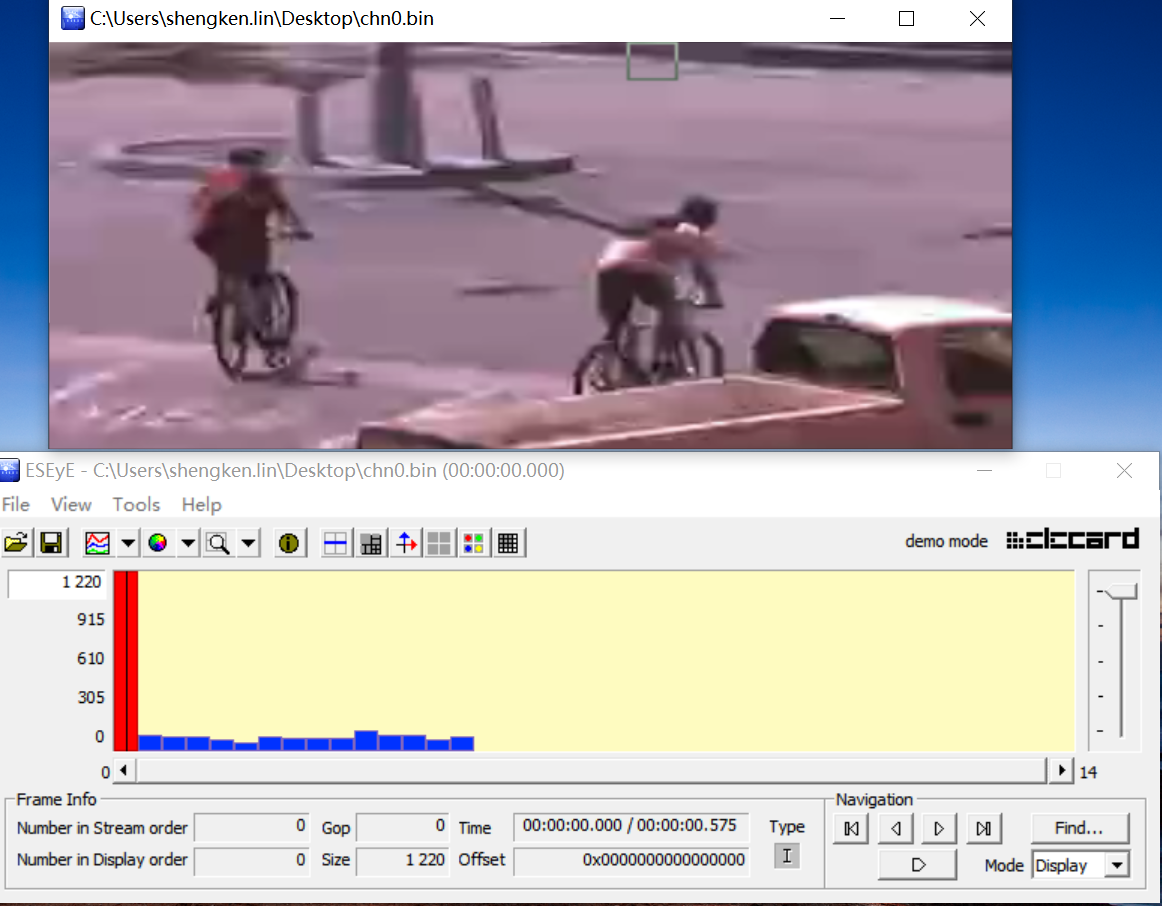
### 3.6.1 venc sample测试

将 YUV 数据编码成 H264

**注意，默认只支持图片格式为nv12的YUV文件**

|  |
| --- |
| adb push .\sample\_venc /data  adb push Z:\workspace\C3-file\yuv420p\_320x240.yuv /data  # width: 448  # height: 960  # encType: H.264: 96; H.265: 265; JPEG: 26;  # profile:  # fps: 24 30 60 # 帧率：  # gop: 25 # 1帧间隔  # rcMode:  # bitrate: 16000  /data/sample\_venc /data/encoder\_test\_video320x180\_nv12\_14.yuv 320 180 96 100 24 25 3 16000  # 默认输出路径为 **/mnt/chn0.bin** |

使用eseye\_u工具查看编码后的h.264文件



# 4. refApp使用

## 4.1 c3\_buildroot\_refapp编译与问题解决

### 4.1.1编译C3 buildroot

|  |
| --- |
| source setenv.sh  # 选择409  56. c3\_aw409\_a32\_release # 没有refapp  57. c3\_aw409\_refapp\_a32\_release |

### 4.1.2 修复无法使用adb问题

添加补丁： <https://scgit.amlogic.com/#/c/270669/>

|  |
| --- |
| cd /mnt/fileroot/shengken.lin/workspace/c3\_buildroot\_refapp/output/c3\_aw409\_refapp\_a32\_release/build/linux-amlogic-5.15-dev/common\_drivers  ln -s /mnt/fileroot/shengken.lin/workspace/c3\_buildroot\_refapp/.repo/projects/kernel/aml-5.15/common\_drivers.git .git  cd /mnt/fileroot/shengken.lin/workspace/c3\_buildroot\_refapp/kernel/aml-5.15/common\_drivers  git fetch ssh://shengken.lin@scgit.amlogic.com:29418/kernel/common\_drivers refs/changes/69/270669/6 && git cherry-pick FETCH\_HEAD  make linux-rebuild  make mbd-adla-rebuild  make mbd-audio-rebuild  make mbd-base-rebuild  make mbd-camera-rebuild  make mbd-cve-rebuild  make mbd-dewarp-rebuild  make mbd-ge2d-rebuild  make mbd-ppu-rebuild  make mbd-region-rebuild  make mbd-venc-rebuild  make mbd-vpu-rebuild  make pmz-rebuild  make |

### 4.1.3 解决网络问题

#### 4.1.3.1 网线直连

如果网线直连无法获取IP，尝试使用以下命令自动获取IP。

|  |
| --- |
| dhcpcd eth0 # 自动获取IP |

#### 4.1.3.2 配置wifi

|  |
| --- |
| wpa\_cli -iwlan0 remove\_network 0  wpa\_cli -iwlan0 add\_network 0  wpa\_cli -iwlan0 set\_network 0 ssid '"Amlogic-vpn04\_5G"'  wpa\_cli -iwlan0 set\_network 0 key\_mgmt WPA-PSK  wpa\_cli -iwlan0 set\_network 0 psk '"Aml1234566"'  wpa\_cli -iwlan0 set\_network 0 pairwise CCMP  wpa\_cli -iwlan0 set\_network 0 group CCMP  wpa\_cli -iwlan0 set\_network 0 proto RSN  wpa\_cli -iwlan0 enable\_network 0  wpa\_cli -iwlan0 status  wpa\_cli -iwlan0 save  dhcpcd wlan0 # 自动获取 wifi 网卡 Ip |

### 4.1.4 修复ipc\_alarm\_pld编译错误

|  |
| --- |
| c3\_aw409\_refapp\_a32\_release/build/ipc-reference-1.0/modules/alarm/src/ipc\_alarm\_pld.c:171:86: error: request for member ‘rect’ in something not a structure or union |

最新的C3-SDK暂时未修复该问题，可以通过checkout 到上一个commit解决。

|  |
| --- |
| # /mnt/fileroot/shengken.lin/workspace/c3\_buildroot\_refapp/vendor/amlogic/ipc/refapp/src  # 最新的是 commit id : af5a893738d9ba8372bee88fd141f4a7213e2763  git checkout 875e347a447e9bae8329392e7eee8e006d8b3c44  # /mnt/fileroot/shengken.lin/workspace/c3\_buildroot\_refapp/vendor/amlogic/ipc/mbp/prebuilt  # 最新的是 commit id : 87552832d73c94d9bf8d7739f4bff5d73c422f69  git checkout 29a9ddfcba947984e19e079c73e9a7f2f572a9cf  make mbi-rebuild  make ipc-reference-rebuild  make |

## 4.2 启动和停止refApp

|  |
| --- |
| /etc/init.d/S81ipc-refapp stop # 停止  /etc/init.d/S81ipc-refapp start # 启动  ifconfig  # inet addr:10.28.39.22 |

通过浏览器访问 **10.28.39.22**

账户：admin

密码：admin



**效果图**

