Melis RTOS 多媒体解码
开发指南

版本号: 1.0
发布日期: 2021.05.18

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.05.18	AWA1417	介绍 cedar 相关接口使用说明以及常用功能测试用例的使用说明;

目　录

1 概述

Melis 操作系统是由全志科技自主研发的一套精简、高效、易用的实时多任务操作系统。Cedar 中间件是基于 Melis 系统上开发的多媒体播放中间件，支持所有常见格式的音视频文件的解码和播放流程控制，为播放器应用程序提供统一的操作接口，使播放器的开发者能够把精力集中在用户操作体验、界面效果上，而不必关心繁杂多样的音视频文件格式、编码格式等。

1.1 编写目的

本文档主要讲述 Cedar 提供的对外 API，使开发者能迅速了解这些 API 的作用和调用规范，从而基于 Cedar 开发自己的多媒体播放器。

1.2 适用范围

本文档适用于 F 系列所有平台，如 F133、F1C800 等。

1.3 相关人员

Melis RTOS 下进行音视频模块开发的工程师，以及播放器开发和测试人员。

2 软件环境配置

在 melis 的根目录执行 make menuconfig 命令，选中 cedar 模块。

```
Modules Setup
    -->cedar module support
```

如下图所示：

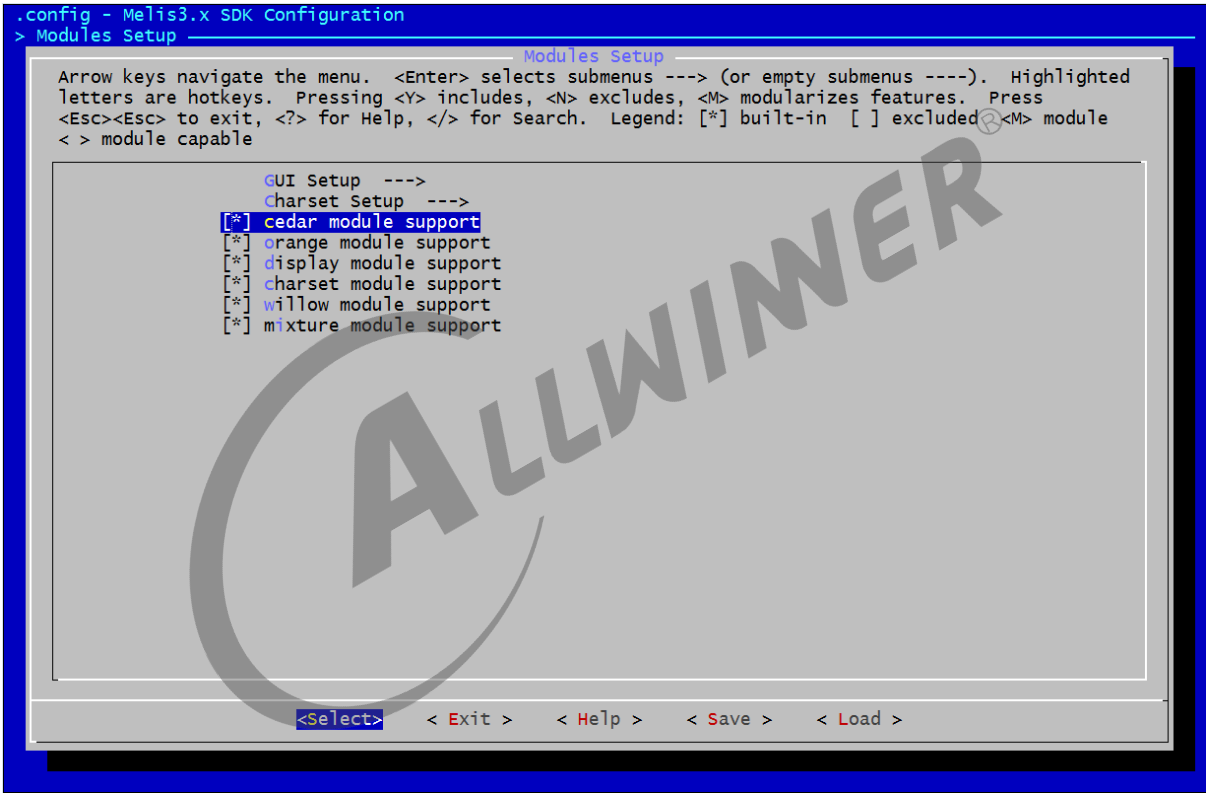

图 2-1: 配置 cedar 模块

3 CEDAR 模块

3.1 功能介绍

CEDAR 主要提供以下几个方面的操作接口：

1）音视频文件的播放流程控制，包括开始 (包含断点续播)、停止、暂停、快进快退、跳播、换音轨、变速播放、AB 播放。
2）播放过程中获取文件信息，例如得到播放总时间、当前时间、文件音视频编码格式和其他信息。
3）显示效果的操作，提供全屏、满屏、4:3、16:9、裁边等模式，另外提供自定义模式，让应用程序自行设置。
4）解码图像旋转功能，提供 90、180、270、水平镜像、垂直镜像等选项。
5）字幕处理，提供字幕的自动检测、解码、切换字幕等功能。
6）截图功能。
7）预览图获取和文件信息预取。
8）杂项。例如 callback 文件读取方式，用于读取加密文件；关闭外挂字幕检测，用于加快打开文件过程；与应用程序的通信管道连接。

3.2 接口函数说明

3.2.1 CEDAR_CMD_SET_MEDIAFILE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置要播放的文件信息，主要是路径和断点信息。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_MEDIAFILE；
  - aux = 0；
  - pbuffer = __cedar_media_file_inf*；
- 返回：设置结果
  - EPDK_OK：成功

- EPDK_FAIL：失败

- 示例：

```
//设置文件信息
esMODS_MIoctrl( robin_hced, CEDAR_CMD_SET_MEDIAFILE, 0, &(msg_p->file_info) );
```

3.2.2 CEDAR_CMD_GET_MESSAGE_CHN

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：返回一个消息通道，让应用程序和 cedar 能够通信。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_MESSAGE_CHNE；
  - aux = 0；
  - pbuffer = NULL；
- 返回：消息通道，类型为 __krnl_event_t *
- 示例：

```
__s32  get_feedback_msgQ( void )
{
    robin_cedar_msgQ = (g_Queue) esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_MESSAGE_CHN,
   0, NULL );
    if( robin_cedar_msgQ == NULL )
    {
        __err("Error in getting cedar error channel.\n");
        return -1;
    }

    return 0;
}
```

3.2.3 CEDAR_CMD_GET_ERROR_TYPE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取媒体播放器的错误类型，没有实现。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_ERROR_TYPE；
  - aux = 0；
  - pbuffer = NULL；
- 返回：得到错误类型。
- 示例：无

3.2.4 CEDAR_CMD_PLAY

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：启动播放，在调用这个接口之间，必须先 CEDAR_CMD_SET_MEDIAFILE，设置文件信息。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_PLAY；
  - aux = 0；
  - pbuffer = NULL；
- 返回：启动播放的结果
  - EPDK_OK：启动成功
  - EPDK_FAIL：启动失败
- 示例：

```
ret = esMODS_MIoctrl( robin_hced, CEDAR_CMD_PLAY, 0, NULL );
if( ret != EPDK_OK )
{
    __wrn("Fail in setting play cmd.\n");
    return SYN_OP_RET_SEND_CMD_ERR;
}
```

3.2.5 CEDAR_CMD_STOP

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：停止播放文件。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_STOP；
  - aux = 0；
  - pbuffer = NULL；
- 返回：停止播放的结果
  - EPDK_OK：停止成功
  - EPDK_FAIL：停止失败
- 示例：

```
ret = esMODS_MIoctrl( robin_hced, CEDAR_CMD_STOP, 0, NULL );
if( ret != EPDK_OK )
{
    __wrn("Fail in setting stop cmd.\n");
    return SYN_OP_RET_SEND_CMD_ERR;
}
```

3.2.6 CEDAR_CMD_PAUSE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：暂停播放，如果 cedar 当前已处于暂停状态，或处于正常播放状态，一般会成功；如果 cedar 处于快进快退状态，暂停无效。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_PAUSE；
  - aux = 0；
  - pbuffer = NULL；
- 返回：暂停的结果
  - EPDK_OK：暂停成功
  - EPDK_FAIL：暂停失败
- 示例：

```
ret = esMODS_MIoctrl( robin_hced, CEDAR_CMD_PAUSE, 0, NULL );
if( ret != EPDK_OK )
{
    __wrn("Fail in setting pause cmd.\n");
    return SYN_OP_RET_SEND_CMD_ERR;
}
```

3.2.7 CEDAR_CMD_FF

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：命令 cedar 进入快进状态。返回值为 EPDK_OK 时，只是表明 Cedar 接受了快进命令，但不一定能成功转入快进状态。所以还需要查 cedar 的状态以确认。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_FF；
  - aux = 0；
  - pbuffer = NULL；

- 返回：快进的结果
  - EPDK_OK：cedar 接受快进命令
  - EPDK_FAIL：cedar 拒绝快进命令

- 示例：

```
__s32 syn_op_ff( void )
{
    __s32  ret;
    __u8   err;
    __s32  msg;
    __u32  counter;

    /* clear cedar message Queue */
    g_QFlush( robin_cedar_msgQ );
    /* send ff command */
    ret = esMODS_MIoctrl( robin_hced, CEDAR_CMD_FF, 0, NULL );
    if( ret != EPDK_OK )
    {
        __wrn("Fail in setting ff cmd.\n");
        return SYN_OP_RET_SEND_CMD_ERR;
    }

    /* check cedar status until some message is gotten */
    counter = 0;
    while( 1 )
    {
        ret = esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_STATUS, 0, NULL );
        if( ret == CEDAR_STAT_FF )
        {
            return SYN_OP_RET_OK;
        }
        msg = (__s32)g_QAccept( robin_cedar_msgQ, &err );
        if( msg != NULL )
        {
            /* feedback msg */
            if( msg == CEDAR_FEDBAK_NO_ERROR )
            {
                return SYN_OP_RET_OK;
            }
            else
            {
                g_QFlush( robin_feedbackQ );
                g_QPost( robin_feedbackQ, (void *)msg );        // only feed back error
   message
                return SYN_OP_RET_CEDAR_FEEDBACK_ERR;
            }
        }
        if( to_quit_robin )
            return SYN_OP_RET_TO_QUIT_ROBIN;

        if( ++counter >= OVERTIME_THRESHOLD )
            return SYN_OP_RET_OVERTIME;

        g_delay( 5 );
    }
}
```

3.2.8 CEDAR_CMD_REV

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：命令 cedar 进入快退状态。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_REV；
  - aux = 0；
  - pbuffer = NULL；
- 返回：快退的结果
  - EPDK_OK：cedar 接受快退命令
  - EPDK_FAIL：cedar 拒绝快退命令
- 示例：同 CEDAR_CMD_FF

3.2.9 CEDAR_CMD_JUMP

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：跳播命令被接受后，cedar 内部的线程会正式执行跳播的相关操作，有可能执行过程中会失败。所以即使 cedar 接受了跳播命令，也不一定能成功跳播到目的时间点。有可能会引发异常而终止播放。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_JUMP；
  - aux = nPts；跳播的目的时间，单位毫秒。
  - pbuffer = 精确跳播的标记；1 表示精确跳播，0 表示可以有误差。精确跳播用于恒定码率的 mp3 文件的跳播，复读机等产品的音频文件一般是 CBR 的 mp3，对跳播精确度要求非常高，故特别加此参数。
- 返回：跳播命令是否被接受
  - EPDK_OK：cedar 接受跳播命令
  - EPDK_FAIL：cedar 拒绝跳播命令
- 示例：

```
__s32 syn_op_jump( __u32 time )
{
    __s32  ret;
    __u8   err;
    __s32  msg;
    __u32  counter;
```

```
    /* clear cedar message Queue */
    g_QFlush( robin_cedar_msgQ );
    /* send play command */
    ret = esMODS_MIoctrl( robin_hced, CEDAR_CMD_JUMP, time, NULL );
    if( ret != EPDK_OK )
    {
        __wrn("Fail in setting jump cmd.\n");
        return SYN_OP_RET_SEND_CMD_ERR;
    }

    /* check cedar status until some message is gotten */
    counter = 0;
    while( 1 )
    {
        ret = esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_STATUS, 0, NULL );
        if( ret == CEDAR_STAT_PLAY || ret == CEDAR_STAT_STOP )
        {
            return SYN_OP_RET_OK;
        }
        msg = (__s32)g_QAccept( robin_cedar_msgQ, &err );
        if( msg != NULL )
        {
            /* feedback msg */
            if( msg == CEDAR_FEDBAK_NO_ERROR )
            {
                return SYN_OP_RET_OK;
            }
            else
            {
                g_QFlush( robin_feedbackQ );
                g_QPost( robin_feedbackQ, (void *)msg ); // only feed back error
message
                return SYN_OP_RET_CEDAR_FEEDBACK_ERR;
            }
        }
        if( to_quit_robin )
            return SYN_OP_RET_TO_QUIT_ROBIN;

        if( ++counter >= OVERTIME_THRESHOLD )
            return SYN_OP_RET_OVERTIME;

        g_delay( 5 );
    }

    return SYN_OP_RET_OK;
}
```

3.2.10 CEDAR_CMD_GET_STATUS

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：CEDAR_STAT_JUMP 表示当前处于跳播处理状态。另外，如果当前 cedar 正处于状态转换期，那么返回值是 stat|0x80，以表示是临时状态。如果参数 aux = 0，那么在这个接口内部会循环查询，直到状态平稳，再返回。Aux=1，就立即返回。
- 参数：

- mp = cedar 模块的句柄；
- cmd = CEDAR_CMD_GET_STATUS；
- aux = 否等到平稳状态再返回。1: 表示立即返回 cedar 的状态，0：表示等 cedar 的状态平稳之后再返回。因为 cedar 的状态如果异常，可能一直不会恢复到平稳状态，所以 aux=0 有可能引起应用程序线程的死锁。
- pbuffer = NULL；

- 返回：__cedar_status_t：现在定义了 6 种状态：

  - CEDAR_STAT_PLAY=0,
  - CEDAR_STAT_PAUSE,
  - CEDAR_STAT_STOP,
  - CEDAR_STAT_FF,
  - CEDAR_STAT_RR,
  - CEDAR_STAT_JUMP,

- 示例：

```
ret = esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_STATUS, 0, NULL );
if( ret == CEDAR_STAT_PLAY )
{
    return SYN_OP_RET_OK;
}
```

3.2.11 CEDAR_CMD_AUDIO_RAW_DATA_ENABLE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：命令 cedar 以 rawdata 输出音频。必须在播放前设置，播放过程中设置无效。Cedar 并不是所有的音频编码格式都以 rawdata 输出，目前 ac3 和 dts 是会以 rawdata 输出，其他格式仍然是解码后再输出。
- 参数：

  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_AUDIO_RAW_DATA_ENABLE；
  - aux = 0,1；音频是否以 rawdata 输出
  - pbuffer = NULL；

- 返回：设置音频 rawdata 输出的结果

  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败

- 示例：

```
__s32 robin_enable_raw_data( __bool flag )
{
    __u8   err;
    __s32  ret;

    g_pend_mutex( robin_cedar_mutex, &err );

    ret = esMODS_MIoctrl( robin_hced, CEDAR_CMD_AUDIO_RAW_DATA_ENABLE, flag, NULL );

    g_post_mutex( robin_cedar_mutex );

    return ret;
}
```

3.2.12 CEDAR_CMD_GET_TOTAL_TIME

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：查询文件的总时间。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_TOTAL_TIME；
  - aux = 0；
  - pbuffer = NULL；
- 返回：文件的总时间：单位毫秒。
- 示例：

```
__u32  robin_get_total_time( void )
{
    __u32  time;

    if( get_msg_nr_of_q( robin_cmdQ ) != 0 )
        return 0;

    time = (__u32)esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_TOTAL_TIME, 0, NULL );

    return time;
}
```

3.2.13 CEDAR_CMD_GET_CUR_TIME

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：得到当前的播放时间。
- 参数：
  - mp = cedar 模块的句柄；

- cmd = CEDAR_CMD_GET_CUR_TIME;
- aux = 0;
- pbuffer = NULL;

- 返回：当前的播放时间：单位毫秒
- 示例：

```
__u32  robin_get_cur_time( void )
{
    return (__u32)esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_CUR_TIME, 0, NULL );
}
```

3.2.14 CEDAR_CMD_GET_TAG

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取当前播放时间点的标签信息，app 可以将其作为断点信息保存下来，下次再播放该文件时，将其传入 cedar，cedar 就可以从 tag 所标识的时间点开始播放。
- 参数：
  - mp = cedar 模块的句柄;
  - cmd = CEDAR_CMD_GET_TAG;
  - aux = 0;
  - pbuffer = __cedar_tag_inf_t*，APP 分配内存;
- 返回：是否获取成功
  - EPDK_OK：获取成功
  - EPDK_FAIL：获取失败
- 示例：

```
__s32  robin_get_tag( __cedar_tag_inf_t  *tag_info_p )
{
    if( tag_info_p == NULL )
        return -1;

    if( esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_TAG, 0, tag_info_p ) != EPDK_OK )
        return -1;

    return 0;
}
```

3.2.15 CEDAR_CMD_SET_FRSPEED

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`

- 功能描述：设置快进快退的速度。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_FRSPEED；
  - aux = 快进快退的速度，现在规定在 1~128 倍速之间；
  - pbuffer = NULL；
- 返回：是否设置成功
  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败
- 示例：

```
__s32  robin_set_ff_rr_speed( __u32 ff_rr_speed )
{
    if( esMODS_MIoctrl( robin_hced, CEDAR_CMD_SET_FRSPEED, ff_rr_speed, NULL ) ==
  EPDK_OK )
        return 0;
    else
        return -1;
}
```

3.2.16 CEDAR_CMD_GET_FRSPEED

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取当前设置的快进快退的速度值。如果返回值为 0，表示没有设置。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_FRSPEED；
  - aux = 0；
  - pbuffer = NULL；
- 返回：获取当前的快进快退速度
  - 1~128：倍速
  - 0：获取失败
- 示例：

```
__u32  robin_get_ff_rr_speed( void )
{
    return esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_FRSPEED, 0, NULL );
}
```

3.2.17 CEDAR_CMD_SET_TAG

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：将断点信息设入 cedar，之后调用 CEDAR_CMD_PLAY 时，即从断点信息标识的地方开始播放，实现断点续播功能。一般在接口 CEDAR_CMD_SET_MEDIAFILE 的参数 __cedar_media_file_inf 中附带了断点信息 __cedar_tag_inf_t*，可以在调用 CEDAR_CMD_SET_MEDIAFILE 时顺带输入断点信息。但也可以单独输入，如果想单独输入，CEDAR_CMD_SET_TAG 必须在 CEDAR_CMD_SET_MEDIAFILE 之后调用才有效。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_TAG；
  - aux = 0；
  - pbuffer = __cedar_tag_inf_t*，APP 分配内存；
- 返回：设置断点信息是否成功
  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败
- 示例：无

3.2.18 CEDAR_CMD_GET_ABSTYPE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取当前正在播放的音频流的编码类型。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_ABSTYPE；
  - aux = 0；
  - pbuffer = NULL；
- 返回：音频编码类型，__cedar_audio_fmt_t
- 示例：

```
__cedar_audio_fmt_t  robin_get_audio_encoding( void )
{
    __cedar_audio_fmt_t  ret;

    robin_wait_no_file( );

    ret = (__cedar_audio_fmt_t)esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_ABSTYPE, 0,
  NULL );

    g_post_mutex( robin_cedar_mutex );
```

```
    return ret;
}
```

3.2.19 CEDAR_CMD_GET_AUDBPS

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取音频流的平均码率。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CCEDAR_CMD_GET_AUDBPS；
  - aux = 0；
  - pbuffer = NULL；
- 返回：码率，单位是 Bit/s
- 示例：

```
__u32  robin_get_audio_bps( void )
{
    __u32  ret;
    __u8   err;

    g_pend_mutex( robin_cedar_mutex, &err );
    robin_wait_no_file( );

    ret = (__u32)esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_AUDBPS, 0, NULL );

    g_post_mutex( robin_cedar_mutex );
    return ret;
}
```

3.2.20 CEDAR_CMD_GET_SAMPRATE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取当前播放的音频流的采样率。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_SAMPRATE；
  - aux = 0；
  - pbuffer = NULL；
- 返回：采样率：sample/s
- 示例：

```
__u32  robin_get_audio_sample_rate( void )
{
    return (__u32)esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_SAMPRATE, 0, NULL );
}
```

3.2.21 CEDAR_CMD_SET_CHN

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置声道类型，即把左声道的声音作为双声道输出，或者右声道。目前该功能由应用程序设置，Cedar 不实现该功能。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_CHN；
  - aux = 0；
  - pbuffer = NULL；
- 返回：是否设置成功
  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败
- 示例：无

3.2.22 CEDAR_CMD_GET_CHN

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取当前播放的音频流的声道数。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_CHN；
  - aux = 0；
  - pbuffer = NULL；
- 返回：声道数
- 示例：无

3.2.23 CEDAR_CMD_SET_VOL

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置当前播放的音频流的音量。
- 参数：

  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_VOL；
  - aux = 0~30，音量值；
  - pbuffer = NULL；
- 返回：设置的音量值。
- 示例：

```
esMODS_MIoctrl(robin_hce, CEDAR_CMD_SET_VOL, AUDIO_DEV_VOLUME, NULL);
```

3.2.24 CEDAR_CMD_GET_VOL

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取当前播放的音频流的音量。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_VOL；
  - aux = 0；
  - pbuffer = NULL；
- 返回：当前的音量值。
- 示例：

```
volume = esMODS_MIoctrl(robin_hce, CEDAR_CMD_GET_VOL, 0, NULL);
```

3.2.25 CEDAR_CMD_VOLUP

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：音量值上调一档。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_VOLUP；
  - aux = 0；
  - pbuffer = NULL；
- 返回：上调一档以后的音量值
- 示例：无

3.2.26 CEDAR_CMD_VOLDOWN

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：音量值下调一档。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_VOLDOWN；
  - aux = 0；
  - pbuffer = NULL；
- 返回：下调一档以后的音量值
- 示例：无

3.2.27 CEDAR_CMD_SET_EQ

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置音效模式，音效模式的值为 __cedar_audio_eq_t。其中有用户自定义的音效类型，如果是用户自定义，那么 pbuffer 用来传递这个音效的各频段的参数值。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_EQ；
  - aux = __cedar_audio_eq_t；
  - pbuffer = __s8 * UsrEq, 实际类型是 __s8 UsrEq[USR_EQ_BAND_CNT]；
- 返回：用户设置以后的 EQ 值，__cedar_audio_eq_t
- 示例：

```
__s32  robin_set_eq_mode( __cedar_audio_eq_t  eq_mode )
{
    if( esMODS_MIoctrl( robin_hced, CEDAR_CMD_SET_EQ, eq_mode, NULL ) == EPDK_OK )
        return 0;
    else
        return -1;
}

__s32 robin_set_eq_value( const __s8 *eq_value_list )
{
    if( eq_value_list == NULL )
        return -1;

    if( esMODS_MIoctrl( robin_hced, CEDAR_CMD_SET_EQ, CEDAR_AUD_EQ_TYPE_USR_MODE, (void
   *)eq_value_list ) == EPDK_OK )
        return 0;
    else
        return -1;
}
```

3.2.28 CEDAR_CMD_GET_EQ

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：得到当前设定的音效模式。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_EQ；
  - aux = 0；
  - pbuffer = NULL；
- 返回：音效模式，__cedar_audio_eq_t
- 示例：

```
__cedar_audio_eq_t  robin_get_eq_mode( void )
{
    return  (__cedar_audio_eq_t)esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_EQ, 0, NULL )
  ;
}
```

3.2.29 CEDAR_CMD_SET_VPS

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置变速值。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_VPS；
  - aux = 变速范围，-4~10，关于变速的计算，变速后的播放速度 v = 1 * (1 + aux/10)。所以变速后的播放速度的范围是：0.6 倍速 ~2 倍速之间；
  - pbuffer = NULL；
- 返回：设置之后的变速值，正常情况下和 aux 应该相等。
- 示例：

```
__s32  robin_set_play_speed( __s32 play_speed )
{
    if( esMODS_MIoctrl( robin_hced, CEDAR_CMD_SET_VPS, play_speed, NULL ) == EPDK_OK )
        return 0;
    else
        return -1;
}
```

3.2.30 CEDAR_CMD_GET_VPS

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取当前的变速值。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_VPS；
  - aux = 0；
  - pbuffer = NULL；
- 返回：当前的变速值。
- 示例：

```
__s32  robin_get_play_speed( void )
{
    return esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_VPS, 0, NULL );
}
```

3.2.31 CEDAR_CMD_SET_AB_A

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：这是 ab 播放的第一版接口，用于设置 a 点，第一版 ab 播放只支持音频文件，并且只能以调用该接口时的播放时间作为 a 点，所以不设置 a 点的时间参数。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_AB_A；
  - aux = 0；
  - pbuffer = NULL；
- 返回：是否设置成功
  - EPDK_OK：设置 ab 播放的 a 点成功
  - EPDK_FAIL：设置失败
- 示例：

```
__s32  robin_set_ab_a( void )
{
    /*-----------------===== to be refreshed =====-----------------------*/
    if( esMODS_MIoctrl( robin_hced, CEDAR_CMD_SET_AB_A, 0, NULL ) == EPDK_OK )
        return 0;
    else
        return -1;
}
```

3.2.32 CEDAR_CMD_SET_AB_B

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：这是 ab 播放的第一版接口，用于设置 b 点，第一版 ab 播放只支持音频文件，并且只能以调用该接口时的播放时间作为 b 点。设置成功后，立即启动 ab 播放，即跳转到 a 点开始播放。所以，调用该接口前，必须先调用 CEDAR_CMD_SET_AB_LOOPCNT 设置好循环次数。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_AB_B；
  - aux = 0；
  - pbuffer = NULL；
- 返回：是否设置成功
  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败
- 示例：

```
__s32  robin_set_ab_b( void )
{
    /*--------------------===== to be refreshed =====-----------------------*/
    if( esMODS_MIoctrl( robin_hced, CEDAR_CMD_SET_AB_B, 0, NULL ) == EPDK_OK )
        return 0;
    else
        return -1;
}
```

3.2.33 CEDAR_CMD_SET_AB_LOOPCNT

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置 ab 播放的循环次数，该接口只用于 ab 播放第一版，必须在调用 CEDAR_CMD_SET_AB_B 之前设置。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_AB_LOOPCNT；
  - aux = oopcnt，希望的 ab 循环次数；
  - pbuffer = NULL；
- 返回：是否设置成功

- EPDK_OK：设置成功
- EPDK_FAIL：设置失败

- 示例：

```
__s32  robin_set_ab_loop_count( __u32 count )
{
    /*-----------------===== to be refreshed =====----------------------*/
    if( esMODS_MIoctrl( robin_hced, CEDAR_CMD_SET_AB_LOOPCNT, count, NULL ) == EPDK_OK
  )
        return 0;
    else
        return -1;
}
```

3.2.34 CEDAR_CMD_CLEAR_AB

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：清除 ab 播放，清除以后，视频文件播放到 B 点时，不会在跳转回 A 点，而是继续播放。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_CLEAR_AB；
  - aux = 0；
  - pbuffer = NULL；
- 返回：
  - EPDK_OK：清除 AB 播放成功
  - EPDK_FAIL：清除 AB 播放失败
- 示例：

```
__s32  robin_cancle_ab( void )
{
    /*-----------------===== to be refreshed =====----------------------*/
    if( esMODS_MIoctrl( robin_hced, CEDAR_CMD_CLEAR_AB, 0, NULL ) == EPDK_OK )
        return 0;
    else
        return -1;
}
```

3.2.35 CEDAR_CMD_SET_SPECTRUM

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`

- 功能描述：如果打开频谱解析，那么在做音频渲染时，音频渲染库就会同时计算频谱参数。一般情况下，只有播放音频文件时，才需要打开频谱解析。所以，如果是视频文件，一般返回值是 EPDK_FAIL。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_SPECTRUM；
  - aux = 1 或 0，1 表示打开频谱解析，0 表示关闭频谱解析；
  - pbuffer = NULL；
- 返回：打开频谱是否成功
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：

```
void  robin_enable_spectrum( void )
{
    esMODS_MIoctrl( robin_hced, CEDAR_CMD_SET_SPECTRUM, 1, NULL );
}
```

3.2.36 CEDAR_CMD_GET_SPECTRUM

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取一组频谱，频谱总共有 1024 行，每行 32 列。这个接口是获取其中一行的 32 列。Cedar 内部会根据当前的音频播放时间，选取合适的一行，把 32 个列的值给到 app 设下来的 buffer 里。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_SPECTRUM；
  - aux = 0；
  - pbuffer = __u16*；实际类型为 __u16 uSpectValColumn [SPECTRUM_COLUMN_SIZE]，app 必须分配这么大的内存，然后把指针设给 cedar。
- 返回：是否获取到频谱
  - EPDK_OK：获取成功
  - EPDK_FAIL：获取失败
- 示例：

```
__s32 __dsk_wkm_get_spectrum_info( __u16 *value_list )
{
    __s32  ret;
```

```
    if( value_list == NULL )
    {
        return -1;
    }

    ret = esMODS_MIoctrl( dsk_wkm_hced, CEDAR_CMD_GET_SPECTRUM, 0, value_list );
    if( ret == EPDK_OK )
    {
        return 0;
    }
    else                                    // no instant spectrum info
    {
        return -1;
    }
}
```

3.2.37 CEDAR_CMD_SEL_AUDSTREAM

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：通知 cedar 换音轨，aux 为目标音轨在音轨数组里的下标号。Cedar 内部会判断是否能够换音轨。如果可以，那么返回 EPDK_OK。然后把消息发到自己的消息队列里，cedar 的内部线程开始进行换音轨的操作。之所以这样做，是因为换音轨的操作比较复杂，涉及到众多 cedar 的内部模块，容易出现异常，所以不想放在 app 的线程里去做，避免影响 app 的运行。所以，返回值为 EPDK_OK，并不代表换音轨完成，只是表明 cedar 接受了换音轨操作。判断换音轨结束，是通过调用 CEDAR_CMD_GET_STATUS 来完成的，在换音轨过程中，cedar 的状态处于临时态，当换音轨操作结束，才会恢复到平稳态。但是是否换音轨成功，还要再通过 CEDAR_CMD_GET_AUDSTREAM 的接口得到当前正在播放的音轨号，和 app 想换到的音轨号进行比较，如果相同才表示换音轨成功。在这里，音轨和音频流意义完全相同。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SEL_AUDSTREAM；
  - aux = 音频流数组的下标号；
  - pbuffer = NULL；
- 返回：Cedar 是否接受换音轨的命令
  - EPDK_OK：接受
  - EPDK_FAIL：拒绝
- 示例：

```
__s32  robin_select_track( __u32 track_index )
{
    __u8   err;

    g_pend_mutex( robin_cedar_mutex, &err );
    robin_wait_no_file( );
```

```
    if( esMODS_MIoctrl( robin_hced, CEDAR_CMD_SEL_AUDSTREAM, track_index, NULL ) ==
EPDK_OK )
    {
        g_post_mutex( robin_cedar_mutex );
        return 0;
    }
    else
    {
        g_post_mutex( robin_cedar_mutex );
        return -1;
    }
}
```

3.2.38 CEDAR_CMD_GET_AUDSTREAM

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取当前正在播放的音频流的下标号，下标号一般从 0 开始。音频流数组也是 cedar 给 app 的。音频流的下标号就是在音频流数组里的下标号。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_AUDSTREAM；
  - aux = 0；
  - pbuffer = NULL；
- 返回：当前正在播放的音频流的下标号：从 0 开始。-1 表示异常。
- 示例：

```
__s32  robin_get_track_index( void )
{
    __s32  ret;
    __u8   err;

    g_pend_mutex( robin_cedar_mutex, &err );
    robin_wait_no_file( );

    ret = (__s32)esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_AUDSTREAM, 0, NULL );

    g_post_mutex( robin_cedar_mutex );
    return ret;
}
```

3.2.39 CEDAR_CMD_GET_AUDSTREAM_PROFILE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`

- 功能描述：获取视频文件的音频流信息，这是第一版的接口，在数据结构 __audstream_profile_t 里规定了音频流数组的最多元素为 4 个。所以如果视频文件包含了超过 4 个音频流，就不能全部告诉应用程序了。为克服这个缺点，又设计了第二版接口。两版接口都可以使用，但不能混用。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_AUDSTREAM_PROFILE；
  - aux = 0；
  - pbuffer = __audstream_profile_t*, app 分配内存；
- 返回：是否成功获取文件的音频流信息
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：

```
__s32  robin_get_track_info( __audstream_profile_t  *track_info_p )
{
    __u8   err;

    if( track_info_p == NULL )
        return -1;

    g_pend_mutex( robin_cedar_mutex, &err );
    robin_wait_no_file( );

    if( esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_AUDSTREAM_PROFILE, 0, track_info_p )
  == EPDK_OK )
    {
        g_post_mutex( robin_cedar_mutex );
        return 0;
    }
    else
    {
        g_post_mutex( robin_cedar_mutex );
        return -1;
    }
}
```

3.2.40 CEDAR_CMD_GET_AUDSTREAM_CNT

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取当前播放的视频文件的音轨数。音频文件一般不调用该接口，如果调用，返回值为-1。
- 参数：
  - mp = cedar 模块的句柄；

- cmd = CEDAR_CMD_GET_AUDSTREAM_CNT;
- aux = 0;
- pbuffer = NULL;

- 返回：音轨数，-1 表示异常
- 示例：无

3.2.41 CEDAR_CMD_GET_AUDSTREAM_PROFILE_V2

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：数据结构 __audstream_profile_v2_t 内部含有一个指针 __audio_bs_info_t*，由 app 分配内存，分配足够数量的元素，所以一般情况下，app 应该先调用 CEDAR_CMD_GET_AUDSTREAM_CNT 得到音频流的数量，然后分配内存，再调用本接口。
- 参数：
  - mp = cedar 模块的句柄;
  - cmd = CEDAR_CMD_GET_AUDSTREAM_PROFILE_V2;
  - aux = 0;
  - pbuffer = (__audstream_profile_v2_t*);
- 返回：
  - EPDK_OK：获取成功
  - EPDK_FAIL：获取失败
- 示例：

```
//得到音频流信息
result = esMODS_MIoctrl(pCedar, CEDAR_CMD_GET_AUDSTREAM_CNT, 0, NULL);
WARNING("file audio stream count is [%d]\n", result);
pTestApp->AProf.nAudStrmMaxCnt = (__u8)result;
pTestApp->AProf.nAudStrmNum = 0;
if(pTestApp->AProf.AudStrmListArray)
{
    free(pTestApp->AProf.AudStrmListArray);
}
pTestApp->AProf.AudStrmListArray = (__audio_bs_info_t*)malloc(sizeof(__audio_bs_info_t)
  *pTestApp->AProf.nAudStrmMaxCnt);
if(pTestApp->AProf.nAudStrmMaxCnt == NULL)
{
    WARNING("malloc fail\n");
    goto _exit_test_app;
}
memset(pTestApp->AProf.AudStrmListArray, 0, sizeof(__audio_bs_info_t)*pTestApp->AProf.
  nAudStrmMaxCnt);
result = esMODS_MIoctrl(pCedar, CEDAR_CMD_GET_AUDSTREAM_PROFILE_V2, 0, &pTestApp->AProf
  );
```

3.2.42 CEDAR_CMD_QUERY_BUFFER_USAGE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：查询当前的视频解码驱动的 vbs buffer 的使用率, 和音频解码驱动的 abs buffer 的使用率。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_QUERY_BUFFER_USAGE；
  - aux = 0；
  - pbuffer = __buffer_usage_t*, app 分配内存；
- 返回：
  - EPDK_OK：查询成功
  - EPDK_FAIL：查询失败
- 示例：无

3.2.43 CEDAR_CMD_SET_AB_A_V2

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：这是 ab 播放第二版的接口，用于设置 a 点，aux 用来指定 a 点的 pts, 如果想用当前的播放时间作为 a 点，aux=-1 即可。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_AB_A_V2；
  - aux = a 点的时间点, 单位 ms, -1 表示取当前的播放时间作为 a 点的时间点；
  - pbuffer = NULL；
- 返回：是否设置成功
  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败
- 示例：无

3.2.44 CEDAR_CMD_SET_AB_B_V2

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：这是 ab 播放第二版的接口，用于设置 b 点，aux 用来指定 b 点的 pts, 如果想用当前的播放时间作为 b 点，aux=-1 即可。注意与第一版不同的是，设置完 b 点后，ab 播放不会生效，要使 ab 播放生效，还要调用 CEDAR_CMD_ENABLE_AB_V2 接口才可以。

- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_AB_B_V2；
  - aux = b 点的时间点，-1 表示取当前的播放时间作为 b 点时间点；
  - pbuffer = NULL；
- 返回：是否设置成功
  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败
- 示例：无

3.2.45 CEDAR_CMD_SET_AB_LOOPCNT_V2

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：这是 ab 播放第二版的接口，用于设置 ab 播放时的循环次数。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_AB_LOOPCNT_V2；
  - aux = ab 播放的循环次数；
  - pbuffer = NULL；
- 返回：设置之后的 ab 播放的循环次数，一般就等于 aux，除非设置失败。
- 示例：无

3.2.46 CEDAR_CMD_CLEAR_AB_V2

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：这是 ab 播放第二版的接口，用于清除 ab 播放。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_CLEAR_AB_V2；
  - aux = 清除 ab 播放；
  - pbuffer = NULL；
- 返回：是否清除成功
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：无

3.2.47 CEDAR_CMD_ENABLE_AB_V2

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：启动 ab 播放。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_ENABLE_AB_V2；
  - aux = 0；
  - pbuffer = NULL；
- 返回：启动 ab 播放
  - EPDK_OK：启动成功
  - EPDK_FAIL：启动失败
- 示例：无

3.2.48 CEDAR_CMD_SET_AUDIO_AB_MODE_V2

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：第一版 ab 播放，只支持音频文件，其实现方式是音频解码库记录设置 a 点时读到的文件位置，和设置 b 点时读到的文件位置。如果当前读到的位置超过 b 点，那么就通过 fseek 回到 a 点继续播放。第二版 ab 播放，因为要支持视频文件，所以不能采用上述方法，故采用的是记录时间点，通过跳播回到 a 点时间点的办法实现 ab 播放。在播放音频文件时，第二版接口默认采用跳播方式。但如果应用程序要求采用 fseek 方式，那么就调用该接口，cedar 就会使用 fseek 方式对音频文件进行 ab 播放，不推荐。希望统一采用跳播方式。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_AUDIO_AB_MODE_V2；
  - aux = 音频文件 ab 播放时，回到 a 点的方式。1：用 fseek 操作回到 a 点；0：用跳播方式跳到 a 点；
  - pbuffer = NULL；
- 返回：
  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败
- 示例：无

3.2.49 CEDAR_CMD_GET_VBSTYPE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`

- 功能描述：获取当前正在播放的视频流的编码格式。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_VBSTYPE；
  - aux = 0；
  - pbuffer = NULL；
- 返回：视频编码类型，__cedar_video_fmt_t。
- 示例：

```
__cedar_video_fmt_t  robin_get_video_encoding( void )
{
    __cedar_video_fmt_t  ret;
    __u8   err;

    g_pend_mutex( robin_cedar_mutex, &err );
    robin_wait_no_file( );

    ret = (__cedar_video_fmt_t)esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_VBSTYPE, 0,
  NULL );

    g_post_mutex( robin_cedar_mutex );
    return ret;
}
```

3.2.50 CEDAR_CMD_GET_VIDBITRATE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取当前正在播放的视频流的比特率。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_VIDBITRATE；
  - aux = 0；
  - pbuffer = NULL；
- 返回：比特率。
- 示例：

```
__u32  robin_get_video_bps( void )
{
    __u32  ret;
    __u8   err;

    g_pend_mutex( robin_cedar_mutex, &err );
    robin_wait_no_file( );

    ret = (__u32)esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_VIDBITRATE, 0, NULL );
```

```
    g_post_mutex( robin_cedar_mutex );
    return ret;
}
```

3.2.51 CEDAR_CMD_GET_VIDFPS

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取当前正在播放的视频流的帧率。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_VIDFPS；
  - aux = 0；
  - pbuffer = NULL；
- 返回：帧率，数值放大 1000 倍，即 25 帧每秒，返回值是 25000。
- 示例：

```
__u32  robin_get_video_frame_rate( void )
{
    __u32  ret;
    __u8   err;

    g_pend_mutex( robin_cedar_mutex, &err );
    robin_wait_no_file( );

    ret = (__u32)esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_VIDFPS, 0, NULL );

    g_post_mutex( robin_cedar_mutex );
    return ret;
}
```

3.2.52 CEDAR_CMD_GET_FRAMESIZE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取当前正在播放的视频流的帧的宽高，宽高合在一个 __s32 的数中给出。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_FRAMESIZE；
  - aux = 0；
  - pbuffer = NULL；
- 返回：数值 __s32 的意义为 (width << 16) + height；单位为像素。

- 示例：

```
__s32  robin_get_video_frame_size( __u32 *width_p, __u32 *height_p )
{
    __u32  size;
    __u8   err;

    if( width_p == NULL || height_p == NULL )
        return -1;

    g_pend_mutex( robin_cedar_mutex, &err );
    robin_wait_no_file( );

    size = (__u32)esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_FRAMESIZE, 0, NULL );
    *width_p  = ( size & ( 0xFFFFU<<16) ) >> 16;
    *height_p = size & 0xFFFF;

    g_post_mutex( robin_cedar_mutex );
    return 0;
}
```

3.2.53 CEDAR_CMD_SET_VID_LAYERHDL

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置图层句柄给 cedar。Cedar 自己不会主动申请图层句柄，图层句柄由 app 申请，并传给 cedar。Cedar 得到句柄后，会自己设置图层属性，一般 scaler 属性一定会设置。
- 参数：
  - mp = cedar 模块的句柄;
  - cmd = CEDAR_CMD_SET_VID_LAYERHDL;
  - aux = 0;
  - pbuffer = vid_layer_hdl;
- 返回：
  - EPDK_OK：设置图层句柄成功
  - EPDK_FAIL：设置图层句柄失败
- 示例：

```
__hdle  robin_request_video_layer( const RECT *rect_p, __s32 pipe, __s32 prio )
{
    //__disp_layer_para_t     image_layer_para;
    //FB                      image_fb_para;
    __disp_layer_info_t     image_layer_info;
    __disp_fb_t             image_fb_para;
    RECT                    image_win;
    __hdle                  hlay = NULL;
    __u32 arg[3];

    if( rect_p == NULL )
        return NULL;
```

```
    /* frame buffer attribute is ignored here, that is set by Cedar */

    image_fb_para.size.height       = 0;                    // DONT'T CARE
    image_fb_para.size.width        = 0;                    // DONT'T CARE
    image_fb_para.addr[0]           = NULL;
    image_fb_para.format            = DISP_FORMAT_RGB565;         // DONT'T CARE
    image_fb_para.seq               = DISP_SEQ_ARGB;      // DONT'T CARE
    image_fb_para.mode              = 0;                    // DONT'T CARE
    image_fb_para.br_swap           = 0;                // DONT'T CARE
    image_fb_para.cs_mode           = NULL;

    image_layer_info.mode           = DISP_LAYER_WORK_MODE_NORMAL;
    image_layer_info.pipe           = pipe;
    image_layer_info.prio           = prio;
    image_layer_info.alpha_en       = 0;
    image_layer_info.alpha_val      = 255;
    image_layer_info.ck_enable      = 0;
    image_layer_info.src_win.x      = 0;
    image_layer_info.src_win.y      = 0;
    image_layer_info.src_win.width  = rect_p->width ;
    image_layer_info.src_win.height = rect_p->height;
    image_layer_info.scn_win.x      = rect_p->x     ;
    image_layer_info.scn_win.y      = rect_p->y     ;
    image_layer_info.scn_win.width  = rect_p->width ;
    image_layer_info.scn_win.height = rect_p->height;
    image_layer_info.fb             = image_fb_para;

    arg[0] = DISP_LAYER_WORK_MODE_NORMAL;
    arg[1] = 0;
    arg[2] = 0;
    hlay = g_fioctrl( robin_hdis, DISP_CMD_LAYER_REQUEST, 0, (void *)arg );

    if( hlay == NULL )
    {
        __err("Error in applying for the video layer.\n");
        goto error;
    }

    arg[0] = hlay;
    arg[1] = (__u32)&image_layer_info;
    arg[2] = 0;
    g_fioctrl( robin_hdis, DISP_CMD_LAYER_SET_PARA, 0, (void *)arg );

    image_win.x      = rect_p->x;
    image_win.y      = rect_p->y;
    image_win.width  = rect_p->width ;
    image_win.height = rect_p->height;

    if(esMODS_MIoctrl(robin_hced, CEDAR_CMD_SET_VID_LAYERHDL, 0, (void *)hlay) !=
EPDK_OK)
    {
        __wrn("Fail in setting video layer handle to cedar!\n");
        goto error;
    }
    //set video window information to cedar
    if(esMODS_MIoctrl(robin_hced, CEDAR_CMD_SET_VID_WINDOW, 0, &image_win) != EPDK_OK)
    {
        __wrn("Fail in setting video window information to cedar!\n");
        goto error;
```

```
    }

    return hlay;

error:
    if( hlay != NULL )
    {
        arg[0] = hlay;
        arg[1] = 0;
        arg[2] = 0;
        g_fioctrl( robin_hdis, DISP_CMD_LAYER_RELEASE, 0, (void *)arg );
        hlay = NULL;
    }
    return NULL;
}
```

3.2.54 CEDAR_CMD_SET_VID_WINDOW

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置显示窗口的位置和大小。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_VID_WINDOW；
  - aux = 0；
  - pbuffer = __rect_t *；
- 返回：
  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败
- 示例：

```
__s32  robin_set_video_win( __s32 x, __s32 y, __s32 width, __s32 height )
{
    RECT  image_win;

    image_win.x      = x;
    image_win.y      = y;
    image_win.width  = width;
    image_win.height = height;
    if( esMODS_MIoctrl( robin_hced, CEDAR_CMD_SET_VID_WINDOW, 0, &image_win ) !=
  EPDK_OK )
    {
        __wrn("Fail in setting video window information to cedar!\n");
        return -1;
    }

    return 0;
}
```

3.2.55 CEDAR_CMD_GET_VID_WINDOW

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取显示窗口的位置和大小。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_VID_WINDOW；
  - aux = 0；
  - pbuffer = NULL；
- 返回：显示窗口的位置和大小，__rect_t*。
- 示例：

```
__s32 robin_get_video_win( RECT *rect_p )
{
    RECT  *rect_i;

    if( rect_p == NULL )
        return -1;

    rect_i = (RECT *)esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_VID_WINDOW, 0, NULL );
    if( rect_i == NULL )
        return -1;

    g_memcpy( rect_p, rect_i, sizeof(RECT) );

    return 0;
}
```

3.2.56 CEDAR_CMD_SET_VID_SHOW_MODE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置显示模式。显示模式本质上是解码出来的 framebuffer 和显示驱动的 sceenbuffer 之间的映射关系。即指定 framebuffer 的某个区域，映射到 sceenbuffer 的某个区域。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_VID_SHOW_MODE；
  - aux = __cedar_vide_window_ratio_mode_t；
  - pbuffer = NULL；
- 返回：设置之后的 cedar 当前的显示模式。
- 示例：

```
__s32  robin_set_zoom( robin_zoom_e zoom )
{
    __cedar_vide_window_ratio_mode_t   cedar_zoom;
    __u32  screen_width;
    __u32  screen_height;
    __s32  x;
    __s32  y;
    __s32  width;
    __s32  height;

    cedar_zoom = map_crs2cedar( zoom );

    get_screen_size( &screen_width, &screen_height );
    x      = 0;
    y      = 0;
    width  = screen_width;
    height = screen_height;

    if( zoom == ROBIN_ZOOM_FIT_VIEW )
    {
        __disp_output_type_t  output;

        output  = (__disp_output_type_t)g_fioctrl( robin_hdis, DISP_CMD_GET_OUTPUT_TYPE
, 0, 0 ) ;

        if( output == DISP_OUTPUT_TYPE_TV || output == DISP_OUTPUT_TYPE_HDMI )
        {
            if( screen_width == 1280 && screen_height == 720 )
            {
                x      = (1280-1208)>>1;
                y      = (720-680)>>1;
                width  = 1208;
                height = 680;
            }
            else if( screen_width == 1920 && screen_height == 1080 )
            {
                x      = (1920-1812)>>1;
                y      = (1080-1020)>>1;
                width  = 1812;
                height = 1020;
            }
            else if( screen_width == 720 && screen_height == 480 )
            {
                x      = (720-660)>>1;
                y      = (480-440)>>1;
                width  = 660;
                height = 440;
            }
            else if( screen_width == 720 && screen_height == 576 )
            {
                x      = (720-660)>>1;
                y      = (576-536)>>1;
                width  = 660;
                height = 536;
            }
        }
    }

    robin_set_video_win( x, y, width, height );
    esMODS_MIoctrl( robin_hced, CEDAR_CMD_SET_VID_SHOW_MODE, cedar_zoom, NULL );
```

```
    cur_zoom = zoom;

    return 0;
}
```

3.2.57 CEDAR_CMD_GET_VID_SHOW_MODE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取当前的显示模式。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_VID_SHOW_MODE；
  - aux = 0；
  - pbuffer = NULL；
- 返回：__cedar_vide_window_ratio_mode_t。
- 示例：

```
robin_zoom_e  robin_get_zoom( void )
{
    __cedar_vide_window_ratio_mode_t  cedar_zoom;

    if( cur_zoom == ROBIN_ZOOM_UNKNOWN )
    {
        cedar_zoom = (__cedar_vide_window_ratio_mode_t)esMODS_MIoctrl( robin_hced,
  CEDAR_CMD_GET_VID_SHOW_MODE, 0, NULL );
        return map_cedar2crs( cedar_zoom );
    }
    else
        return cur_zoom;
}
```

3.2.58 CEDAR_CMD_SWITCH_VID_SHOW

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：开关屏的操作。在播放过程中，调用该接口，可以开屏或关屏。关屏后，仍在播放，但屏幕不显示图像了。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SWITCH_VID_SHOW；
  - aux = 是否打开屏。1 表示开屏；0 表示关屏。
  - pbuffer = NULL；

- 返回：
  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败
- 示例：无

3.2.59 CEDAR_CMD_SET_FRPIC_SHOWTIME

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置快进快退状态下，希望的一帧显示的持续时间。Psr 模块就以这个持续时间为准选取关键帧。因为快进快退状态下，时间轴是倍速于正常时间轴的，一般是 64 倍速。那么意味着一帧如果显示 30ms，这个持续时间等于正常情况下 30*64ms，在这么长的时间里，可以显示很多帧，那么在快进快退下，这些帧就应跳过，选下一帧。这就是快进快退一帧显示时间的含义。默认为 30ms。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_FRPIC_SHOWTIME；
  - aux = 希望的显示持续时间，单位 ms；
  - pbuffer = NULL；
- 返回：
  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败
- 示例：无

3.2.60 CEDAR_CMD_GET_FRPIC_SHOWTIME

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：Cedar 没有实现该接口。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_FRPIC_SHOWTIME；
  - aux = 0；
  - pbuffer = NULL；
- 返回：得到当前设置的快进快退状态下，一帧的显示持续时间。
- 示例：无

3.2.61 CEDAR_CMD_SET_ROTATE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置解码出的图像的旋转角度。必须在文件播放前设置，即调用 CEDAR_CMD_PLAY 前设置。并且播放过程中不允许再调用该接口改变旋转角度，如果调用，无效。设置完成后，只要 cedar 主控模块不退出，就一直生效。也就是说对所有文件生效，而不仅是当前要播放的文件。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_ROTATE；
  - aux = 旋转角度设置，0：原图；1：顺时针旋转 90 度；2：顺时针 180 度；3：顺时针 270 度；4：水平镜像；5：垂直镜像；
  - pbuffer = NULL；
- 返回：
  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败
- 示例：无

3.2.62 CEDAR_CMD_INVALID_VIDEOLAYER

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：当播放视频过程中，可能在一些情况下，应用程序要把之前设给 cedar 的图层收回做其他用途，但希望 cedar 仍然能继续播放。对于 cedar 而言，没有图层，是可以继续播放的，只是解码出来的图像不再显示而已。所以 cedar 提供了这个接口，允许在播放过程中，将图层设为无效。等 app 使用完毕后，调用 CEDAR_CMD_SET_VID_LAYERHDL 将图层重设给 cedar，再调用本接口，恢复图层。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_INVALID_VIDEOLAYER；
  - aux = 是否设置图层无效，1：通知 cedar，图层句柄无效；0：通知 cedar，图层句柄恢复有效；
  - pbuffer = NULL；
- 返回：
  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败
- 示例：无

3.2.63 CEDAR_CMD_SET_FILE_SWITCH_VPLY_MODE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：在文件切换时，cedar 提供了两种模式，一是文件切换过程中关屏；二是切换过程中始终显示上一个文件的切换之前正在显示的帧。第二种模式又称无缝切换。现在默认采用无缝切换模式。如果应用程序希望采用第一种模式，调用本接口设置即可。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_FILE_SWITCH_VPLY_MODE；
  - aux = CedarFileSwitchVplyMode；
  - pbuffer = NULL；
- 返回：设置文件切换模式的结果
  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败
- 示例：无

3.2.64 CEDAR_CMD_ENABLE_VIDEO_AUTO_SCALE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：因为芯片性能的原因，在解 1080p 等高分辨率视频文件时，常常带宽不够造成屏幕闪烁，这个接口允许应用程序定制 cedar 内部的视频解码驱动的行为：遇到高分辨率文件时，是否做 scale 以减少带宽和内存消耗。所谓 scale 是指在解码时压缩图像，例如一幅图像原本正常解码出来的宽高是 1920*1080*，经过 *scale*，解码出来变为 *960*1080 等，这样图像变小了，输出的显示带宽自然变小了，内存消耗也小了。Cedar 默认不做自动 scale。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_ENABLE_VIDEO_AUTO_SCALE；
  - aux = 1：允许在解码过程中自动做 scale；0：不允许自动做 scale；
  - pbuffer = NULL；
- 返回：
  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败
- 示例：无

3.2.65 CEDAR_CMD_GET_LBSTYPE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`

- 功能描述：获取当前的字幕文件格式，对于内置字幕，也定义了一套文件格式的值。需要注意的是：对于文本字幕，其内部文字的编码格式，是另有一套枚举类型定义值的，就是 __cedar_subtitle_encode_t。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_LBSTYPE；
  - aux = 0；
  - pbuffer = NULL；
- 返回：字幕文件格式：__cedar_lyric_fmt_t。
- 示例：

```
__cedar_lyric_fmt_t  robin_get_subtitle_format( void )
{
    __cedar_lyric_fmt_t  ret;
    __u8   err;

    g_pend_mutex( robin_cedar_mutex, &err );
    robin_wait_no_file( );

    ret = (__cedar_lyric_fmt_t)esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_LBSTYPE, 0,
  NULL );

    g_post_mutex( robin_cedar_mutex );
    return ret;
}
```

3.2.66 CEDAR_CMD_GET_SUB_INFO

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取当前字幕流的字幕条目，或者根据 PTS 拿一个合适的字幕条目，或者拿全部的。因为内存是 cedar 内部分配的，所以应用程序要注意，不要改动内存的内容。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_SUB_INFO；
  - aux = nPts, 单位 ms；
  - pbuffer = __cedar_get_sub_inf_t，字幕条目的类型；具体有：CEDAR_GET_SUB_INF_ALL：一次拿到所有的字幕条目。CEDAR_GET_SUB_INF_ITEM：根据送入的 nPts 选一条合适的字幕条目给出来。类型不同，返回值的类型也不同。
- 返回：如果类型是 CEDAR_GET_SUB_INF_ALL，返回值为 __cedar_buf_inf_t *；如果类型是 CEDAR_GET_SUB_INF_ITEM ，返回值为 __cedar_subtitle_item_t* 。
- 示例：

```
__s32  robin_get_subtitle_item( __u32 time, __cedar_subtitle_item_t *subtitle_item_p )
{
    __cedar_subtitle_item_t  *p = NULL;
    __u8   err;

    if( subtitle_item_p == NULL )
        return -1;

    g_pend_mutex( robin_cedar_mutex, &err );
    robin_wait_no_file( );

    p = (__cedar_subtitle_item_t *)esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_SUB_INFO,
  time,
                                                   (void *)CEDAR_GET_SUB_INF_ITEM );
    if( p == NULL )           // no subtitle information at present
    {
        g_post_mutex( robin_cedar_mutex );
        return -1;
    }
    else
    {
        g_memcpy( subtitle_item_p, p, sizeof(__cedar_subtitle_item_t) );
        g_post_mutex( robin_cedar_mutex );
        return 0;
    }
}
```

3.2.67 CEDAR_CMD_GET_SUBTITLE_PROFILE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取所有的字幕流信息。这是第一版接口，规定了最多只能有 8 个字幕流。第二版接口做了扩展，可以获取的字幕流数量不受限制。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_SUBTITLE_PROFILE；
  - aux = 0；
  - pbuffer = __subtitle_profile_t *, app 分配内存；
- 返回：
  - EPDK_OK：获取成功
  - EPDK_FAIL：获取失败
- 示例：

```
__s32  robin_get_subtitle_list( __subtitle_profile_t  *subtitle_info_p )
{
    __u8   err;

    if( subtitle_info_p == NULL )
        return -1;
```

```
    g_pend_mutex( robin_cedar_mutex, &err );
    robin_wait_no_file( );

    if( esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_SUBTITLE_PROFILE, 0, subtitle_info_p
  ) == EPDK_OK )
    {
        g_post_mutex( robin_cedar_mutex );
        return 0;
    }
    else
    {
        g_post_mutex( robin_cedar_mutex );
        return -1;
    }
}
```

3.2.68 CEDAR_CMD_SELECT_SUBTITLE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：通知 cedar 换字幕流。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SELECT_SUBTITLE；
  - aux = 字幕流的下标；
  - pbuffer = NULL；
- 返回：
  - EPDK_OK：设置成功
  - EPDK_FAIL：设置失败
- 示例：

```
__s32  robin_select_subtitle( __u32 subtitle_index )
{
    __u8   err;

    g_pend_mutex( robin_cedar_mutex, &err );
    robin_wait_no_file( );

    if( esMODS_MIoctrl( robin_hced, CEDAR_CMD_SELECT_SUBTITLE, subtitle_index, NULL )
  == EPDK_OK )
    {
        g_post_mutex( robin_cedar_mutex );
        return 0;
    }
    else
    {
        g_post_mutex( robin_cedar_mutex );
        return -1;
```

```
    }
}
```

3.2.69 CEDAR_CMD_GET_SUBTITLE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取当前正在播放的字幕流下标。下标从 0 开始。-1 表示字幕还没有检测完毕。遇到这种情况，应用程序应该 delay 一段时间继续查。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_SUBTITLE；
  - aux = 0；
  - pbuffer = NULL；
- 返回：字幕流下标。
- 示例：

```
__s32  robin_get_subtitle_index( void )
{
    __s32  ret;
    __u8   err;

    g_pend_mutex( robin_cedar_mutex, &err );
    robin_wait_no_file( );

    ret = (__s32)esMODS_MIoctrl( robin_hced, CEDAR_CMD_GET_SUBTITLE, 0, NULL );

    g_post_mutex( robin_cedar_mutex );
    return ret;
}
```

3.2.70 CEDAR_CMD_GET_SUBTITLE_CNT

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：该接口配合 CEDAR_CMD_GET_SUBTITLE_PROFILE_V2 一起使用；-1 表示字幕解析过程还没完成，所以得不到字幕流的数量。应用程序遇到这种情况应 delay 一段时间继续调用本接口。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_SUBTITLE_CNT；
  - aux = 0；
  - pbuffer = NULL；

- 返回：字幕流的数量，-1 表示字幕解析过程还没完成。
- 示例：无

3.2.71 CEDAR_CMD_GET_SUBTITLE_PROFILE_V2

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取视频文件所有的音频流信息。如果是音频文件，返回值为 EPDK_FAIL。该接口一般和 CEDAR_CMD_GET_SUBTITLE_CNT 联合使用，CEDAR_CMD_GET_SUBTITLE_CNT 得到音轨数，然后为 __subtitle_profile_v2_t 分配内存，再调用该接口。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_SUBTITLE_PROFILE_V2；
  - aux = 0；
  - pbuffer = __subtitle_profile_v2_t*；app 分配内存；
- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：无

3.2.72 CEDAR_CMD_SET_SUBTITLE_ITEM_POST_PROCESS

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置是否对解码出来的文本字幕条目进行后处理，去除特殊字符。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_SUBTITLE_ITEM_POST_PROCESS；
  - aux = 1: 对文本字幕条目进行后处理，所谓后处理就是去除一些特殊字符；0：不进行后处理；
  - pbuffer = NULL；
- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：无

3.2.73 CEDAR_CMD_ENABLE_EXTERN_SUBTITLE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置 cedar 是否进行外挂字幕检测，不进行外挂字幕检测可以加快文件打开速度。Cedar 默认检测，如果应用程序不想检测字幕，可以调用该接口设置。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_ENABLE_EXTERN_SUBTITLE；
  - aux = 1：进行外挂字幕文件检测；0：禁止检测外挂字幕文件；
  - pbuffer = NULL；
- 返回：图像操作的结果
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：无

3.2.74 CEDAR_CMD_CAPTURE_PIC

已废弃。

3.2.75 CEDAR_CMD_ASK_PIC_BUFSIZE

已废弃。

3.2.76 CEDAR_CMD_GET_FRAME_PIC

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：截图，app 指定截取之后的图的类型，大小，并给定 buffer。准确的说，这个接口并不会给出一张完整的图。例如 BMP，这个接口只给出 bmp 图的内容部分，bmp 的头信息还是由应用程序自己制作。这个接口是同步的，一旦返回就表示截图完成。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_FRAME_PIC；
  - aux = 0；
  - pbuffer = FB*；APP 分配内存，必须用 ARGB 格式。FB 的参数也必须配置正确，因为 cedar 不查错。
- 返回：

- EPDK_OK：成功
- EPDK_FAIL：失败

- 示例：无

3.2.77 CEDAR_DUCKWEED_CMD_OPEN_MEDIAFILE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：打开文件。这个接口是做预览图的接口之一。Cedar 后期集成了预览图中间件，所以也继承了原预览图中间件几乎所有的接口，并和原中间件的操作尽量保持一致，以方便应用程序的修改。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_DUCKWEED_CMD_OPEN_MEDIAFILE；
  - aux = 0；
  - pbuffer = NULL；
- 返回：操作的结果
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：

```
//打开文件
result = esMODS_MIoctrl(pCedar, CEDAR_DUCKWEED_CMD_OPEN_MEDIAFILE, 0, NULL);
```

3.2.78 CEDAR_DUCKWEED_CMD_CLOSE_MEDIAFILE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：关闭媒体文件。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_DUCKWEED_CMD_CLOSE_MEDIAFILE；
  - aux = 0；
  - pbuffer = NULL；
- 返回：操作的结果
  - EPDK_OK：成功

- EPDK_FAIL：失败

- 示例：

```
//关闭文件
result = esMODS_MIoctrl(pCedar, CEDAR_DUCKWEED_CMD_CLOSE_MEDIAFILE, 0, NULL);
```

3.2.79 CEDAR_DUCKWEED_CMD_GET_FILE_FORMAT

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_DUCKWEED_CMD_GET_FILE_FORMAT；
  - aux = 0；
  - pbuffer = NULL；
- 返回：文件格式；__cedar_media_file_fmt_t。
- 示例：

```
//得到文件格式
result = esMODS_MIoctrl(pCedar, CEDAR_DUCKWEED_CMD_GET_FILE_FORMAT, 0, NULL);
```

3.2.80 CEDAR_DUCKWEED_CMD_GET_FILE_SIZE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取文件大小，因为返回值是 __s32，而有的文件长度超出 __s32 的范围，所以文件长度放在 pbuffer 中返回给应用程序。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_DUCKWEED_CMD_GET_FILE_SIZE；
  - aux = 0；
  - pbuffer = __s64*；
- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：

```
//得到文件大小
result = esMODS_MIoctrl(pCedar, CEDAR_DUCKWEED_CMD_GET_FILE_SIZE, 0, &file_size);
```

3.2.81 CEDAR_DUCKWEED_CMD_GET_TOTAL_TIME

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取视频播放的总时间。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_DUCKWEED_CMD_GET_TOTAL_TIME；
  - aux = 0；
  - pbuffer = NULL；
- 返回：视频的总时间，单位 ms。
- 示例：

```
//得到播放总时间
result = esMODS_MIoctrl(pCedar, CEDAR_DUCKWEED_CMD_GET_TOTAL_TIME, 0, NULL);
```

3.2.82 CEDAR_DUCKWEED_CMD_GET_VIDSTREAM_CNT

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取视频文件的视频流数量。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_DUCKWEED_CMD_GET_VIDSTREAM_CNT；
  - aux = 0；
  - pbuffer = NULL；
- 返回：视频文件的视频流数量。
- 示例：无

3.2.83 CEDAR_DUCKWEED_CMD_GET_VIDSTREAM_PROFILE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取视频流信息。
- 参数：
  - mp = cedar 模块的句柄；

- cmd = CEDAR_DUCKWEED_CMD_GET_VIDSTREAM_PROFILE;
- aux = 0;
- pbuffer = __vidstream_profile_v2_t*；app 分配内存

- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败

- 示例：

```
//得到视频流信息
result = esMODS_MIoctrl(pCedar, CEDAR_DUCKWEED_CMD_GET_VIDSTREAM_CNT, 0, NULL);
pTestApp->VProf.nVidStrmMaxCnt = (__u8)result;
pTestApp->VProf.nVidStrmNum = 0;
if(pTestApp->VProf.VidStrmListArray)
{
    free(pTestApp->VProf.VidStrmListArray);
}
pTestApp->VProf.VidStrmListArray = (__video_bs_info_t*)malloc(sizeof(__video_bs_info_t)
  *pTestApp->VProf.nVidStrmMaxCnt);
if(pTestApp->VProf.VidStrmListArray == NULL)
{
    WARNING("malloc fail\n");
    goto _exit_test_app;
}
memset(pTestApp->VProf.VidStrmListArray, 0, sizeof(__video_bs_info_t)*pTestApp->VProf.
  nVidStrmMaxCnt);
result = esMODS_MIoctrl(pCedar, CEDAR_DUCKWEED_CMD_GET_VIDSTREAM_PROFILE, 0, &pTestApp
  ->VProf);
```

3.2.84 CEDAR_DUCKWEED_CMD_GET_AUDSTREAM_CNT

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取视频文件的音频流数量。
- 参数：
  - mp = cedar 模块的句柄;
  - cmd = CEDAR_DUCKWEED_CMD_GET_AUDSTREAM_CNT;
  - aux = 0;
  - pbuffer = NULL;
- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：无

3.2.85 CEDAR_DUCKWEED_CMD_GET_AUDSTREAM_PROFILE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取音频流信息。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_DUCKWEED_CMD_GET_AUDSTREAM_PROFILE；
  - aux = 0；
  - pbuffer = __audstream_profile_v2_t*；app 分配内存
- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：

```
//得到音频流信息
result = esMODS_MIoctrl(pCedar, CEDAR_DUCKWEED_CMD_GET_AUDSTREAM_CNT, 0, NULL);
WARNING("file audio stream count is [%d]\n", result);
pTestApp->AProf.nAudStrmMaxCnt = (__u8)result;
pTestApp->AProf.nAudStrmNum = 0;
if(pTestApp->AProf.AudStrmListArray)
{
    free(pTestApp->AProf.AudStrmListArray);
}
pTestApp->AProf.AudStrmListArray = (__audio_bs_info_t*)malloc(sizeof(__audio_bs_info_t)
  *pTestApp->AProf.nAudStrmMaxCnt);
if(pTestApp->AProf.nAudStrmMaxCnt == NULL)
{
    WARNING("malloc fail\n");
    goto _exit_test_app;
}
memset(pTestApp->AProf.AudStrmListArray, 0, sizeof(__audio_bs_info_t)*pTestApp->AProf.
  nAudStrmMaxCnt);
result = esMODS_MIoctrl(pCedar, CEDAR_DUCKWEED_CMD_GET_AUDSTREAM_PROFILE, 0, &pTestApp
  ->AProf);
```

3.2.86 CEDAR_DUCKWEED_CMD_GET_PREVIEW_FB

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取视频文件的第一帧图像。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_DUCKWEED_CMD_GET_PREVIEW_FB；

- aux = 0;
- pbuffer = FB*; app 分配内存

- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败

- 示例：

```
memset(fb, 0, sizeof(FB));
fb->size.width = PREVIEW_WIDTH;
fb->size.height = PREVIEW_HEIGHT;
fb->addr[0] = esMEMS_Palloc(((fb->size.width  * fb->size.height * 4 + 1023)>>10),
  MEMS_PALLOC_MODE_BND_NONE | MEMS_PALLOC_MODE_BNK_NONE);
if(NULL == fb->addr[0])
{
    WARNING("palloc fail\n");
    return EPDK_FAIL;
}
fb->addr[1] = NULL;
fb->addr[2] = NULL;
fb->fmt.type = FB_TYPE_RGB;
fb->fmt.cs_mode = BT601;
fb->fmt.fmt.rgb.pixelfmt = PIXEL_COLOR_ARGB8888;
fb->fmt.fmt.rgb.br_swap = 0;
fb->fmt.fmt.rgb.pixseq = RGB_SEQ_ARGB;
fb->fmt.fmt.rgb.palette.addr = NULL;
fb->fmt.fmt.rgb.palette.size = 0;
esMODS_MIoctrl(pCedar, CEDAR_DUCKWEED_CMD_GET_PREVIEW_FB, 0, &fb);
```

3.2.87 CEDAR_DUCKWEED_CMD_GET_PREVIEW_FB_BY_PTS

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：在指定的时间点附件选一个关键帧，解码获取其图像。转为 FB* 指定的规格。
- 参数：
  - mp = cedar 模块的句柄;
  - cmd = CEDAR_DUCKWEED_CMD_GET_PREVIEW_FB_BY_PTS;
  - aux = 时间点，单位 ms;
  - pbuffer = FB*,APP 分配内存;

- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败

- 示例：

```
result = esMODS_MIoctrl(pCedar, CEDAR_DUCKWEED_CMD_GET_PREVIEW_FB_BY_PTS, 3000, &
  pTestApp->previewFB);
```

3.2.88 CEDAR_CMD_SET_USER_FILEOP

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：如果应用程序想通过 callback 方式，由自己去读文件，那么就通过这个接口，将 callback 函数设置给 cedar，cedar 读取文件时，实际上是调用 app 的函数去读取。这个方式一般用于读取加密文件，cedar 不知道解密方法，而 app 知道，所以 app 在自己的函数里读取加密文件并解密，这样 cedar 就不需要关心解密的操作了。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_USER_FILEOP；
  - aux = 0；
  - pbuffer = __cedar_usr_file_op_t *,app 分配内存；
- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：

```
__cedar_usr_file_op_t   UsrFileOp;
memset(&UsrFileOp, 0, sizeof(__cedar_usr_file_op_t));
pApp->pFile = eLIBs_fopen(pApp->media_file.file_path, "r");
if(!pApp->pFile)
{
    return EPDK_FAIL;
}
fseek(pApp->pFile, 0, SEEK_END);
UsrFileOp.usr_fread = esKRNL_GetCallBack(usr_file_read);//inka_file_read
UsrFileOp.fp = (__u32)pApp->pFile;
UsrFileOp.media_fmt = CEDAR_MEDIA_FILE_FMT_AVI; //这里用普通的AVI文件做demo,
UsrFileOp.file_size = ftell(pApp->pFile);
UsrFileOp.flag = 0; //没有用到
UsrFileOp.usr_fseek = NULL; //没有用到
fseek(pApp->pFile, 0, SEEK_SET);
esMODS_MIoctrl(pCedar, CEDAR_CMD_SET_USER_FILEOP, 0, &UsrFileOp);
```

3.2.89 CEDAR_CMD_SET_STOP_MODE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置 stop play 时播放器是否卸载所有插件，包括 parser、decoder、playbak 等插件和驱动。

- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_STOP_MODE；
  - aux = CedarStopMode。stop 时是保留所有插件，还是卸载所有插件。aux = CEDAR_STOP_MODE_KEEP_PLUGINS：保留插件，会加快文件切换速度，默认使用保留插件的模式；aux = CEDAR_STOP_MODE_UNINSTALL_PLUGINS：卸载所有插件，在这种模式下，无缝切换无效。
  - pbuffer = NULL；
- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：

```
result = esMODS_MIoctrl(pCedar, CEDAR_CMD_SET_STOP_MODE, cedar_stop_mode, NULL);
```

3.2.90 CEDAR_CMD_SET_PITCH

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置音频播放时候变调播放，并设置变调速度。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_PITCH；
  - aux = 音频变调范围，-12~12；
  - pbuffer = NULL；
- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：

```
result = esMODS_MIoctrl(pCedar, CEDAR_CMD_SET_PITCH, aux, NULL);
```

3.2.91 CEDAR_CMD_GET_PITCH

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取音频变调速度。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_PITCH；
  - aux = 0；
  - pbuffer = NULL；
- 返回：音调速度。
- 示例：

```
result = esMODS_MIoctrl(pCedar, CEDAR_CMD_GET_PITCH, 0, NULL);
```

3.2.92 CEDAR_CMD_PLAY_AUX_WAV_FILE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：用于在音视频播放的时候播放一些附加的 wav 声音数据，该数据跟解码音频数据混音播放。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_PLAY_AUX_WAV_FILE；
  - aux = __cedar_play_aux_wav_mode_t；
  - pbuffer = wav 数据播放文件路径；
- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：

```
result = esMODS_MIoctrl(pCedar, CEDAR_CMD_PLAY_AUX_WAV_FILE, aux, pbuffer);
```

3.2.93 CEDAR_CMD_PLAY_AUX_WAV_BUFFER

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`

- 功能描述：用于在音视频播放的时候播放一些附加的 wav 声音数据，该数据跟解码音频数据混音播放，功能和 CEDAR_CMD_PLAY_AUX_WAV_FILE 一样，只是参数不一样。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_PLAY_AUX_WAV_BUFFER；
  - aux = __cedar_play_aux_wav_mode_t；
  - pbuffer = wav 数据播放参数 __cedar_pcm_info_t；
- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：

```
eLIBs_memset(&pcm_info, 0, sizeof(__cedar_pcm_info_t));
pcm_info.Chan = wav.uChannels;
pcm_info.PcmLen = wav.uSampDataSize/(wav.uBitsPerSample/8)/wav.uChannels;
pcm_info.preamp = 0;
pcm_info.SampleRate = wav.uSampleRate;
pcm_info.PCMPtr = (__u16*)pwav_buf;
ret = MIoctrl(pCedar, CEDAR_CMD_PLAY_AUX_WAV_BUFFER, aux, &pcm_info);
```

3.2.94 CEDAR_CMD_SET_AUX_WAV_BUFFER_SIZE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：CEDAR_CMD_PLAY_AUX_WAV_BUFFER 播放的附加声音数据的 buffer 大小，音频驱动初始化时就会按此值分配好内存。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_AUX_WAV_BUFFER_SIZE；
  - aux = 设置 wav 数据播放 buffer 大小, aux=buffer size；
  - pbuffer = NULL；
- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：

```
result= esMODS_MIoctrl(pCedar, CEDAR_CMD_SET_AUX_WAV_BUFFER_SIZE, aux, NULL);
```

3.2.95 CEDAR_CMD_GET_AUX_WAV_BUFFER_SIZE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：CEDAR_CMD_PLAY_AUX_WAV_BUFFER 播放的附加声音数据的 buffer 大小，音频驱动初始化时就会按此值分配好内存。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_AUX_WAV_BUFFER_SIZE；
  - aux = 0；
  - pbuffer = NULL；
- 返回：返回 wav 数据播放 buffer 大小。
- 示例：

```
result= esMODS_MIoctrl(pCedar, CEDAR_CMD_GET_AUX_WAV_BUFFER_SIZE, 0, NULL);
```

3.2.96 CEDAR_CMD_GET_PLY_DATA_STATUS

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取 playback 中一定时间内是否有数据，即是否有音视频数据马上要播放。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_PLY_DATA_STATUS；
  - aux = 0；
  - pbuffer = NULL；
- 返回：
  - 非 1：没有数据
  - 1：有数据
- 示例：

```
result= esMODS_MIoctrl(pCedar, CEDAR_CMD_GET_PLY_DATA_STATUS, 0, NULL);
```

3.2.97 CEDAR_CMD_STREAM_SET_INFO

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：应用传输 h264/h265 裸流数据播放前，设置视频信息。

- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_STREAM_SET_INFO；
  - aux = 0；
  - pbuffer = 设置 h264/h265 裸流数据信息，__stream_inf_t。
- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：

```
__s32 robin_set_stream_info(__stream_inf_t* streaminf)
{
return esMODS_MIoctrl( robin_hced, CEDAR_CMD_STREAM_SET_INFO,0, (void*)streaminf);
}
```

3.2.98 CEDAR_CMD_STREAM_QUERY_BUFFER

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：专用于 h264/h265 裸流数据播放接口。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_STREAM_QUERY_BUFFER；
  - aux = 0；
  - pbuffer = NULL；
- 返回：返回播放器缓存区域还有几帧数据没有解码，个数
- 示例：

```
__s32 robin_set_stream_info(__stream_inf_t* streaminf)
{
return esMODS_MIoctrl( robin_hced, CEDAR_CMD_STREAM_QUERY_BUFFER,0, (void*)streaminf);
}
```

3.2.99 CEDAR_CMD_STREAM_WRITE_BUFFER

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：专用于 h264/h265 裸流数据播放接口。
- 参数：

- mp = cedar 模块的句柄；
- cmd = CEDAR_CMD_STREAM_WRITE_BUFFER；
- aux = 一帧数据的长度；
- pbuffer = 帧数据 buffer，注意数据前面 4 字节代表帧长度，可能需要应用在数据头加上。

- 返回：

  - EPDK_OK：成功
  - EPDK_FAIL：失败

- 示例：

```
__s32 robin_write_buffer(__u32 buffer_len, __u8 * buffer)
{
    return esMODS_MIoctrl( robin_hced, CEDAR_CMD_STREAM_WRITE_BUFFER, buffer_len, (void
   *)buffer );
}
```

3.2.100 CEDAR_CMD_STREAM_END

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：专用于 h264/h265 裸流数据播放接口。通知播放器数据传输完毕。
- 参数：

  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_STREAM_END；
  - aux = 0；
  - pbuffer = NULL；

- 返回：

  - EPDK_OK：成功
  - EPDK_FAIL：失败

- 示例：

```
__s32 robin_set_stop(void)
{
    return esMODS_MIoctrl( robin_hced, CEDAR_CMD_STREAM_END, 0, NULL );
}
```

3.2.101 CEDAR_CMD_FAST_LOOPPLAY_ENABLE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：使能音频循环播放模式，只支持音频格式。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = ；
  - aux = 0；
  - pbuffer = NULL；
- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：

```
void robin_loop_play_enble()
{
    esMODS_MIoctrl(robin_hced, CEDAR_CMD_FAST_LOOPPLAY_ENABLE, 0, NULL);
}
```

3.2.102 CEDAR_CMD_FAST_LOOPPLAY_DISENABLE

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：关闭音频循环播放模式，只支持音频格式。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_FAST_LOOPPLAY_DISENABLE；
  - aux = 0；
  - pbuffer = NULL；
- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：

```
void robin_loop_play_disenble()
{
    esMODS_MIoctrl(robin_hced, CEDAR_CMD_FAST_LOOPPLAY_DISENABLE, 0, NULL);
}
```

3.2.103 CEDAR_CMD_SET_FAST_LOOPPLAY_CNT

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：设置音频循环播放次数，只支持音频格式。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_SET_FAST_LOOPPLAY_CNT；
  - aux = 循环播放次数；
  - pbuffer = NULL；
- 返回：
  - EPDK_OK：成功
  - EPDK_FAIL：失败
- 示例：

```
void robin_set_loop_play_cnt(__s32 count)
{
   esMODS_MIoctrl(robin_hced, CEDAR_CMD_SET_FAST_LOOPPLAY_CNT, count, NULL);
}
```

3.2.104 CEDAR_CMD_GET_FAST_LOOPPLAY_CNT

- 函数原型：`__s32 esMODS_MIoctrl(__mp *mp, __u32 cmd, __s32 aux, void *pbuffer);`
- 功能描述：获取音频循环播放次数，只支持音频格式。
- 参数：
  - mp = cedar 模块的句柄；
  - cmd = CEDAR_CMD_GET_FAST_LOOPPLAY_CNT；
  - aux = 0；
  - pbuffer = NULL；
- 返回：返回循环次数
- 示例：

```
__s32 robin_get_loop_play_cnt()
{
   return esMODS_MIoctrl(robin_hced, CEDAR_CMD_GET_FAST_LOOPPLAY_CNT, 0, NULL);
}
```

4 常用功能 DEMO

4.1 音视频播放 demo

4.1.1 软件环境配置

在 melis 的根目录执行 make menuconfig 命令，选中 cedar test 测试样例。

```
Kernel Setup
    -->Drivers Test Sample
        -->cedar Test
```

如下图所示：

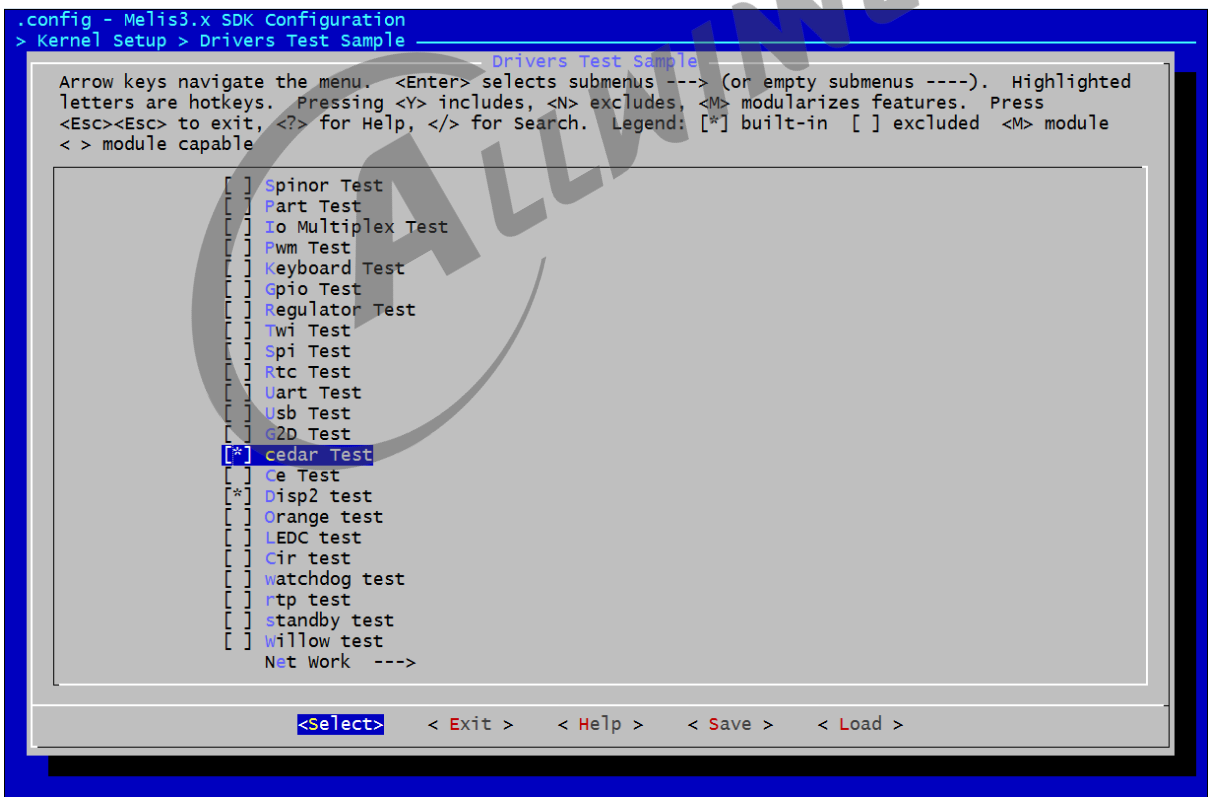

图 4-1: 配置 cedar test 测试样例

4.1.2 代码示例

```
static __s32 do_play_file(__cedar_media_file_inf *param)
{
    __cedar_status_t status;

    mid_cedar = esMODS_MInstall("d:\\mod\\cedar.mod", 0);
    if (!mid_cedar)
    {
        printf("open cedar fail!\n");
        return -1;
    }
    robin_hce = esMODS_MOpen(mid_cedar, 0);

    disp_open();

    video_layer = setup(robin_hce);
    if (!video_layer)
    {
        printf("set up layer failed!\n");
        return -1;
    }

    esMODS_MIoctrl(robin_hce, CEDAR_CMD_SET_VOL, AUDIO_DEV_VOLUME, NULL);
    esMODS_MIoctrl(robin_hce, CEDAR_CMD_SET_MEDIAFILE, 0, (void *)param);
    esMODS_MIoctrl(robin_hce, CEDAR_CMD_STOP, 0, NULL);
    esMODS_MIoctrl(robin_hce, CEDAR_CMD_PLAY, 0, NULL);

    return 0;
}
```

具体实现流程可以参考 ekernel\drivers\test\test_video_play.c，ekernel\drivers\test\test_audio_play.c，或者 app_movie.c。

4.2 Wav/Pcm 音频播放 demo

```
__s32 dsk_keytone_init(const char *keytone_file)
{
    ES_FILE                 *pfile2 = NULL;
    __audio_dev_para_t      pbuf2;
    __wave_header_t1        wav;
    unsigned long arg[2]={0};
    reg_system_para_t *setting_reg_para = (reg_system_para_t *)dsk_reg_get_para_by_app(
    REG_APP_SYSTEM);
#if (USE_KEY_TONE==0)
    return EPDK_OK;//112350
#endif
    keytone = (DKTone *)g_malloc(sizeof(DKTone));

    if (!keytone)
    {
        __err(" DKTone malloc error ");
        return EPDK_FAIL;
```

```
    }

    eLIBs_memset(keytone, 0x00, sizeof(DKTone));
    keytone->f_audiodev = open("/dev/audio_play1", O_RDWR);

    if (keytone->f_audiodev == NULL)
    {
        __err("keytone->f_audiodev == NULL");
    }

    //  keytone->state = SET_KEYTONE_ON;
    keytone->state = setting_reg_para->keytone;
    keytone->state = SET_KEYTONE_ON;
    pfile2 = eLIBs_fopen(keytone_file, "rb");

    if (pfile2 == 0)
    {
        __err("%s cannot open ", keytone_file);
        return EPDK_OK;
    }

    eLIBs_memset(&wav, 0x00, sizeof(__wave_header_t1));
    eLIBs_fread(&wav, 1, sizeof(__wave_header_t1), pfile2);
    g_wav_sample_size = wav.uSampDataSize;
    keytone->wavsize = wav.uSampDataSize;
    keytone->tonebuf    = (__u8 *)esMEMS_Malloc(0, g_wav_sample_size);

    if (!keytone->tonebuf)
    {
        __err(" tonebuf malloc error ");
        return EPDK_FAIL;
    }

    eLIBs_memset(keytone->tonebuf, 0x00, g_wav_sample_size);
    eLIBs_fread(keytone->tonebuf, 1, g_wav_sample_size, pfile2);
    eLIBs_fclose(pfile2);
    pfile2 = NULL;
    eLIBs_memset(&pbuf2, 0x00, sizeof(__audio_dev_para_t));
    pbuf2.bps   = wav.uBitsPerSample;
    pbuf2.chn   = wav.uChannels;
    pbuf2.fs    = wav.uSampleRate;
    arg[0] = 0;
    arg[1] = &pbuf2;
    ioctl(keytone->f_audiodev, AUDIO_DEV_CMD_SET_PARA,    arg);
    arg[0] = AUDIO_PLAY_USR_KEY;
    arg[1] = 0;
    ioctl(keytone->f_audiodev, AUDIO_DEV_CMD_REG_USERMODE, arg);
    arg[0] = 0;
    arg[1] = 0;
    ioctl(keytone->f_audiodev, AUDIO_DEV_CMD_START,        arg);
    return EPDK_OK;
}

__s32 dsk_keytone_on(void)
{
#if (USE_KEY_TONE==0)
    return EPDK_OK;//112350
#endif
    unsigned long arg[2]={0};
```

```
    if ((keytone->state == SET_KEYTONE_ON) && (keytone->tonebuf) && (keytone->f_audiodev))
    {
        g_mute_status = dsk_amplifier_is_on();
        dsk_amplifier_onoff(1);//打开功放
        esKRNL_TimeDly(10); //(20);

        write(keytone->f_audiodev,keytone->tonebuf, keytone->wavsize);
        arg[0] = 0;
        arg[1] = 0;
        ioctl(keytone->f_audiodev, AUDIO_DEV_CMD_DRAIN, arg);
        esKRNL_TimeDly(15); //(50);

        if (1 == g_mute_status)
        {
            dsk_amplifier_onoff(1);//打开功放
        }
        else
        {
            dsk_amplifier_onoff(0);//关闭功放
        }
    }

    return EPDK_OK;
}
```

Pcm/Wav 音频播放可以直接通过往音频播放设备写数据实现，流程简单。具体参考 livedesk\beetles\mod_desktop\functions\keytone\dsk_keytone.c。

4.3 H264 裸流数据播放 demo

```
__s32 video_decode_file(__u8 rotate_mode)
{
    static __s32 count = 0;
    __u32 uFreeMemSize = 0;
    ES_FILE     *fp_video = NULL;
    __s32       copy_size = 0;
    __u8        num[4] = {0};
    __u32       buffer_len = 0;
    robin_open_arg_t arg;
    __s32 ret;
    __u8 i;
    __cedar_media_file_inf *pbuffer;
    __cedar_media_file_inf *pbuffer_backup;
    __s32 screen_width, screen_height;
    RECT vedio_rect;
    __hdle hvedio_lyr;
    ES_FILE *hdisp;
    /*------------------------------------------------------ */
    hdisp = eLIBs_fopen("b:\\DISP\\DISPLAY", "r");

    if (!hdisp)
    {
        __msg("open disp fail...\n");
    }
```

```
    eLIBs_printf("open disp success...\n");
    arg.reserve_mem_size = 750 * 1024;
    ret = robin_open(ROBIN_MODE_VIDEO_MAX, &arg);

    if (EPDK_OK != ret)
    {
        __msg("robin_open fail...\n");

        if (hdisp)
        {
            eLIBs_fclose(hdisp);
            hdisp = NULL;
        }

        return EPDK_FAIL;
    }

    dsk_display_get_size(&screen_width, &screen_height);
    vedio_rect.x = 0;
    vedio_rect.y = 0;
    vedio_rect.width = screen_width;
    vedio_rect.height = screen_height;
    eLIBs_printf("vedio screen=%d %d %d %d", vedio_rect.x, vedio_rect.y, vedio_rect.width,
    vedio_rect.height);
    hvedio_lyr = robin_request_video_layer(&vedio_rect, 0, 0xff);

    if (!hvedio_lyr)
    {
        __msg("robin_request_vedio_layer fail...\n");
        goto l_decode_exit;
        //return EPDK_FAIL;
    }

    eLIBs_printf("robin_request_vedio_layer success...\n");
    {
        __s32 ret;
        __u64 arg[3] = {0};
        arg[0] = hvedio_lyr;
        ret = esMODS_MIoctrl(hdisp, MOD_DISP_CMD_LAYER_BOTTOM, 0, (void *)arg);

        if (0 != ret)
        {
            __msg("MOD_DISP_CMD_LAYER_BOTTOM fail...\n");
        }
        else
        {
            __msg("MOD_DISP_CMD_LAYER_BOTTOM success...\n");
        }
    }
    /*------------------------------------------------------ */
#if 1
    eLIBs_printf("open %s\n", TEST_H264_FILE__);
    fp_video = eLIBs_fopen(TEST_H264_FILE__, "rb");

    if (!fp_video)
    {
        eLIBs_printf("=======open logo file fail=========\n");
        return -1;
        //goto l_decode_exit;
    }
```

```
    copy_size = eLIBs_fread(num, 1, 4, fp_video);

    if (copy_size != 4)
    {
        eLIBs_fclose(fp_video);
        fp_video = 0;
        __wrn("copy size =%d\n", copy_size);
        goto l_decode_exit;
        //return 0;
    }

    eLIBs_printf("[11111111111111111111111111]\n");
    buffer_len = (num[0] & 0xff) | ((num[1] & 0xff) << 8) | ((num[2] & 0xff) << 16) | ((num
    [3] & 0xff) << 24);
    pbuffer = (__cedar_media_file_inf *)esMEMS_Malloc(0, sizeof(__cedar_media_file_inf));

    if (pbuffer == 0)
    {
        __wrn("pbuffer malloc failed\n");
        //return EPDK_FAIL;
        goto l_decode_exit;
    }

    pbuffer->stream_buf_inf.buffer = (__u8 *)esMEMS_Malloc(0, FRAME_BUFFER_SIZE);

    if (!pbuffer->stream_buf_inf.buffer)
    {
        __wrn("pbuffer->stream_buf_inf.buffer malloc failed\n");
        //return -1;
        goto l_decode_exit;
    }

    eLIBs_memset(pbuffer->stream_buf_inf.buffer, 0, FRAME_BUFFER_SIZE);
    pbuffer->stream_buf_inf.buffer[0] = num[0];
    pbuffer->stream_buf_inf.buffer[1] = num[1];
    pbuffer->stream_buf_inf.buffer[2] = num[2];
    pbuffer->stream_buf_inf.buffer[3] = num[3];
    copy_size = eLIBs_fread(pbuffer->stream_buf_inf.buffer + 4, 1, buffer_len, fp_video);

    if (copy_size != buffer_len)
    {
        eLIBs_fclose(fp_video);
        fp_video = 0;
        esMEMS_Mfree(0, pbuffer->stream_buf_inf.buffer);
        esMEMS_Mfree(0, pbuffer);
        __wrn("copy_size!=buffer_len\n");
        //return 0;
        goto l_decode_exit;
    }

#endif
    eLIBs_memcpy(pbuffer->file_path, "XXX.H264", 9);
    pbuffer->tag_inf_validflag = 0;
    pbuffer->stream_buf_inf.buffer_len = buffer_len + 4;
    pbuffer->stream_buf_inf.frame_rate = 30;
    eLIBs_printf("file_path=%s,buffer=0x%x,len=0x%x\n", pbuffer->file_path, pbuffer->
    stream_buf_inf.buffer, pbuffer->stream_buf_inf.buffer_len);
    robin_set_mediafile(pbuffer);
    // robin_set_rotate_mode(rotate_mode);
```

```
    eLIBs_printf("before robin_set_play\n");
    robin_set_play();
    pbuffer_backup = pbuffer;
    eLIBs_printf("[22222222222222222222]\n");

    while (1)
    {
        ret = robin_query_buffer();

        if (ret <= 0)
        {
            __wrn(" **********robin_query_buffer failed**************\n");
            esKRNL_TimeDly(1);
            continue;
        }

        if (buffer_len > ret)
        {
            __wrn("buffer_len=%d > ret:%d\n", buffer_len, ret);
            __wrn(" **********robin_query_buffer failed**************\n");
        }

        if (!(count++ % 30))
        {
            __msg("ringbuffer freesize=0x%x(%d)\n", ret, ret); // printf  ringbuffer memory
     left size
            //uFreeMemSize = esMEMS_FreeMemSize();
            //__msg("left memory=0x%x(%d)\n",uFreeMemSize,uFreeMemSize);//printf system
    memory left size
        }

        if (ret  < FRAME_BUFFER_SIZE)
        {
            esKRNL_TimeDly(1);
            continue;
        }

        if (pbuffer->stream_buf_inf.buffer_len <= FRAME_BUFFER_SIZE)
        {
            ret = robin_write_buffer(pbuffer->stream_buf_inf.buffer_len, pbuffer->
    stream_buf_inf.buffer);
        }
        else
        {
            eLIBs_printf("frame lenth(%d) large then %d, skip this frame\n", buffer_len,
    FRAME_BUFFER_SIZE);
        }

        copy_size = eLIBs_fread(num, 1, 4, fp_video);

        if (copy_size != 4)
        {
            __log("**************copy size =%d**************\n", copy_size);
            //eLIBs_fclose(fp_video);
            //fp_video = 0;
            //esMEMS_Mfree(0, pbuffer->stream_buf_inf.buffer);
            //pbuffer->stream_buf_inf.buffer = 0;
            //esMEMS_Mfree(0, pbuffer);
            //pbuffer = 0;
            goto l_decode_exit;
```

```
            //return 0;
        }

        buffer_len = (num[0] & 0xff) | ((num[1] & 0xff) << 8) | ((num[2] & 0xff) << 16) |
    ((num[3] & 0xff) << 24);

        if (buffer_len > FRAME_BUFFER_SIZE)
        {
            eLIBs_printf(" frame lenth(%d) learge then %d\n", buffer_len, FRAME_BUFFER_SIZE
    );
            continue;
        }

        eLIBs_memset(pbuffer->stream_buf_inf.buffer, 0, FRAME_BUFFER_SIZE);
        pbuffer->stream_buf_inf.buffer[0] = num[0];
        pbuffer->stream_buf_inf.buffer[1] = num[1];
        pbuffer->stream_buf_inf.buffer[2] = num[2];
        pbuffer->stream_buf_inf.buffer[3] = num[3];
        copy_size = eLIBs_fread(pbuffer->stream_buf_inf.buffer + 4, 1, buffer_len, fp_video
    );

        if (copy_size != buffer_len)
        {
            __log("**************copy_size!=buffer_len**************\n");
            //eLIBs_fclose(fp_video);
            //fp_video = 0;
            //esMEMS_Mfree(0, pbuffer->stream_buf_inf.buffer);
            //pbuffer->stream_buf_inf.buffer = 0;
            //esMEMS_Mfree(0, pbuffer);
            //pbuffer = 0;
            //break;
            goto l_decode_exit;
            //return 0;
        }

        pbuffer->stream_buf_inf.buffer_len = buffer_len + 4;
        //eLIBs_printf("buffer:0x%x,0x%x,0x%x,0x%x.len=0x%x,copy_size:0x%x\n",pbuffer->
    stream_buf_inf.buffer[0],pbuffer->stream_buf_inf.buffer[1],
        //pbuffer->stream_buf_inf.buffer[2],pbuffer->stream_buf_inf.buffer[3],pbuffer->
    stream_buf_inf.buffer_len,copy_size);
    }

l_decode_exit:
    //esKRNL_TimeDly(200);
    eLIBs_printf("[YG3]\n");
    robin_set_stop();
    //robin_close();
    eLIBs_printf("[YG2]\n");

    if (pbuffer->stream_buf_inf.buffer)
    {
        esMEMS_Mfree(0, pbuffer->stream_buf_inf.buffer);
        pbuffer->stream_buf_inf.buffer = NULL;
    }

    eLIBs_printf("[YG434]\n");

    if (pbuffer)
    {
        esMEMS_Mfree(0, pbuffer);
```

```
        pbuffer = NULL;
    }

    eLIBs_printf("[YG9999999999999]\n");
    robin_set_cmd_stop();
    eLIBs_printf("[YG55555]\n");

    if (hvedio_lyr)
    {
        robin_release_video_layer((void *)hvedio_lyr);
        hvedio_lyr = NULL;
    }

    eLIBs_printf("[YG4]\n");

    if (hdisp)
    {
        eLIBs_fclose(hdisp);
        hdisp = NULL;
    }

    if (fp_video)
    {
        eLIBs_fclose(fp_video);
        fp_video = NULL;
    }

    eLIBs_printf("[YG1]\n");
    return 0;
}
```

具体流程参考 livedesk\beetles\sun20iw1_app\apps\play\ios_h264.c。

著作权声明

商标声明

免责声明