Goscam SDK接口协议

高斯贝尔智能家居有限公司

二○一四年十月

Revision record 修订记录

Date 日期	**Revision version** 修订版本	**CR ID / Defect ID** **CR号**	**Section Number** 修改章节	**Description** 描述	**Author** 作者
2014.10			初稿	初稿	
2018.06	V2.0		第5章、第8章	根据最新SDK修改部分描述	

目录

1 概述

1.1 说明

Goscam是一家专业的安防设备生产商，集研发、生产、销售于一体，旗下产品包括婴儿看护、家庭安防、运动DV等系列产品。本文档用于帮助客户将我们的IP系列产品接入客户的平台，客户只需要维护一个云平台，即可通过少许的工作将我们的IPC变成客户自有产品，实现终端+平台的整体解决方案，向终端用户提供可远程访问的高质量、性能丰富的IP摄像头。

本帮助文档的内容包括IPC SDK的系统架构、功能模块介绍、用户使用说明及接口的详细使用范例，用户基于此SDK包，不仅可以轻易的实现云协议对接，而且还可以进行一些功能扩展，实现一些个性化的功能开发。

1.2 开发所需资源

Goscam出品的IPC设备；
SDK开发包，包括库文件和相应头文件；
交叉编译工具链

1.3 术语表

术语	含义
SDK	Software Development Kit

2 SDK开发套件交付清单

1. 我们提供给客户的套件主要包括如下几个部分，任一部分均可以在以下链接下载获取：
   http://pan.baidu.com/s/1bn6PPjp?qq-pf-to=pcqq.c2c

2. Tree图

```
Goscam_IPC_SDK_Development_KITs/
|-- cross_compilation_tools
|   `-- amba-toolchain-2014.04.zip  //cross compiling tools
|-- documents
|   |-- goscam_burn_image_to_device.doc.pdf
|   `-- goscam_sdk_Interface_protocol_20150518.pdf
|-- examples
|-- goscam_ambarella_burn_imgae
|   |-- DirectUSB II-Setup.exe   //driver
|   |-- DirectUSB-tools.rar      //pc tools
|   `-- amboot_bld_hal_pba_kernel_lnx_release.bin  //default flash image
`-- sdk_srcs
    |-- compile.txt
    |-- include
    |   |-- sdk_commonstruct.h
    |   |-- sdk_define.h
    |   `-- sdkout_impl.h
    |-- lib
    |   |-- libGoscamDevSdk.a
    |   |-- libamrc.so
    |   |-- libamutils.so
    |   |-- libasound.so
    |   |-- libfreetype.so
    |   `-- liblbr.so
    `-- sample
        `-- sdk_main.c
```

3 如何开始工作

3.1 开发环境安装

如果有安霸芯片开发经验，可以略过。

1. 安装虚拟机或linux系统，要求系统Fedora 15~20，或Ubuntu 12.04及以后；
2. 安装交叉编译器。
   进入目录：…\Goscam_IPC_SDK_Development_KITs\cross_compilation_tools\amba-toolchain-2014.04，根据readme.txt，执行相应操作，执行完毕后，可在/usr/local下看到有交叉编译工具链的目录生成。

3.2 试用机器

1. 此时，您手头应该会有一台我们的IPC设备，当这台设备到您手头时，我们已经在里面烧录好程序，上电即可运行。设备端wifi默认处于AP模式状态，热点名称是一串16位的数

字，类似6320388800000070，您此时仅可以使用我们的手机APP(ios/android: ulife)直连IPC查看视频，以验证机器的硬件环境是OK的；您也可以通过扫描手机二维码，让设备获取SSID/PASSWD，连上指定的路由器，在局域网内搜索IPC。

2. 若机器运行不正常，或在您的开发过程中，因某些原因需要让机器恢复到初始状态，您可以重新烧写flash，方法参考：..\Goscam_IPC_SDK_Development_KITs\documents\goscam_burn_image_to_devic.pdf .

3.3 编程

1. 验证机器硬件环境无误后，接下来就是基于我们提供的..\Goscam_IPC_SDK_Development_KITs\sdk_srcs\sample\sdk_main.c 完成平台对接工作，包括音视频流的获取， 远程的信息交互， IPC 参数的动态控制等等。可参考..\Goscam_IPC_SDK_Development_KITs\examples下的例子。
2. 编译，生成可执行程序。编译方法请参考..\Goscam_IPC_SDK_Development_KITs\sdk_srcs\compile.txt。

3.4 调试与验证

1. 上一步生成可执行程序后，接下来是验证您工作的成果，首先，将可执行程序拷贝到IPCamera的文件系统的目录/opt/ipnc下，如何拷贝？可以考虑NFS挂载文件系统；
2. 将可执行程序重命名为GS_IPC，重新启动设备或在当前目录下执行 ./GS_IPC &。运行出错？不要慌，返回3.2，检查您的代码，重新编译，然后重复3.3.

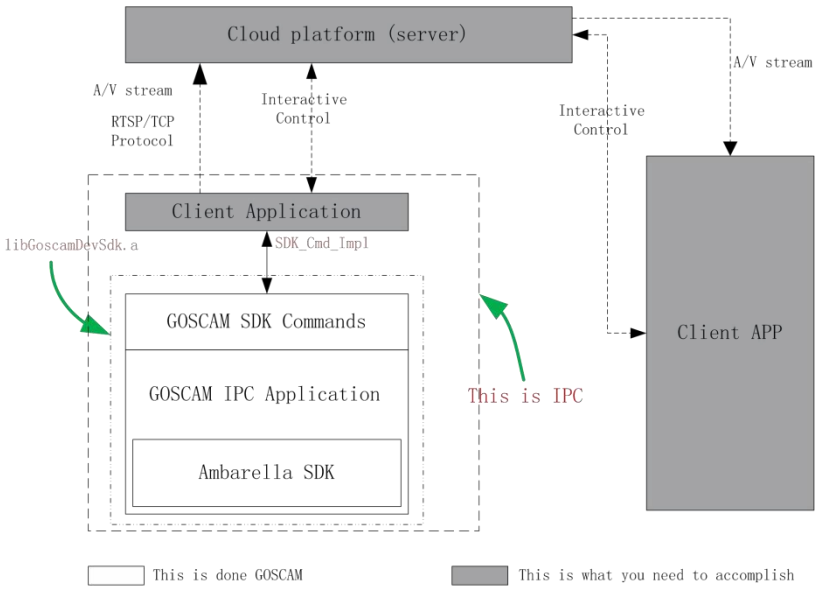
3.5 生产

通过您的努力，反复调试确认GS_IPC运行无误后，请将此文件发还给我们，我们将用它制作新的amboot_bld_hal_pba_kernel_lnx_release.bin镜像文件，批量烧录flash，生产成品机器。

至此，恭喜您，对接工作完成了！

4 系统架构图

设备底层驱动、功能及逻辑控制都已经在SDK中完成，客户仅需要通过命令方式与SDK交互，即可以实现音视频流数据的获取，完成客户云平台的对接后，远程APP与IPC可进行动态交互。SDK的调用关系图如下所示。



5 功能模块

SDK依据功能划分成若干个模块，包括系统模块，视频模块，音频模块，sensor模块，报警模块，OSD模块，录像模块，抓拍模块，升级模块，网络模块，云台模块，调试打印模块，对应不同模块有各自命令接口。接下来的内容是对每个命令的使用条件、参数进行介绍，并对关键命令给出使用范例。

客户在使用各模块命令时，都是用的统一调用接口：int SDK_Cmd_Impl(CMD，Param)。

5.1 系统模块

此模块完成系统初始化，系统启动等操作，利用这些命令，可以将GOSCAM IPC顺利的跑起来，但是没有与云端交互。系统模块包括以下命令：

- SDKCMD_SYS_INIT：初始化IPC系统
- SDKCMD_SYS_RUN：运行IPC系统
- SDKCMD_SYS_EXIT：退出IPC系统
- SDKCMD_GET_DEVICE_INFO： 获取设备型信息
- SDKCMD_SAVE_ALL_PARAM：保存全部参数
- SDKCMD_REBOOT_DEVICE：IPC重启
- SDKCMD_RESET_DEVICE：IPC重置
- SDKCMD_SET_DEVICE_TIME：设置IPC时间
- SDKCMD_SET_NIGHT_VISION：设置夜视相关参数
- SDKCMD_GET_NIGHT_VISION：获取夜视相关参数
- SDKCMD_SET_CONNECTED_PLATFORM_STATUS：设置登录平台状态
- SDKCMD_SET_LED_STATUS：设置LED状态
- SDKCMD_SET_DEVICE_AUTHENTICATION：设置置IPC接入TUTK平台的鉴权信息
- SDKCMD_GET_DEVICE_AUTHENTICATION：获取IPC接入TUTK平台的鉴权信息

- SDKCMD_GET_DEVICE_ABILITY：获取IPC能力集
- SDKCMD_SET_LIGHT_SWITCH：设置灯开关
- SDKCMD_GET_LIGHT_SWITCH：获取灯开关
- SDKCMD_SET_LIGHT_DURATION：设置灯照时长，包括手动开灯时长、触发亮灯时长
- SDKCMD_GET_LIGHT_DURATION：获取灯照时长，包括手动开灯时长、触发亮灯时长
- SDKCMD_SET_LIGHT_TIMING_INFO：设置定时亮灯时间点(晚上开灯、早晨灭灯时间点，以及一周生效的天数)
- SDKCMD_GET_LIGHT_TIMING_INFO：获取定时亮灯时间点(晚上开灯、早晨灭灯时间点，以及一周生效的天数)

5.1.1 SDKCMD_SYS_INIT

【描述】

必须调用。

【参数】

无。

【返回值】

返回值	描述
0	成功
非0	失败，其值为错误代码

【需求】

- 头文件
- 库文件

【范例】

```
            case SDKCMD_SYS_INIT:
            {
                retcode = SDK_sysInit();
                if(retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\nSDK_sysInit Error,
    ERRCODE: %d\n", retcode);
                    return retcode;
                }
                break;
            }
```

【相关命令】

- SDKCMD_SYS_RUN
- SDKCMD_SYS_EXIT

5.1.2 SDKCMD_SYS_RUN

【描述】

必须调用，且需在调用SDKCMD_SYS_INIT命令之后。

【参数】
无。

【返回值】

返回值	描述
0	成功
非0	失败，其值为错误代码

【范例】

```
            case SDKCMD_SYS_RUN:
            {
                retcode = SDK_sysRun();
                if(retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\nSDK_sysRun Error,
    ERRCODE: %d\n", retcode);
                    return retcode;
                }
                break;
            }
```

5.1.3 SDKCMD_SYS_EXIT

【描述】

退出系统运行，回收资源。

【参数】
无。

【返回值】

返回值	描述
0	成功
非0	失败，其值为错误代码

【范例】

```
            case SDKCMD_SYS_EXIT:
            {
                retcode = SDK_sysExit();
                if(retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\nSDK_sysExit Error,
    ERRCODE: %d\n", retcode);
                    return retcode;
                }
                break;
            }
```

5.1.4 SDKCMD_GET_DEVICE_INFO

【描述】

非必须调用，获取设备型信息。

【参数】

```
typedef struct
{
char a_name[64];
char a_type[64];
char a_software_version[64];
char a_hardware_version[64];
char a_id[64];
char a_wifi_mac[64];
char a_line_mac[64];
}T_SDK_DEVICE_INFO;
```

【范例】

```
            case SDKCMD_GET_DEVICE_INFO://获取设备型信息
            {
                T_SDK_DEVICE_INFO *pt_deviceInfo = (T_SDK_DEVICE_INFO*)param;
                retcode = SDK_GetDeviceInfo(pt_deviceInfo);
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\nDEVICE TYPE GET FAILED!\n");
                    return SDK_FAILUR;
                }
                break;
            }
```

5.1.5 SDKCMD_SAVE_ALL_PARAM

【描述】

保存全部参数。

【参数】

无。

【范例】

```
            case SDKCMD_SAVE_ALL_PARAM:
            {
                retcode = SDK_SaveAllParam();
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\nDEVICE SAVE ALL PARAM
    FAILED!\n");
                    return SDK_FAILUR;
                }
                break;
            }
```

5.1.6 SDKCMD_REBOOT_DEVICE

【描述】

IPC重启。

【参数】

无。

【范例】

```
            case SDKCMD_REBOOT_DEVICE://设备重启
            {
                 Gos_Man_De_St.reboot_flag = 1;
                 return SDK_SUCESS;
                break;
            }
```

5.1.7 SDKCMD_RESET_DEVICE

【描述】

对设备执行复位操作，恢复出厂默认值。

【参数】

无。

【范例】

```
             case SDKCMD_RESET_DEVICE:
             {
                 retcode = SDK_ResetDevice();
                 if( SDK_SUCESS != retcode)
                 {
                     Dbg_Trace(GOS_LOG_ERR, "\nDEVICE SET RESET
    FAILED!\n");
                     return SDK_FAILUR;
                 }
                 break;
             }
```

5.1.8 SDKCMD_SET_DEVICE_TIME

【描述】

设置IPC时间。

【参数】

```
typedef struct
{
 unsigned int un_year;  //value range  (1970~2055)
 unsigned int un_month; //value range (1~12)
 unsigned int un_day;   //value range (1~31)
 unsigned int un_hour;  //value range(0~23)
 unsigned int un_minute;      //value range (0~59)
 unsigned int un_second;      //value range (0~59)
}T_SDK_DEVICE_TIME;
```

【范例】

```
             case SDKCMD_SET_DEVICE_TIME:
             {
                  T_SDK_DEVICE_TIME *param_output =
    (T_SDK_DEVICE_TIME*)param;
                 retcode = SDK_SetDeviceTime(param_output);
                 if(0 != retcode)
```

```
                return retcode;
               break;
           }
```

5.1.9 SDKCMD_SET_NIGHT_VISION

【描述】

设置夜视相关参数。

【参数】

```
typedef struct
{
 unsigned int un_auto;      //value range (0:off 1:on)
 unsigned int un_day_night;   //value range(0:day 1:night)
}T_SDK_NIGHT_VISION;
```

【范例】

```
            case SDKCMD SET NIGHT VISION:
            {
                T_SDK_NIGHT_VISION *pData = (T_SDK_NIGHT_VISION*)param;
                retcode = SDK_SetNightVision(pData);
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\nSet Night Vision FAILED!\n");
                    return SDK_FAILUR;
                }
                 break;
            }
```

5.1.10 SDKCMD_GET_NIGHT_VISION

【描述】

获取夜视相关参数。

【参数】

```
typedef struct
{
 unsigned int un_auto;      //value range (0:off 1:on)
 unsigned int un_day_night;   //value range(0:day 1:night)
}T_SDK_NIGHT_VISION;
```

【范例】

```
            case SDKCMD_GET_NIGHT_VISION:
            {
                retcode = SDK_GetNightVision((T_SDK_NIGHT_VISION *)param);
                if( SDK_SUCESS != retcode)
                {
                   Dbg_Trace(GOS_LOG_ERR, "\n SDK_GetNightVision
            FAILED!\n");
                   return SDK_FAILUR;
                 }
                break;
             }
```

5.1.11 SDKCMD_SET_CONNECTED_PLATFORM_STATUS

【描述】

设置登录平台状态。

【参数】

param。

【范例】

```
            case SDKCMD_SET_CONNECTED_PLATFORM_STATUS:
            {
                retcode = SDK_ConnectedPlatform((unsigned int *)param);
                if(SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "SDK_ConnectedPlatform
    FAILED!\n");
                    return SDK_FAILUR;
                }
                break;
            }
```

5.1.12 SDKCMD_SET_LED_STATUS

【描述】

设置LED状态.

【参数】

```
typedef struct
{
 unsigned int un_gpio_0;
 unsigned int un_gpio_1;
 unsigned int un_value;
}T_SDK_LED;
```

【范例】

```
            case SDKCMD_SET_LED_STATUS:
            {
                retcode = SDK_SetGPIO((T_SDK_LED *)param);
                if(SDK_SUCESS != retcode)
                {
                    Dbg Trace(GOS LOG ERR, "SDKCMD SET LED STATUS
    FAILED!\n");
                    return SDK_FAILUR;
                }
                break;

            }
```

5.1.13 SDKCMD_SET_DEVICE_AUTHENTICATION

【描述】

设置用户策略。

【参数】

```
typedef struct
{
 int  index;   //输入参数0 ,1 ,2
 char user_name[64];
 char passwd[64];
 char reserved[4];
}T_SDK_DEVICE_AUTHENTICATION_INFO;
```

【范例】

```
            case SDKCMD_SET_DEVICE_AUTHENTICATION:
            {
                   T_SDK_DEVICE_AUTHENTICATION_INFO *set_param =
            (T_SDK_DEVICE_AUTHENTICATION_INFO*)param;
                   retcode = SDK_Set_Device_Authentication(set_param);

                   if(SDK_SUCESS != retcode)
                   {
                          Dbg_Trace(GOS_LOG_ERR,
               "SDK_Set_Device_Authentication FAILED!\n");
                          return SDK_FAILUR;
                   }
                   break;
            }
```

【注意事项】

无。

5.1.14 SDKCMD_GET_DEVICE_AUTHENTICATION

【描述】

获取用户策略。

【参数】

```
typedef struct
{
  int  index;                                    //输入参数0 ,1 ,2
  char user_name[64];
  char passwd[64];
  char reserved[4];
}T_SDK_DEVICE_AUTHENTICATION_INFO;
```

【范例】

```
             case SDKCMD_GET_DEVICE_AUTHENTICATION:
            {
                T_SDK_DEVICE_AUTHENTICATION_INFO *get_param =
    (T_SDK_DEVICE_AUTHENTICATION_INFO*)param;
               retcode = SDK_Get_Device_Authentication(get_param);
               if(SDK_SUCESS != retcode)
               {
                   Dbg_Trace(GOS_LOG_ERR, "SDK_Get_Device_Authentication
    FAILED!\n");
                   return SDK_FAILUR;
               }
               break;
            }
```

5.1.15 SDKCMD_GET_DEVICE_ABILITY

【描述】

获取IPC能力集。

【参数】

```
//IPC 能力集结构体，主要用于给APP端提供隐藏或显示相关件UI 的依据
```

```
typedef struct
{
 unsigned int  encrypted_ic_flag;   //是否有加密IC
 unsigned int  pir_flag;                  //是否有PIR传感器，0:无，1:有，
下同
 unsigned int  ptz_flag;                  //是否有云台
 unsigned int  mic_flag;                  //是否有咪头
 unsigned int  speaker_flag;        //是否有喇叭
 unsigned int  sd_flag;                   //是否有SD卡
 unsigned int  temperature_flag;    //是否有温感探头
 unsigned int  timezone_flag;       //是否支持同步时区
 unsigned int  night_vison_flag;          //是否支持夜视
 unsigned int  resolution_0_flag;   //主码流分辨率大小 Width:高16位 Height:
低16位  Ming@2016.06.14
 unsigned int  resolution_1_flag;   //子码流分辨率大小 Width:高16位 Height:
低16位  Ming@2016.06.14
 unsigned int  reserver[8];
}T_SDK_DEVICE_ABILITY_INFO;

//IPC 能力集结构体，主要用于给APP端提供隐藏或显示相关件UI 的依据
typedef struct
{
 unsigned int   c_device_type; //设备类型    900中性版      101彩益
100海尔     901高世安
 unsigned int   un_resolution_0_flag;     //主码流分辨率大小 Width:高16位
Height:低16位  Ming@2016.06.14
 unsigned int   un_resolution_1_flag;     //子码流
 unsigned int   un_resolution_2_flag;     //第3路码流
 unsigned char  c_encrypted_ic_flag;      //是否有加密IC
 unsigned char  c_pir_flag;               //是否有PIR传感器，0:无，1:有，
下同
 unsigned char  c_ptz_flag;               //是否有云台
 unsigned char  c_mic_flag;               //是否有咪头
 unsigned char  c_speaker_flag;           //是否有喇叭
 unsigned char  c_sd_flag;                //是否有SD卡
 unsigned char  c_temperature_flag;       //是否有温感探头
 unsigned char  c_timezone_flag;          //是否支持同步时区
 unsigned char  c_night_vison_flag;//是否支持夜视

 unsigned char  ethernet_flag;      //是否带网卡0:wifi 1有线2wifi加有线
 unsigned char  c_smart_connect_flag;     //是否支持smart扫描0代表不支持，
1代表7601smart  2代表8188smart
 unsigned char  c_motion_detection_flag; //是否支持移动侦测
 unsigned char  c_record_duration_flag;
}T_SDK_DEVICE_ABILITY_INFO1;

//IPC 能力集结构体，主要用于给APP端提供隐藏或显示相关件UI 的依据
typedef struct
{
 unsigned int   c_device_type; //设备类型    900中性版      101彩益
100海尔     901高世安      200NVR设备  180 VR设备(180)  300 360全景摄像机
 unsigned int   un_resolution_0_flag;     //主码流分辨率大小 Width:高16位
Height:低16位  Ming@2016.06.14
 unsigned int   un_resolution_1_flag;     //子码流
 unsigned int   un_resolution_2_flag;     //第3路码流
 unsigned char  c_encrypted_ic_flag;      //是否有加密IC  //是否支持硬解绑
```

```
 unsigned char  c_pir_flag;                //是否有PIR传感器，0:无，1:有，
下同
 unsigned char  c_ptz_flag;                //是否有云台
 unsigned char  c_mic_flag;                //是否有咪头

 unsigned char  c_speaker_flag;            //是否有喇叭
 unsigned char  c_sd_flag;                 //是否有SD卡
 unsigned char  c_temperature_flag;        //是否有温感探头
 unsigned char  c_timezone_flag;           //是否支持同步时区

 unsigned char  c_night_vison_flag; //是否支持夜视
 unsigned char  ethernet_flag;      //是否带0     代表不支持，
                                           1     代表7601smart
                                           2     代表8188smart
                                           3     代表ap6212
                                           9     不支持二维码扫描
                                           10    只支持二维码扫描
                                           11代表二维码扫描+7601smart
                                           12代表二维码扫描+8188smart
                                           13代表二维码扫描+ap6212smart
                                           14代表AP添加
                                           15代表AP添加+8188smart
                                                  */
网卡0:wifi 1有线2wifi加有线
 unsigned char  c_smart_connect_flag;     /* 是否支持smart扫描


 unsigned char  c_motion_detection_flag; //是否支持移动侦测

 unsigned char  c_record_duration_flag; // 是否有设置录像录像时长
 unsigned char  c_light_flag; // 是否有设置照明灯开关
 unsigned char  c_audio_alarm_detection_flag; //是否支持声音侦测报警
 unsigned char  align1;  // 是否支持摇篮曲 0.不支持 1.5886HAB 2.GD8202KE
3.VOXX系列
 /*
 reserver_default_off[0]       // 是否带电池 0. 无 ，1. 有
 reserver_default_off[1]       // 是否支持WIFI远程唤醒 0.不支持， 1支持
 reserver_default_off[2]  // 是否支持状态灯开关 0.不支持， 1支持

 reserver_default_off[3]  // 是否支持摄像头开关 0.不支持， 1支持
 reserver_default_off[4]  // 是否支持摄像头麦克风开关 0.不支持， 1支持
 reserver_default_off[5]  // 是否支持云存储 0.不支持， 1支持
 reserver_default_off[6]       // 是否支持打开流鉴权

 reserver_default_off[7]       // 是否支持Alexa_Voice_Service
 reserver_default_off[8]       // 是否支持Alexa_Skills_Kit
 reserver_default_off[9]      // 是否支持湿度
 reserver_default_off[10]     // 是否支持WBGT
 */
 unsigned char  reserver_default_off[32]; // 预留能力集默认关闭
 unsigned char  reserver_default_on[32]; // 预留能力集默认开启
}T_SDK_DEVICE_ABILITY_INFO2;
```

【范例】

```
            case SDKCMD_GET_DEVICE_ABILITY:
            {
                  T_SDK_DEVICE_ABILITY_INFO *get_param = NULL ;
                  T_SDK_DEVICE_ABILITY_INFO1 *get_param1 = NULL ;
                  T_SDK_DEVICE_ABILITY_INFO2 *get_param2 = NULL ;
                  switch(SDK_DEVICE_ABILITY_VERSION)
                  {
                        case 0:
                              get_param =
(T_SDK_DEVICE_ABILITY_INFO*)param;
                              retcode = SDK_Get_Device_Ability(get_param);

                              if(SDK_SUCESS != retcode)
                              {
                                    Dbg_Trace(GOS_LOG_ERR,
"SDK_Get_Device_Authentication ability0 FAILED!\n");
                                    return SDK_FAILUR;
                              }
                              break;
                        case 1:
                              get_param1 =
(T_SDK_DEVICE_ABILITY_INFO1*)param;
                              retcode =
SDK_Get_Device_Ability1(get_param1);
                              if(SDK_SUCESS != retcode)
                              {
                                    Dbg_Trace(GOS_LOG_ERR,
"SDK_Get_Device_Authentication ability1 FAILED!\n");
                                    return SDK_FAILUR;
                              }
                              break;
                        case 2:
                              get_param2 =
(T_SDK_DEVICE_ABILITY_INFO2*)param;
                              retcode =
SDK_Get_Device_Ability2(get_param2);
                              if(SDK_SUCESS != retcode)
                              {
                                    Dbg_Trace(GOS_LOG_ERR,
"SDK_Get_Device_Authentication ability2 FAILED!\n");
                                    return SDK_FAILUR;
                              }
                              break;
                              default:
                                    Dbg_Trace(GOS_LOG_ERR, "not support
this ability search version!\n");
                                    break;
                  }
```

5.1.16 SDKCMD_SET_LIGHT_SWITCH

【描述】

设置灯开关。

【参数】

param_input。

【范例】

```
            case SDKCMD_SET_LIGHT_SWITCH:
            {
                  unsigned int *param_input = (unsigned int*)param;
                  retcode = SDK_Set_Light_Switch(*param_input);
```

```
                    if(0 != retcode)
                          return retcode;
                    break;
              }
```

5.1.17 SDKCMD_GET_LIGHT_SWITCH

【描述】

获取灯开关。

【参数】

param_output。

【范例】

```
              case SDKCMD_GET_LIGHT_SWITCH:
              {
                    unsigned int *param_output = (unsigned int*)param;
                    retcode = SDK_Get_Light_Switch(param_output);
                    if(0 != retcode)
                          return retcode;
                    break;
              }
```

5.1.18 SDKCMD_SET_LIGHT_DURATION

【描述】

设置灯照时长，包括手动开灯时长、触发亮灯时长。

【参数】

```
typedef struct

{

    unsigned int  un_trigger_time; //触发亮灯时间 pir / motion

    unsigned int  un_manual_time;  //手动开灯亮灯时间

 int   n_reserve;

}T_SDK_DEVICE_LIGHT_DURATION;
```

【范例】

```
              case SDKCMD_SET_LIGHT_DURATION:
              {
                    T_SDK_DEVICE_LIGHT_DURATION* param_input =
(T_SDK_DEVICE_LIGHT_DURATION*)param;
                    retcode = SDK_Set_Light_Open_Time(param_input-
>un_trigger_time,param_input->un_manual_time);
                    if(0 != retcode)
                          return retcode;
                    break;
              }
```

5.1.19 SDKCMD_GET_LIGHT_DURATION

【描述】

获取灯照时长，包括手动开灯时长、触发亮灯时长。

【参数】

```
typedef struct
{
    unsigned int  un_trigger_time; //触发亮灯时间 pir / motion
    unsigned int  un_manual_time;  //手动开灯亮灯时间
 int   n_reserve;
}T_SDK_DEVICE_LIGHT_DURATION;
```

【范例】

```
            case SDKCMD_GET_LIGHT_DURATION:
            {
                  T_SDK_DEVICE_LIGHT_DURATION* param_output =
(T_SDK_DEVICE_LIGHT_DURATION*)param;
                  retcode = SDK_Get_Light_Open_Time(&(param_output-
>un_trigger_time),&(param_output->un_manual_time));
                  if(0 != retcode)
                        return retcode;
                  break;
            }
```

5.1.20 SDKCMD_SET_LIGHT_TIMING_INFO

【描述】

设置定时亮灯时间点(晚上开灯、早晨灭灯时间点，以及一周生效的天数)。

【参数】

```
typedef struct

{

    unsigned int  un_on_hour;

    unsigned int  un_on_min;

    unsigned int  un_off_hour;

    unsigned int  un_off_min;

 unsigned int     un_wday_switch;  //按 0~6位表示，第0位表示星期天，第1位表
示星期一 0->关闭 1->打开

 int   n_reserve;

}T_SDK_DEVICE_LIGHT_TIMING;//定时开灯时间点，开关
```

【范例】

```
            case SDKCMD_SET_LIGHT_TIMING_INFO:
            {
                  T_SDK_DEVICE_LIGHT_TIMING *param_input =
(T_SDK_DEVICE_LIGHT_TIMING *)param;

                  retcode = SDK_Set_Light_Timing_Info(param_input);
                  if(0 != retcode)
                        return retcode;
                  break;
            }
```

5.1.21 SDKCMD_GET_LIGHT_TIMING_INFO

【描述】

获取定时亮灯时间点(晚上开灯、早晨灭灯时间点，以及一周生效的天数)。

【参数】

```
typedef struct
{
    unsigned int  un_on_hour;
    unsigned int  un_on_min;
    unsigned int  un_off_hour;
    unsigned int  un_off_min;
 unsigned int     un_wday_switch;   //按 0~6位表示，第0位表示星期天，第1位表示星期一 0->关闭 1->打开
 int   n_reserve;
}T_SDK_DEVICE_LIGHT_TIMING;//定时开灯时间点，开关
```

【范例】

```
            case SDKCMD_GET_LIGHT_TIMING_INFO:
            {
                  T_SDK_DEVICE_LIGHT_TIMING *param_output =
(T_SDK_DEVICE_LIGHT_TIMING *)param;

                  retcode = SDK_Get_Light_Timing_Info(param_output);
                  if(0 != retcode)
                        return retcode;
                  break;
            }
```

5.2 视频命令

此模块用于控制设置视频以及相关的参数，视频模块包括以下命令：

- SDKCMD_SET_VIDEO_ENCODE_DEFAULT_PARAM：设置视频默认参数
- SDKCMD_REGISTER_STREAM_DATA_CALLBACK：视频码流数据获取的回调函数注册
- SDKCMD_FORCE_VIDEO_ENCODE_I_FRAME：强制获取I帧
- SDKCMD_SET_VIDEO_ENCODE_SWITCH： 设置视频编码开关
- SDKCMD_SET_VIDEO_ENCODE_RESOLUTION：设置视频编码分辨率
- SDKCMD_SET_VIDEO_ENCODE_I_FRAME_INTERVAL：设置编码GOP
- SDKCMD_SET_VIDEO_ENCODE_BITRATE：设置视频编码码率
- SDKCMD_SET_VIDEO_ENCODE_FRAMERATE：设置视频帧率
- SDKCMD_SET_VIDEO_ENCODE_QUALITY：设置视频图像质量
- SDKCMD_SET_VIDEO_ENCODE_LEVEL：设置视频编码等级
- SDKCMD_SET_VIDEO_ENCODE_QP：设置视频编码qp等级
- SDKCMD_GET_VIDEO_ENCODE_PARAM：获取当前视频编码参数
- SDKCMD_SET_VIDEO_ENCODE_MIRROR：设置翻转镜像的值
- SDKCMD_GET_VIDEO_ENCODE_MIRROR：获取翻转镜像的值
- SDKCMD_GET_VIDEO_ENCODE_QUALITY：获取当前视频质量

- SDKCMD_SET_CURR_STREAM_QUALITY：设置当前码流质量
- SDKCMD_GET_CURR_STREAM_QUALITY：获取当前码流质量
- SDKCMD_SET_SNAPSHOT_QUALITY：设置当前图像抓拍质量
- SDKCMD_GET_SNAPSHOT_QUALITY：获取当前图像抓拍质量

5.2.1 设置视频默认参数

命令：SDKCMD_SET_VIDEO_ENCODE_DEFAULT_PARAM

参数：无。

使用范例：

```
            case SDKCMD_SET_VIDEO_ENCODE_DEFAULT_PARAM:
             {
                 retcode = GOS_SDK_SetVideoDefaultParm();
                 if(0 != retcode)
                     return retcode;
                 break;
             }
```

注意事项：无。

5.2.2 视频码流数据获取的回调函数注册

命令：SDKCMD_REGISTER_STREAM_DATA_CALLBACK

参数：

```
typedef int (*F_SDK_Stream_Callback)(T_SDK_STREAM_CALLBACK_PARAM
*pt_encode_data);

typedef struct
{
 F_SDK_Stream_Callback   fp_callback;
 unsigned int un_video_switch;
 unsigned int un_audio_switch;
 unsigned int un_video_channel;
 unsigned int un_audio_channel;
 int   n_reserve;
}T_CALLBACK_INFO;

typedef struct
{
 T_CALLBACK_INFO   ta_callback_info[4];
 int   n_reserve;
}T_SDK_STREAM_REGISTER_CALLBACK;
```

使用范例：

```
            case SDKCMD_REGISTER_STREAM_DATA_CALLBACK:
            {
                T_SDK_STREAM_REGISTER_CALLBACK* param_input =
    (T_SDK_STREAM_REGISTER_CALLBACK *)param;
            retcode = sdk_RigisterStreamCallback(param_input);
```

```
                if(0 != retcode)
                     return retcode;
                break;
            }
```

注意事项：无。

5.2.3 强制获取I帧

命令：SDKCMD_FORCE_VIDEO_ENCODE_I_FRAME

参数：

```
typedef struct
{
  unsigned int     un_encode_channel_id;
  unsigned int     un_force_num;
}T_SDK_FORCE_I_FARME;
```

使用范例：

```
            case SDKCMD_FORCE_VIDEO_ENCODE_I_FRAME://强制获取I帧
            {
                   T_SDK_FORCE_I_FARME
            *param_input =
            (T_SDK_FORCE_I_FARME*)param;
                retcode = GOS_SDK_VENC_RequestIFrame(0,param_input-
    >un_encode_channel_id,param_input->un_force_num);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

注意事项：无。

5.2.4 设置视频编码开关

命令：SDKCMD_SET_VIDEO_ENCODE_SWITCH

参数：

```
typedef enum  _E_VIDEO_ENCODER_TYPE
{
  E_VENC_NONE        = 0x00,
  E_VENC_H264        = 0x01,
  E_VENC_MPEG4       = 0x02,
  E_VENC_MJPEG       = 0x03,
  E_VENC_JPEG        = 0x04,
}E_SDK_VIDEO_ENCODER_TYPE;
```

```
typedef struct
```

{

unsigned int un_encode_channel_id;

E_SDK_VIDEO_ENCODER_TYPE e_type;

}T_SDK_VIDEO_ENCODE_TYPE;

使用范例：

```
            case SDKCMD_SET_VIDEO_ENCODE_SWITCH://设置编码算法
            {
                T_SDK_VIDEO_ENCODE_TYPE *param_input =
    (T_SDK_VIDEO_ENCODE_TYPE*)param;
                retcode = GOS_SDK_VENC_SetVENCParam_EncType(param_input-
    >un_encode_channel_id, param_input->e_type);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

5.2.5 设置视频编码分辨率

命令：SDKCMD_SET_VIDEO_ENCODE_RESOLUTION

参数：

typedef struct

{

unsigned int un_encode_channel_id;

unsigned int un_width;

unsigned int un_height;

}T_SDK_VIDEO_ENCODE_RESOLUTION;

使用范例：

```
            case SDKCMD_SET_VIDEO_ENCODE_RESOLUTION://设置编码分辨率
            {
                T_SDK_VIDEO_ENCODE_RESOLUTION *param_input =
    (T_SDK_VIDEO_ENCODE_RESOLUTION*)param;
                retcode =
    GOS_SDK_VENC_SetVENCParam_Resolution(param_input-
    >un_encode_channel_id, param_input->un_width,param_input->un_height);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

注意事项：无。

5.2.6 设置编码GOP

命令：SDKCMD_SET_VIDEO_ENCODE_I_FRAME_INTERVAL

参数：

```
typedef struct
{
    unsigned int  un_encode_channel_id;
    unsigned int  un_interval;
```

```
}T_SDK_VIDEO_ENCODE_I_FRAME_INTERVAL;
```

使用范例：

```
            case SDKCMD_SET_VIDEO_ENCODE_I_FRAME_INTERVAL://设置编码GOP
            {
                T_SDK_VIDEO_ENCODE_I_FRAME_INTERVAL *param_input =
    (T_SDK_VIDEO_ENCODE_I_FRAME_INTERVAL*)param;
                retcode =
    GOS_SDK_VENC_SetVENCParam_KeyInterval(param_input-
    >un_encode_channel_id, param_input->un_interval);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

注意事项：无。

5.2.7 设置视频编码码率

命令：SDKCMD_SET_VIDEO_ENCODE_BITRATE

参数：

```
typedef struct
{
 unsigned int      un_encode_channel_id;
 unsigned int      un_bitrate_type; //value range (0: CBR; 1: VBR; 2:
CBR keep quality; 3: VBR keep quality)
 unsigned int      un_average_bitrate;
 unsigned int      un_max_bitrate;
 unsigned int      un_min_bitrate;
}T_SDK_VIDEO_ENCODE_BITRATE;
```

使用范例：

```
            case SDKCMD_SET_VIDEO_ENCODE_BITRATE://设置编码码率
            {
                T_SDK_VIDEO_ENCODE_BITRATE *param_input =
                   (T_SDK_VIDEO_ENCODE_BITRATE*)param;
                retcode = GOS_SDK_VENC_SetVENCParam_Bitrate(param_input);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

注意事项：无

5.2.8 设置视频帧率

命令：SDKCMD_SET_VIDEO_ENCODE_FRAMERATE

参数：

```
typedef struct
{
 unsigned int    un_encode_channel_id;
 unsigned int    un_framerate;
}T_SDK_VIDEO_ENCODE_FRAMERATE;
```

使用范例：

```
            case SDKCMD_SET_VIDEO_ENCODE_FRAMERATE://设置编码帧率
            {
                T_SDK_VIDEO_ENCODE_FRAMERATE *param_input =
    (T_SDK_VIDEO_ENCODE_FRAMERATE*)param;
                retcode = GOS_SDK_VENC_SetVENCParam_FrameRate(param_input-
    >un_encode_channel_id, param_input->un_framerate);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

注意事项：无。

5.2.9 设置视频图像质量

命令：SDKCMD_SET_VIDEO_ENCODE_QUALITY

参数：

```
typedef struct
{
  unsigned int     un_encode_channel_id;
  unsigned int     un_quality;       //value range(0-5)
}T_SDK_VIDEO_ENCODE_QUALITY;
```

使用范例：

```
            case SDKCMD_SET_VIDEO_ENCODE_QUALITY://设置编码图像质量
            {
                T_SDK_VIDEO_ENCODE_QUALITY *param_input =
    (T_SDK_VIDEO_ENCODE_QUALITY*)param;
                retcode = GOS_SDK_VENC_SetVENCParam_Qulity(param_input-
    >un_encode_channel_id, param_input->un_quality);
                if(0 != retcode)
                    return retcode;
                break;
            }               resp.result = -1;
                }
```

注意事项：无。

5.2.10 设置视频编码等级

命令：SDKCMD_SET_VIDEO_ENCODE_LEVEL

参数：

```
typedef struct
{
  unsigned int     un_encode_channel_id;
  unsigned int     un_profile;       //value range of Encode level ( 0: baseline; 1:MP; 2:HP)
HiSilicon not permitted set it
}T_SDK_VIDEO_ENCODE_LEVEL;
```

使用范例：

```
            case SDKCMD_SET_VIDEO_ENCODE_LEVEL: //编码等级 0: baseline;
    1:MP; 2:HP    POE暂不允许修改
            {
                T_SDK_VIDEO_ENCODE_LEVEL *param_input =
    (T_SDK_VIDEO_ENCODE_LEVEL*)param;
                retcode = GOS_SDK_VENC_SetVENCParam_Profile(param_input-
    >un_encode_channel_id, param_input->un_profile);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

注意事项：无。

5.2.11 设置视频编码qp等级

命令：SDKCMD_SET_VIDEO_ENCODE_QP

参数：

```
typedef struct
{
  unsigned int     un_encode_channel_id;
  unsigned int     un_I_frame_max_Qp;
  unsigned int     un_I_frame_min_Qp;
  unsigned int     un_P_frame_max_Qp;
  unsigned int     un_P_frame_min_Qp;
}T_SDK_VIDEO_ENCODE_QP;
```

使用范例：

```
                case SDKCMD_SET_VIDEO_ENCODE_QP://设置编码qp等级
            {
                T_SDK_VIDEO_ENCODE_QP *param_input =
    (T_SDK_VIDEO_ENCODE_QP*)param;
                retcode = GOS_SDK_VENC_SetVENCParam_Qp(param_input-
    >un_encode_channel_id, param_input->un_I_frame_max_Qp,param_input-
    >un_I_frame_min_Qp);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

注意事项：无。

5.2.12 获取当前视频编码参数

命令：SDKCMD_GET_VIDEO_ENCODE_PARAM

参数：

```
typedef struct
{
  unsigned int     un_encode_channel_id;          //(input Param)
  unsigned int     un_encode_switch;      //value range(0:off 1:on)
  E_SDK_VIDEO_ENCODER_TYPE     e_encode_type;
```

```
unsigned int     un_width;
unsigned int     un_height;
unsigned int     un_I_frame_interval;
T_SDK_VIDEO_ENCODE_BITRATE  t_bitrate;
unsigned int     un_framerate;
T_SDK_LOWER_BITRATE_CONTORL          t_lower_bitrate_control;          //only for
Ambarella
unsigned int     un_quality;       //value range(0-5)
unsigned int     un_profile;       //value range of Encode level ( 0: baseline; 1:MP; 2:HP)
HiSilicon not permitted set it
T_SDK_VIDEO_ENCODE_QP t_QP;
unsigned int     un_mirro_type; //value range
0:none,1:horizontal,2:vertical,3:horizonta+vertical
} T_SDK_VIDEO_ENCODE_PARAM;
```

使用范例：

```
            case SDKCMD_GET_VIDEO_ENCODE_PARAM://获取编码视频参数
            {
                T_SDK_VIDEO_ENCODE_PARAM *param_inout_parm =
    (T_SDK_VIDEO_ENCODE_PARAM*)param;
                retcode = GOS_SDK_Get_VENCParam(param_inout_parm-
    >un_encode_channel_id, param_inout_parm);
                if(0 != retcode)
                    return retcode;
                retcode = GOS_SDK_Get_ViMirrorMode(param_inout_parm-
    >un_encode_channel_id, (unsigned int *)&param_inout_parm-
    >un_mirro_type);
                if(0 != retcode)
                    return retcode;
               break;
            }
```

注意事项：无。

5.2.13 设置翻转镜像的值

命令：SDKCMD_SET_VIDEO_ENCODE_MIRROR

参数：

```
*param_input: (0:none,1:horizontal,2:vertical,3:horizonta+vertical)
```

使用范例：

```
            case SDKCMD_SET_VIDEO_ENCODE_MIRROR:  //图像镜像翻转
            {
                unsigned int *param_input = (unsigned int*)param;
                retcode = GOS_SDK_Set_ViMirrorMode(0, *param_input, 1);
                if(0 != retcode)
                    return SDK_FAILUR;
               break;
            }
```

注意事项：无。

5.2.14 获取翻转镜像的值

命令：SDKCMD_GET_VIDEO_ENCODE_MIRROR

参数：

*param_output: (0:none,1:horizontal,2:vertical,3:horizonta+vertical)

使用范例：

```
            case SDKCMD GET VIDEO ENCODE MIRROR:
            {
                unsigned int *param_output = (unsigned int*)param;
                retcode = GOS_SDK_Get_ViMirrorMode(0, param_output);
                if(0 != retcode)
                    return SDK_FAILUR;
                break;
            }
```

注意事项：无。

5.2.15 获取当前视频质量

命令：SDKCMD_GET_VIDEO_ENCODE_QUALITY

参数：

```
typedef struct
{
 unsigned int    un_encode_channel_id;
 unsigned int    un_quality;      //value range(0-5)
}T_SDK_VIDEO_ENCODE_QUALITY;
```

使用范例：

```
    case SDKCMD_GET_VIDEO_ENCODE_QUALITY: //获取编码图像质量
            {
                T_SDK_VIDEO_ENCODE_QUALITY *param_output =
    (T_SDK_VIDEO_ENCODE_QUALITY*)param;
                retcode = GOS_SDK_VENC_GetVENCParam_Qulity(param_output);
                if(0 != retcode)
                    return retcode;
                break;

            }
```

注意事项：无。

5.2.16 设置当前码流质量

命令：SDKCMD_SET_CURR_STREAM_QUALITY

参数：*param_input:（高清：0，标清：1）

使用范例：

```
    case SDKCMD_SET_CURR_STREAM_QUALITY: //设置当前码流质量
            {
                unsigned int *param_input = (unsigned int *)param;
                retcode = GOS_SDK_Set_CurrStream_Quality(param_input);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

注意事项：无。

5.2.17 获取当前码流质量

命令：SDKCMD_GET_CURR_STREAM_QUALITY

参数：*param_output:（高清：0，标清：1）

使用范例：

```
case SDKCMD_GET_CURR_STREAM_QUALITY: //获取当前码流质量
        {
             unsigned int *param_output = (unsigned int *)param;
             retcode = GOS_SDK_Get_CurrStream_Quality(param_output);
             if(0 != retcode)
                 return retcode;
             break;
        }
```

注意事项：无。

5.2.18 设置当前图像抓拍质量

命令：SDKCMD_SET_SNAPSHOT_QUALITY

参数： param_input。

使用范例：

```
case SDKCMD_SET_SNAPSHOT_QUALITY: //设置抓拍分辨率(0主码流 1次码流)
        {
             unsigned int *param_input = (unsigned int *)param;
             retcode = GOS_SDK_SetSnapQuality(param_input);
             if(0 != retcode)
                 return retcode;
            break;
        }
```

注意事项：无。

5.2.19 获取当前图像抓拍质量

命令：SDKCMD_GET_SNAPSHOT_QUALITY

参数： param_output。

使用范例：

```
case SDKCMD_GET_SNAPSHOT_QUALITY: //获取抓拍分辨率(0主码流 1次码流)
        {
             unsigned int *param_output = (unsigned int *)param;
             retcode = GOS_SDK_GetSnapQuality(param_output);
             if(0 != retcode)
                 return retcode;
            break;
        }
```

注意事项：无。

5.3 音频命令

此模块用于控制设置音频以及相关的参数，音频模块包括以下命令：

- SDKCMD_GET_AUDIO_ENCODE_PARAM：获取音频配置参数
- SDKCMD_SET_AUDIO_ENCODE_SWITCH：设置音频编码开关

- SDKCMD_SET_SOUND_MODE：设置音频通道类型
- SDKCMD_SET_AUDIO_ENCODE_TYPE：设置音频编码类型
- SDKCMD_SET_AUDIO_ENCODE_BITRATE：设置音频编码比特率
- SDKCMD_SET_AUDIO_ENCODE_SAMPLERATE：设置音频编码采样率
- SDKCMD_SET_AUDIO_INPUT_VOLUME： 设置音频编码输入声音大小
- SDKCMD_SET_AUDIO_OUTPUT_VOLUME：设置音频编码输出声音大小
- SDKCMD_SET_AUDIO_MIC_LINE：设置音频编码输出声音输入模式
- SDKCMD_SET_AUDIO_ECHO_CANCELL：设置音频编码回音消除
- SDKCMD_SET_INTERCOM_PARAM：设置音频对讲参数
- SDKCMD_INTERCOM_START：音频对讲开启
- SDKCMD_INTERCOM_STOP：音频对讲关闭
- SDKCMD_SEND_INTERCOM_DATA:传输音频对讲数据

5.3.1 获取音频配置参数

命令：SDKCMD_GET_AUDIO_ENCODE_PARAM

参数：

```
typedef enum  _E_AUDIO_ENCODE_TYPE
{
 E_AENC_NONE = 0x0,
 E_AENC_G726 = 0x01,
 E_AENC_G722 = 0x02,
 E_AENC_G711A      = 0x03,
 E_AENC_ADPCM      = 0x04,
 E_AENC_MP3  = 0x05,
 E_AENC_PCM  = 0x06,
 E_AENC_G711U      = 0x07,
 E_AENC_AACLC      = 0x08,
 E_AENC_AMRNB      = 0x09,
 E_AENC_PTAAC      = 0x25,
}E_SDK_AUDIO_ENCODE_TYPE;

typedef struct
{
 unsigned int      un_encode_switch;
 unsigned int      un_sound_mode;    //value range of
soud_mode(0:single 1:stereo)
 unsigned int      un_mic_line_input;     //value range(0: mic input
1:line input)
 E_SDK_AUDIO_ENCODE_TYPE      e_encode_type;
 unsigned int      un_audio_echo_canell_switch; //(1:on 0:off)
 unsigned int      un_bitrate;
 unsigned int      un_sample_rate;
 unsigned int      un_input_volume;
 unsigned int      un_output_volume;
} T_SDK_AUDIO_ENCODE_PARAM;
```

使用范例：

```
                case SDKCMD_GET_AUDIO_ENCODE_PARAM://获取实时音视频流参数
            {
                T_SDK_AUDIO_ENCODE_PARAM *param_output =
    (T_SDK_AUDIO_ENCODE_PARAM*)param;
                retcode = GOS_SDK_AENC_GetAENCParam(param_output);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

注意事项：无。

5.3.2 设置音频编码开关

命令：SDKCMD_SET_AUDIO_ENCODE_SWITCH

参数：*param_input：（0:stop 1:start）

使用范例：

```
    case SDKCMD_SET_AUDIO_ENCODE_SWITCH://设置音频开关
            {
                unsigned int *param_input = (unsigned int *)param;
                retcode =
    GOS_SDK_AENC_SetAENCParam_un_switch(param_input);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

注意事项：无。

5.3.3 设置音频通道类型

命令：SDKCMD_SET_SOUND_MODE

参数：param_input。

使用范例：

```
             case SDKCMD_SET_SOUND_MODE://设置音频声道模式
            {
                unsigned int *param_input = (unsigned int *)param;
                retcode =
    GOS_SDK_AENC_SetAENCParam_sound_mode(param_input);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

注意事项：无。

5.3.4 设置音频编码类型

命令：SDKCMD_SET_AUDIO_ENCODE_TYPE

参数：

AUDIO_ENCODER_E EncType

使用范例：

```
      case SDKCMD_SET_AUDIO_ENCODE_TYPE://设置音频编码类型
            {
                 unsigned int *param_input = (unsigned int *)param;
                 retcode = GOS_AI_Setenc_type(param_input);
                 if(0 != retcode)
                     return retcode;
                 break;
            }
```

注意事项：无。

5.3.5 设置音频编码比特率

命令：SDKCMD_SET_AUDIO_ENCODE_BITRATE

参数：unsigned int bit_rate

使用范例：

```
     case SDKCMD_SET_AUDIO_ENCODE_BITRATE://设置音频码率
            {
                 unsigned int *param_input = (unsigned int *)param;
                 retcode = GOS_AI_Setbitrate(param_input);
                 if(0 != retcode)
                     return retcode;
                 break;
            }
```

注意事项：无。

5.3.6 设置音频编码采样率

命令：SDKCMD_SET_AUDIO_ENCODE_SAMPLERATE

参数：unsigned int sample_rate

使用范例：

```
     case SDKCMD_SET_AUDIO_ENCODE_SAMPLERATE://设置音频采样率
            {
                 unsigned int *param_input = (unsigned int *)param;
                 retcode =
     GOS_SDK_AENC_SetAENCParam_sample_rate(param_input);
                 if(0 != retcode)
                     return retcode;
                 break;
            }
```

注意事项：无。

5.3.7 设置音频编码输入声音大小

命令：SDKCMD_SET_AUDIO_INPUT_VOLUME

参数：unsigned int input_vol

使用范例：

```
            case SDKCMD_SET_AUDIO_INPUT_VOLUME ://设置音频音量输入大小
            {
                unsigned int *param_input = (unsigned int *)param;
                retcode = GOS_AI_SetInVol(param_input);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

注意事项：无。

5.3.8 设置音频编码输出声音大小

命令：SDKCMD_SET_AUDIO_OUTPUT_VOLUME

参数：unsigned int output_vol

使用范例：

```
    case SDKCMD_SET_AUDIO_OUTPUT_VOLUME    ://设置音频音量输出声音大小
            {
                unsigned int *param_input = (unsigned int *)param;
                retcode = GOS_AO_SetOutVol(param_input);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

注意事项：无。

5.3.9 设置音频音量输出模式

命令：SDKCMD_SET_AUDIO_MIC_LINE

参数：unsigned int mic_line_input

使用范例：

```
            case SDKCMD_SET_AUDIO_MIC_LINE://设置音频音量输出模式
            {
                unsigned int *param_input = (unsigned int *)param;
                retcode = GOS_AI_Setmic_line_input(param_input);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

注意事项：无。

5.3.10 设置音频编码回音消除

命令：SDKCMD_SET_AUDIO_ECHO_CANCELL

参数：unsigned int AEC

使用范例：

```
            case SDKCMD_SET_AUDIO_ECHO_CANCELL://设置音频回音消除
            {
```

```
                unsigned int *param_input = (unsigned int *)param;
                retcode = GOS_AI_SetAEC(param_input);
                if(0 != retcode)
                    return retcode;
                break;
            }
```

注意事项：无。

5.3.11 设置音频对讲参数

命令：SDKCMD_SET_INTERCOM_PARAM

参数：无。

使用范例：

```
                case SDKCMD_SET_INTERCOM_PARAM:    //设置音频对讲参数
            {
                return SDK_SUCESS;
                break;
            }
```

注意事项：无。

5.3.12 音频对讲开启

命令：SDKCMD_INTERCOM_START
参 数：无。
使用范例：

```
             case SDKCMD_INTERCOM_START:  //开启音频对讲
            {
                GOS_intercome_start();
                return SDK_SUCESS;
                break;
            }
```

注意事项：无。

5.3.13 音频对讲关闭

命令：SDKCMD_INTERCOM_STOP
参 数： 无。
使用范例：

```
            case SDKCMD_INTERCOM_STOP:    //关闭对讲
            {
                GOS_intercome_stop();
                return SDK_SUCESS;
                break;
            }
```

注意事项：无。

5.3.14 传输音频对讲数据

命令：SDKCMD_SEND_INTERCOM_DATA

参数：

```
typedef struct
{
 unsigned char* cp_data;
 unsigned int    un_data_len;
} T_SDK_INTERCOM_DATA;
```

使用范例：

```
      case SDKCMD_SEND_INTERCOM_DATA:  //传输音频数据
      {
                T_SDK_INTERCOM_DATA *param_input = (T_SDK_INTERCOM_DATA
    *)param;
                retcode = GOS_SendTackDate2Adec(param_input-
    >cp_data,param_input->un_data_len);
                if(0 != retcode)
                    return retcode;
                break;
      }
```

注意事项：无。

5.4 Sensor命令

Sensor模块包括以下命令：

- SDKCMD_SET_SENSOR_DEFAULT_PARAM：设置SENSOR默认参数
- SDKCMD_SET_SENSOR_SHARP：设置输出图像 锐度
- SDKCMD_SET_SENSOR_BRIGHTNESS：设置输出图像 亮度
- SDKCMD_SET_SENSOR_CONTRAST：设置输出图像 对比度
- SDKCMD_SET_SENSOR_HUE：设置输出图像 色度
- SDKCMD_SET_SENSOR_SATURATION：设置输出图像 饱和度
- SDKCMD_GET_SENSOR_NIGHT_DAY_STATUE：获取当前光敏电阻状态
- SDKCMD_SET_SENSOR_GAMMA_LEVEL：设置当前的GAMMA
- SDKCMD_SET_SENSOR_SHADING_SWITCH：设定暗角补偿属性。
- SDKCMD_GET_SENSOR_NTSC_PAL: 获取当前视频制式
- SDKCMD_SET_SENSOR_NTSC_PAL： 设置当前视频制式
- SDKCMD_GET_SENSOR_AUTO_EXPOSURE：获取AE
- SDKCMD_SET_SENSOR_AUTO_EXPOSURE：设定AE
- SDKCMD_SET_SENSOR_3D：设定3D降噪

5.4.1 设置输出图像 锐度

命令：SDKCMD_SET_SENSOR_SHARP

参数：

```
typedef struct
{
 unsigned int    una_sharp[4];   //value range(0-100)
```

}T_SDK_SENSOR_SHARP;
使用范例：

```
            case SDKCMD_SET_SENSOR_SHARP://设置输出锐度
            {
                T_SDK_SENSOR_SHARP *param_input =
    (T_SDK_SENSOR_SHARP*)param;
                retcode = GOS_SDK_VPSS_SetChnSpParam(param_input);
                if(0 != retcode)
                    return SDK_FAILUR;
               break;
            }
```

注意事项：无。

5.4.2 设置输出图像 亮度

命令：SDKCMD_SET_SENSOR_BRIGHTNESS

参数：int BRIGHTNESS // (值0~100)

使用范例：

```
             case SDKCMD_SET_SENSOR_BRIGHTNESS://设置输出图像亮度
            {
                unsigned int*param_input = (unsigned int*)param;
                retcode = GOS_SDK_Vi_Set_CSC_Brightness(*param_input);
                if(0 != retcode)
                    return SDK_FAILUR;
               break;
            }
```

注意事项：无。

5.4.3 设置输出图像 对比度

命令：SDKCMD_SET_SENSOR_CONTRAST

参数：int CONTRAST; // (值0~100)

使用范例：

```
             case SDKCMD_SET_SENSOR_CONTRAST:
            {
                unsigned int*param_input = (unsigned int*)param;
                retcode = GOS_SDK_Vi_Set_CSC_Contrast(*param_input);
                if(0 != retcode)
                    return SDK_FAILUR;
               break;
            }
```

注意事项：无。

5.4.4 设置输出图像 色度

命令：SDKCMD_SET_SENSOR_HUE

参数：int HUE; // (值0~100)

使用范例：

```
                case SDKCMD_SET_SENSOR_HUE://设置输出图像色度
            {
                 unsigned int*param_input = (unsigned int*)param;
                 retcode = GOS_SDK_Vi_Set_CSC_Hue(*param_input);
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
```

注意事项：无。

5.4.5 设置输出图像 饱和度

命令：SDKCMD_SET_SENSOR_SATURATION

参数：int SATURATION; //(值0~100)

使用范例：

```
                 case SDKCMD_SET_SENSOR_SATURATION://设置输出图像饱和度
            {
                 unsigned int*param_input = (unsigned int*)param;
                 retcode = GOS_SDK_Vi_Set_CSC_Satu(*param_input);
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
            }
```

注意事项：无。

5.4.6 获取当前光敏电阻状态

命令：SDKCMD_GET_SENSOR_NIGHT_DAY_STATUE

参数：unsigned int ircut //值 0-1 0:day 1:night

使用范例：

```
                case SDKCMD_GET_SENSOR_NIGHT_DAY_STATUE://获取当前光敏电阻状态
            {
                 unsigned int*param_output = (unsigned int*)param;
                 retcode = GOS_SDK_Vi_Get_Ircut(param_output);
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
            }
```

注意事项：(gd5700不支持)

5.4.7 设置当前的GAMMA

命令：SDKCMD_SET_SENSOR_GAMMA_LEVEL

参数：unsigned int GAMMA //值 0-9

使用范例：

```
                case SDKCMD_SET_SENSOR_GAMMA_LEVEL://设置当前的GAMMA
            {
                 unsigned int*param_input = (unsigned int*)param;
                 retcode = GOS_SDK_ISP_SetGammaTable(*param_input);
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
            }
```

注意事项：(gd5700不支持)

5.4.8 设定暗角补偿属性。

命令：SDKCMD_SET_SENSOR_SHADING_SWITCH

参数：unsigned int bool //值 0-1

使用范例：

```
            case SDKCMD_SET_SENSOR_SHADING_SWITCH://设置当前的暗角补偿属性
        {
             unsigned int*param_input = (unsigned int*)param;
             retcode = GOS_SDK_ISP_SetShading(param_input);
             if(0 != retcode)
                 return SDK_FAILUR;
            break;
        }
```

注意事项：(gd5700不支持)

5.4.9 设置当前视频制式

命令：SDKCMD_SET_SENSOR_NTSC_PAL

参数：unsigned int power //值 50 60

使用范例：

```
        case SDKCMD_SET_SENSOR_NTSC_PAL://设置当前视频制式 N:30 P:25
        {
               unsigned int *param_input = (unsigned int*)param;
               retcode = GOS_SDK_VI_Set_Frame(*param_input);
               if(0 != retcode)
                      return SDK_FAILUR;

               break;
        }
```

注意事项：(gd5700不支持)

5.4.10 获取当前视频制式

命令：SDKCMD_SET_SENSOR_NTSC_PAL

参数：*param //值 50 60

使用范例：

```
        case SDKCMD_GET_SENSOR_NTSC_PAL://获取当前视频制式 N:30 P:25
        {
               unsigned int *param_input = (unsigned int*)param;
               retcode = GOS_SDK_VI_Get_Frame(param_input);
               if(0 != retcode)
                      return SDK_FAILUR;

               break;
        }
```

注意事项：(gd5700不支持)

5.4.11 获取AE

命令：SDKCMD_GET_SENSOR_AUTO_EXPOSURE

参数：

```
typedef struct
{
   unsigned int   un_encode_AE_mode;
   unsigned int   un_ExpTimeMax;
   unsigned int   un_ExpTimeMin;
   unsigned int   un_DGainMax;
   unsigned int   un_DGainMin;
   unsigned int   un_AGainMax;
   unsigned int   un_AGainMin;
   unsigned int   un_ISPDGainMax;
  unsigned int    un_SystemGainMax;
  unsigned int    un_GainThreshold;
  unsigned char  uc_ExpStep;
  short    s_ExpTolerance;
  unsigned char  uc_ExpCompensation;
  unsigned short us_EVBias;
  unsigned int    un_AEStrategyMode;
  unsigned char  uc_MaxHistOffset;
  unsigned short us_HistRatioSlope;
  unsigned int    un_FrameEndUpdateMode;
  unsigned int    un_ByPassAE;
}T_SDK_SENSOR_AUTO_EXPOSURE;
```

使用范例：

```
               case SDKCMD_GET_SENSOR_AUTO_EXPOSURE://获取AE
            {
                T_SDK_SENSOR_AUTO_EXPOSURE*param_output =
    (T_SDK_SENSOR_AUTO_EXPOSURE*)param;
                memset(param_output,0,sizeof(T_SDK_SENSOR_AUTO_EXPOSURE));
                retcode = GOS_SDK_ISP_GetAEAttrEx(param_output);
                if(0 != retcode)
                    return SDK_FAILUR;
               break;
            }
```

注意事项：(gd5700不支持)

5.4.12 设定AE

命令：SDKCMD_SET_SENSOR_AUTO_EXPOSURE

参数：

```
typedef struct
{
   unsigned int   un_encode_AE_mode;
```

```
    unsigned int   un_ExpTimeMax;
    unsigned int   un_ExpTimeMin;
    unsigned int   un_DGainMax;
    unsigned int   un_DGainMin;
    unsigned int   un_AGainMax;
    unsigned int   un_AGainMin;
    unsigned int   un_ISPDGainMax;
  unsigned int     un_SystemGainMax;
  unsigned int     un_GainThreshold;
  unsigned char  uc_ExpStep;
  short    s_ExpTolerance;
  unsigned char  uc_ExpCompensation;
  unsigned short us_EVBias;
  unsigned int     un_AEStrategyMode;
  unsigned char  uc_MaxHistOffset;
  unsigned short us_HistRatioSlope;
  unsigned int     un_FrameEndUpdateMode;
  unsigned int     un_ByPassAE;
}T_SDK_SENSOR_AUTO_EXPOSURE;
```

使用范例：

```
          case SDKCMD_SET_SENSOR_AUTO_EXPOSURE://设置AE
             {
                  T_SDK_SENSOR_AUTO_EXPOSURE*param_input =
     (T_SDK_SENSOR_AUTO_EXPOSURE*)param;
                 retcode = GOS_SDK_ISP_SetAEAttrEx(param_input);
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
             }
```

注意事项：(gd5700不支持)

5.4.13 设定3D降噪

命令：SDKCMD_SET_SENSOR_3D

参数：

```
typedef struct
{
   unsigned int   un_space_denoise;
   unsigned int   un_time_denoise;
   unsigned int   un_chroma_range;
}T_SDK_SENSOR_3D;
```

使用范例：

```
                case SDKCMD_SET_SENSOR_3D://3D 降 噪
            {
                 T_SDK_SENSOR_3D*param_input = (T_SDK_SENSOR_3D*)param;
                 retcode = GOS_SDK_Set_VPSS_3DNr(param_input-
    >un_chroma_range,param_input->un_space_denoise,param_input-
    >un_time_denoise);
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
            }
```

注意事项：(gd5700不支持)

5.5 Alarm命令

Alarm模块包括以下命令：

- SDKCMD_RIGISTER_ALARM_CALLBACK：设置告警回调函数
- SDKCMD_GET_ALARM_PARAM：获取报警参数
- SDKCMD_SET_PIR_ALARM_PARAM：设置告警PIR类型开关
- SDKCMD_SET_MOTION_ALARM_PARAM：设置移动侦测
- SDKCMD_SET_AUDIO_ALARM_PARAM：设置音频侦测
- SDKCMD_GET_AUDIO_ALARM_PARAM：获取音频侦测参数
- SDKCMD_SET_DOORBELL_ALARM_PARAM：设置门铃报警参数
- SDKCMD_SET_IO_ALARM_PARAM: 设置IO口报警参数
- SDKCMD_SET_ONEKEY_ALARM_CONTROL_PARAM：设置IPC一键布防的参数
- SDKCMD_GET_ONEKEY_ALARM_CONTROL_PARAM：获取IPC一键布防的参数
- SDKCMD_SET_ALARM_RING_PARAM：设置IPC报警铃声的参数
- SDKCMD_GET_ALARM_RING_PARAM：获取IPC报警铃声的参数
- SDKCMD_PLAY_ALARM_RING_START：开始播放报警铃声
- SDKCMD_PLAY_ALARM_RING_STOP：停止播放报警铃声
- SDKCMD_SET_TEMPERATURE_ALARM_PARAM：设置温度报警参数
- SDKCMD_GET_TEMPERATURE_ALARM_PARAM：获取温度报警参数
- SDKCMD_SET_HUMIDITY_ALARM_PARAM：设置湿度报警参数
- SDKCMD_GET_HUMIDITY_ALARM_PARAM： 获取湿度报警参数
- SDKCMD_SET_WBGT_ALARM_PARAM：设置WBGT 报警参数
- SDKCMD_GET_WBGT_ALARM_PARAM：获取WBGT 报警参数
- SDKCMD_SET_TEMP_HUM_WBGT_ALARM_PARAM：设置温湿度加wbgt报警参数
- SDKCMD_GET_TEMP_HUM_WBGT_ALARM_PARAM：获取温湿度加wbgt报警参数

5.5.1 设置告警回调函数

命令：SDKCMD_RIGISTER_ALARM_CALLBACK

参数：

```
typedef struct
{
 F_SDK_Alarm_Callback        fp_callbak;
 int       n_reserve;
}T_SDK_ALARM_REGISTER_CALLBACK;
```

使用范例：

```
                case SDKCMD_RIGISTER_ALARM_CALLBACK:
            {
                 //回调实现
                 T SDK ALARM REGISTER CALLBACK *param input =
     (T_SDK_ALARM_REGISTER_CALLBACK* )param;
                 retcode = GOS_SDK_ARLAM_Init(param_input);
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
            }
```

注意事项：无。

5.5.2 获取报警参数

命令：SDKCMD_GET_ALARM_PARAM

参数：

```
typedef enum {
  AM_MD_ALGO_MSE    = (0x01 << 0),
  AM_MD_ALGO_MOG2   = (0x01 << 1),
  AM_MD_ALGO_HYBRID= AM_MD_ALGO_MSE | AM_MD_ALGO_MOG2,
  AM_MD_ALGO_NUM    = 2,
} E_SDK_VIDEO_MOTION_ALGO;

typedef struct {
   unsigned int    un_start_x;
   unsigned int    un_start_y;
   unsigned int    un_width;
   unsigned int    un_height;
   unsigned int    c_switch;
   unsigned int    c_sensitivity;           //value range(1 ~100)，移动侦测
等级阀值,值越小,灵敏度越高。1(max) ~ 100(min):  (关: 100,  低: 90  中: 60  高:
30)
   short     s_threshold; //when below threshold, not considered as
motion; else, trigger motion
}T_SDK_VIDEO_MOTION_SINGEL;

typedef struct
{
 E_SDK_VIDEO_MOTION_ALGO e_algo_type;//AM_MD_ALGO_MSE or AM_MD_ALGO_MOG2
 T_SDK_VIDEO_MOTION_SINGEL t_video_motion_rect; //手动划分时只划分一个区域坐标
 unsigned int un_mode;    // 手动划分坐标0 or 自动多分屏坐标1
 unsigned int un_submode; //多分屏下1x1=0, 2x2=1, 3x3=2, 4x4=3
 unsigned int un_enable;//根据多分屏模式下选择区域是否使能最多4x4=16;
}T_SDK_VIDEO_MOTION_ALARM;

typedef struct
{
```

```
 unsigned intun_switch;
 unsigned intun_sensitivity;  //PIR侦测等级(1-10)，值越小，灵敏度越高  (低：8，
中：5，高：2)
}T_SDK_PIR_ALARM;          //(only for HiSilicon)

typedef struct
{
 unsigned intun_switch;
 unsigned intun_sensitivity;
}T_SDK_DOORBELL_ALARM;     //(only for HiSilicon)

typedef struct
{
 unsigned intun_switch;
 unsigned intun_sensitivity;
}T_SDK_AUDIO_ALARM;

typedef struct
{
 unsigned intun_switch;
 unsigned intun_sensitivity;
}T_SDK_SHELTER_ALARM;     //(not support yet)

typedef struct
{
 unsigned intun_switch;
 unsigned intun_sensitivity;
}T_SDK_IO_ALARM;

typedef struct
{
 T_SDK_VIDEO_MOTION_ALARMt_motion;
 T_SDK_PIR_ALARM   t_PRI;//(only for HiSilicon)
    T_SDK_DOORBELL_ALARM t_doorbell; //(only for HiSilicon)
 T_SDK_AUDIO_ALARM t_audio;
 T_SDK_SHELTER_ALARM     t_shelter;  //(not support yet)
    T_SDK_IO_ALARM t_IO;
}T_SDK_ALARM_PARAM;
```

使用范例：

```
                case SDKCMD_GET_ALARM_PARAM:
            {
                 T_SDK_ALARM_PARAM *param_output =
    (T_SDK_ALARM_PARAM* )param;
                 memset(param_output,0,sizeof(T_SDK_ALARM_PARAM));
                 retcode = GOS_SDK_Get_Alarm_Param(param_output);
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
            }
```

注意事项：无。

5.5.3 设置告警PIR类型开关

命令：SDKCMD_SET_PIR_ALARM_PARAM

参数：

```
typedef struct
{
  unsigned int     un_switch;
  unsigned int     un_sensitivity;  //PIR侦测等级(1-10)，值越小，灵敏度越高  (低: 8, 中: 5, 高: 2)
}T_SDK_PIR_ALARM;         //(only for HiSilicon)
```

使用范例：

```
             case SDKCMD_SET_PIR_ALARM_PARAM://设置PIR告警回调参数
             {
                  T_SDK_PIR_ALARM *param_input = (T_SDK_PIR_ALARM* )param;
                  retcode =  GOS_SDK_Set_Alarm_Pir_Param(param_input);
                  if(0 != retcode)
                      return SDK_FAILUR;
                 break;
             }
```

注意事项：(gd5700不支持)

5.5.4 设置移动侦测

命令：SDKCMD_SET_MOTION_ALARM_PARAM

参数：

```
typedef struct {
   unsigned int    un_start_x;
   unsigned int    un_start_y;
   unsigned int    un_width;
   unsigned int    un_height;
   unsigned int    c_switch;
   unsigned int    c_sensitivity;              //value range(1 ~100)，移动侦测等级阀值,值越小,灵敏度越高。1(max) ~ 100(min):  (关: 100,  低: 90  中: 60  高: 30)
   short   s_threshold;      //when below threshold, not considered as motion; else, trigger motion
}T_SDK_VIDEO_MOTION_SINGEL;


typedef enum {
  AM_MD_ALGO_MSE  = (0x01 << 0),
  AM_MD_ALGO_MOG2          = (0x01 << 1),
  AM_MD_ALGO_HYBRID        = AM_MD_ALGO_MSE | AM_MD_ALGO_MOG2,
  AM_MD_ALGO_NUM  = 2,
} E_SDK_VIDEO_MOTION_ALGO;


typedef struct
{
  E_SDK_VIDEO_MOTION_ALGO e_algo_type;          //AM_MD_ALGO_MSE or AM_MD_ALGO_MOG2
  T_SDK_VIDEO_MOTION_SINGEL t_video_motion_rect; //手动划分时只划分一个区域坐标
  unsigned int un_mode; // 手动划分坐标0 or 自动多分屏坐标1
```

unsigned int un_submode; //多分屏下1x1=0, 2x2=1, 3x3=2, 4x4=3

unsigned int un_enable;//根据多分屏模式下选择区域是否使能最多4x4=16;

}T_SDK_VIDEO_MOTION_ALARM;

使用范例：

```
                case SDKCMD_SET_MOTION_ALARM_PARAM:
            {
                 T_SDK_VIDEO_MOTION_ALARM *param_input =
    (T_SDK_VIDEO_MOTION_ALARM* )param;
                 retcode = GOS_SDK_Set_Alarm_Motion_Param(param_input);
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
            }
```

注意事项：无。

5.5.5 设置音频侦测

命令：SDKCMD_SET_AUDIO_ALARM_PARAM

参数：

typedef struct

{

unsigned int un_switch;

unsigned int un_sensitivity;

}T_SDK_AUDIO_ALARM;

使用范例：

```
            case SDKCMD_SET_AUDIO_ALARM_PARAM://设置音频告警回调参数
            {
                 T_SDK_AUDIO_ALARM *param_input =
    (T_SDK_AUDIO_ALARM*)param;
                 retcode = GOS_SDK_Set_Alarm_Audio_Param(param_input);
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
            }
```

注意事项：无。

5.5.6 获取音频侦测

命令：SDKCMD_GET_AUDIO_ALARM_PARAM

参数：

typedef struct

{

unsigned int un_switch;

unsigned int un_sensitivity;

}T_SDK_AUDIO_ALARM;

使用范例：

```
        case SDKCMD_GET_AUDIO_ALARM_PARAM://获取音频告警回调参数
         {
            T_SDK_AUDIO_ALARM *param_input = (T_SDK_AUDIO_ALARM*)param;
```

```
            retcode = GOS_SDK_Get_Alarm_Audio_Param(param_input);
            if(0 != retcode)
               return SDK_FAILUR;
                   break;
      }
```

注意事项：无。

5.5.7 设置门铃报警参数

命令：SDKCMD_SET_DOORBELL_ALARM_PARAM

参数：

```
typedef struct
{
 unsigned int      un_switch;
 unsigned int      un_sensitivity;
}T_SDK_DOORBELL_ALARM;     //(only for HiSilicon)
```

使用范例：

```
                  case SDKCMD_SET_DOORBELL_ALARM_PARAM://设置按门铃回调参数
            {
                 T_SDK_DOORBELL_ALARM *param_input =
    (T_SDK_DOORBELL_ALARM*)param;
                 retcode = GOS_SDK_Set_Alarm_Calling_Param(param_input);
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
            }
```

注意事项：无。

5.5.8 设置IO口报警参数

命令：SDKCMD_SET_IO_ALARM_PARAM

参数：

```
typedef struct
{
      unsigned int      un_switch;
      unsigned int      un_sensitivity;
}T_SDK_IO_ALARM;
```

使用范例：

```
                  case SDKCMD_SET_IO_ALARM_PARAM://设置IO回调参数 探头报警
            {
                 T_SDK_IO_ALARM *param_input = (T_SDK_IO_ALARM*)param;
                 retcode = GOS_SDK_Set_Alarm_Io_Param(param_input);
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
            }
```

注意事项：无。

5.5.9 设置IPC一键布防的参数

命令： SDKCMD_SET_ONEKEY_ALARM_CONTROL_PARAM

参数：

```
typedef struct
{
 unsigned int      un_switch;
 unsigned int      reserved[3];
}T_SDK_ONEKEY_ALARM_CONTROL;
```

使用范例：

```
      case SDKCMD_SET_ONEKEY_ALARM_CONTROL_PARAM:
      {
            T_SDK_ONEKEY_ALARM_CONTROL *param_input =
      (T_SDK_ONEKEY_ALARM_CONTROL* )param;
            retcode =
      GOS_SDK_Set_OneKey_Alarm_Control_Param(param_input
      );
            if(0 != retcode)
                  return SDK_FAILUR;
            break;
      }
```

注意事项：无。

5.5.10 获取IPC一键布防的参数

命令： SDKCMD_GET_ONEKEY_ALARM_CONTROL_PARAM

参数：

```
typedef struct
{
 unsigned int      un_switch;
 unsigned int      reserved[3];
}T_SDK_ONEKEY_ALARM_CONTROL;
```

使用范例：

```
             case SDKCMD_GET_ONEKEY_ALARM_CONTROL_PARAM:
             {
                   T_SDK_ONEKEY_ALARM_CONTROL *param_output =
(T SDK ONEKEY ALARM CONTROL* )param;
            memset(param output, 0, sizeof(T SDK ONEKEY ALARM CONTROL));
            retcode = GOS_SDK_Get_OneKey_Alarm_Control_Param(param_output);
            if(0 != retcode)
                return SDK_FAILUR;

                   break;
             }
```

注意事项：无。

5.5.11 设置IPC报警铃声的参数

命令： SDKCMD_SET_ALARM_RING_PARAM

参数：

```
typedef struct
{
 unsigned int      alarm_ring_no;
 unsigned int      reserved[3];
}T_SDK_ALARM_RING_PARAM;
```

使用范例：

```
             case SDKCMD_SET_ALARM_RING_PARAM:
             {
                    T_SDK_ALARM_RING_PARAM *param_input =
(T_SDK_ALARM_RING_PARAM* )param;
            retcode = GOS_SDK_Set_Alarm_Ring_Param(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
```

注意事项：无。

5.5.12 获取IPC报警铃声的参数

命令： SDKCMD_GET_ALARM_RING_PARAM

参数：

```
typedef struct
{
 unsigned int      alarm_ring_no;
 unsigned int      reserved[3];
}T_SDK_ALARM_RING_PARAM;
```

使用范例：

```
                    case SDKCMD_GET_ALARM_RING_PARAM:
            {
                T_SDK_ALARM_RING_PARAM *param_output =
     (T_SDK_ALARM_RING_PARAM* )param;
                retcode = GOS_SDK_Get_Alarm_Ring_Param(param_output);
                if(0 != retcode)
                    return SDK_FAILUR;
                break;
            }
```

注意事项：无。

5.5.13 开始播放报警铃声

命令：SDKCMD_PLAY_ALARM_RING_START

参数：无。

使用范例：

```
                    case SDKCMD_PLAY_ALARM_RING_START:
            {
                 retcode = GOS_SDK_Play_Alarm_Ring();
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
            }
```

注意事项：无。

5.5.14 停止播放报警铃声

命令：SDKCMD_PLAY_ALARM_RING_STOP

参数：无。

使用范例：

```
                case SDKCMD_PLAY_ALARM_RING_STOP:
            {
                 retcode = GOS_SDK_Stop_Play_Alarm_Ring();
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
            }
```

注意事项：无。

5.5.15 设置温度报警参数

命令：SDKCMD_SET_TEMPERATURE_ALARM_PARAM

参数：

```
typedef struct
{
 unsigned int alarm_enale;                  //上下限温度报警开关， 0:上下限全部关闭， 1:上限开启，下限关闭，2:上限关闭，下限开启，3:上下限全部开启
 unsigned int temperature_type;             //温度表示类型， 0:表示摄氏温度.C， 1；表示华氏温度.F
 double curr_temperature_value;             //当前温度
 double max_alarm_value;                          //上限报警温度
 double min_alarm_value;                          //下限报警温度
 unsigned char reserved[16];                //
}T_SDK_TEMPERATURE_ALARM_PARAM;
```

使用范例：

```
            case SDKCMD_SET_TEMPERATURE_ALARM_PARAM:
            {
                T_SDK_TEMPERATURE_ALARM_PARAM *param_input =
    (T_SDK_TEMPERATURE_ALARM_PARAM* )param;
```

```
                retcode =
    GOS_SDK_Set_Temperature_Alarm_Param(param_input);
                if(0 != retcode)
                    return SDK_FAILUR;
                break;
            }
```

注意事项：无。

5.5.16 获取温度报警参数

命令：SDKCMD_GET_TEMPERATURE_ALARM_PARAM

参数：

```
typedef struct
{
 unsigned int alarm_enale;                  //上下限温度报警开关， 0:上下限全部关闭， 1:上限开启，下限关闭，2:上限关闭，下限开启，3:上下限全部开启
 unsigned int temperature_type;             //温度表示类型， 0:表示摄氏温度.C， 1；表示华氏温度.F
 double curr_temperature_value;             //当前温度
 double max_alarm_value;                        //上限报警温度
 double min_alarm_value;                        //下限报警温度
 unsigned char reserved[16];                //
}T_SDK_TEMPERATURE_ALARM_PARAM;
```

使用范例：

```
            case SDKCMD_GET_TEMPERATURE_ALARM_PARAM:
            {
                T_SDK_TEMPERATURE_ALARM_PARAM *param_output =
    (T_SDK_TEMPERATURE_ALARM_PARAM* )param;
                memset(param_output, 0,
    sizeof(T_SDK_TEMPERATURE_ALARM_PARAM));
                retcode =
    GOS_SDK_Get_Temperature_Alarm_Param(param_output);
                if(0 != retcode)
                    return SDK_FAILUR;
                break;
            }
```

注意事项：无。

5.5.17 设置湿度报警参数

命令：SDKCMD_SET_HUMIDITY_ALARM_PARAM

参数：

```
typedef struct
{
 unsigned int alarm_enale;                  //上下限湿度报警开关， 0:上下限全
```

部关闭， 1:上限开启，下限关闭，2:上限关闭，下限开启，3:上下限全部开启

```
 unsigned int humidity_type;        //湿度表示类型保留
 double curr_humidity_value;        //当前湿度
 double max_alarm_value;                        //上限报警湿度
 double min_alarm_value;                        //下限报警湿度
 unsigned char reserved[16];              //
}T_SDK_HUMIDITY_ALARM_PARAM;
```

使用范例：

```
            case SDKCMD_SET_HUMIDITY_ALARM_PARAM:
            {
                  T_SDK_HUMIDITY_ALARM_PARAM *param_input =
(T_SDK_HUMIDITY_ALARM_PARAM* )param;
                  retcode = GOS_SDK_Set_Humidity_Alarm_Param(param_input);
                  if(0 != retcode)
                        return SDK_FAILUR;
                  break;
            }
```

注意事项：无。

5.5.18 获取湿度报警参数

命令：SDKCMD_GET_HUMIDITY_ALARM_PARAM

参数：

```
typedef struct
{
 unsigned int alarm_enale;                //上下限湿度报警开关， 0:上下限全
```

部关闭， 1:上限开启，下限关闭，2:上限关闭，下限开启，3:上下限全部开启

```
 unsigned int humidity_type;        //湿度表示类型保留
 double curr_humidity_value;        //当前湿度
 double max_alarm_value;                        //上限报警湿度
 double min_alarm_value;                        //下限报警湿度
 unsigned char reserved[16];              //
}T_SDK_HUMIDITY_ALARM_PARAM;
```

使用范例：

```
            case SDKCMD_GET_HUMIDITY_ALARM_PARAM:
            {
                  T_SDK_HUMIDITY_ALARM_PARAM *param_output =
(T_SDK_HUMIDITY_ALARM_PARAM* )param;
                  memset(param_output, 0,
sizeof(T_SDK_HUMIDITY_ALARM_PARAM));
                  retcode = GOS_SDK_Get_Humidity_Alarm_Param(param_output);
                  if(0 != retcode)
                        return SDK_FAILUR;
                  break;
```

注意事项：无。

5.5.19 设置WBGT 报警参数

命令：SDKCMD_SET_WBGT_ALARM_PARAM

参数：

```
typedef struct
{
 unsigned int alarm_enale;                //上下限WBGT报警开关， 0:上下限全部关闭， 1:上限开启，下限关闭，2:上限关闭，下限开启，3:上下限全部开启
 double curr_wbgt_value;            //当前WBGT
 double max_alarm_value;                        //上限报警WBGT
 double min_alarm_value;                        //下限报警WBGT
 unsigned char reserved[16];              //
}T_SDK_WBGT_ALARM_PARAM;
```

使用范例：

```
            case SDKCMD_SET_WBGT_ALARM_PARAM:
            {
                  T_SDK_WBGT_ALARM_PARAM *param_input =
(T_SDK_WBGT_ALARM_PARAM* )param;
                  retcode = GOS_SDK_Set_WBGT_Alarm_Param(param_input);
                  if(0 != retcode)
                        return SDK_FAILUR;
                  break;
            }
```

注意事项：无。

5.5.20 获取WBGT 报警参数

命令：SDKCMD_GET_WBGT_ALARM_PARAM

参数：

```
typedef struct
{
 unsigned int alarm_enale;                //上下限WBGT报警开关， 0:上下限全部关闭， 1:上限开启，下限关闭，2:上限关闭，下限开启，3:上下限全部开启
 double curr_wbgt_value;            //当前WBGT
 double max_alarm_value;                        //上限报警WBGT
 double min_alarm_value;                        //下限报警WBGT
 unsigned char reserved[16];              //
}T_SDK_WBGT_ALARM_PARAM;
```

使用范例：

```
            case SDKCMD_GET_WBGT_ALARM_PARAM:
            {
                  T_SDK_WBGT_ALARM_PARAM *param_output =
(T_SDK_WBGT_ALARM_PARAM* )param;
                  memset(param_output, 0, sizeof(T_SDK_WBGT_ALARM_PARAM));
                  retcode = GOS_SDK_Get_WBGT_Alarm_Param(param_output);
                  if(0 != retcode)
                        return SDK_FAILUR;
                  break;
            }
```

注意事项：无。

5.5.21 设置温湿度加wbgt报警参数

命令：SDKCMD_SET_TEMP_HUM_WBGT_ALARM_PARAM

参数：

```
//温度报警参数结构体
typedef struct
{
 unsigned int alarm_enale;                  //上下限温度报警开关， 0:上下限全部关闭， 1:上限开启，下限关闭，2:上限关闭，下限开启，3:上下限全部开启
 unsigned int temperature_type;             //温度表示类型， 0:表示摄氏温度.C， 1；表示华氏温度.F
 double curr_temperature_value;             //当前温度
 double max_alarm_value;                          //上限报警温度
 double min_alarm_value;                          //下限报警温度
 unsigned char reserved[16];                //
}T_SDK_TEMPERATURE_ALARM_PARAM;

//湿度报警参数结构体
typedef struct
{
 unsigned int alarm_enale;                  //上下限湿度报警开关， 0:上下限全部关闭， 1:上限开启，下限关闭，2:上限关闭，下限开启，3:上下限全部开启
 unsigned int humidity_type;          //湿度表示类型保留
 double curr_humidity_value;          //当前湿度
 double max_alarm_value;                          //上限报警湿度
 double min_alarm_value;                          //下限报警湿度
 unsigned char reserved[16];                //
```

```
}T_SDK_HUMIDITY_ALARM_PARAM;

//WBGT值报警参数结构体
typedef struct
{
 unsigned int alarm_enale;                  //上下限WBGT报警开关， 0:上下限全部关闭， 1:上限开启，下限关闭，2:上限关闭，下限开启，3:上下限全部开启
 double curr_wbgt_value;              //当前WBGT
 double max_alarm_value;                              //上限报警WBGT
 double min_alarm_value;                              //下限报警WBGT
 unsigned char reserved[16];               //
}T_SDK_WBGT_ALARM_PARAM;

typedef struct
{
 T_SDK_TEMPERATURE_ALARM_PARAM t_temperature;
 T_SDK_HUMIDITY_ALARM_PARAM t_humidity;
 T_SDK_WBGT_ALARM_PARAM t_wbgt;
}T_SDK_TEMP_HUM_WBGT_ALARM_PARAM;
```

使用范例：

```
              case SDKCMD_SET_TEMP_HUM_WBGT_ALARM_PARAM:
              {
                     T_SDK_TEMP_HUM_WBGT_ALARM_PARAM *param_input =
(T_SDK_TEMP_HUM_WBGT_ALARM_PARAM* )param;
                     retcode =
GOS_SDK_Set_TEMP_HUM_WBGT_Alarm_Param(param_input);
                     if(0 != retcode)
                            return SDK_FAILUR;
                     break;
              }
```

注意事项：无。

5.5.22 获取温湿度加wbgt报警参数

命令：SDKCMD_GET_TEMP_HUM_WBGT_ALARM_PARAM

参数：

```
//温度报警参数结构体
typedef struct
{
 unsigned int alarm_enale;                  //上下限温度报警开关， 0:上下限全
```

```
部关闭， 1:上限开启，下限关闭，2:上限关闭，下限开启，3:上下限全部开启
 unsigned int temperature_type;          //温度表示类型， 0:表示摄氏温
度.C， 1；表示华氏温度.F
 double curr_temperature_value;          //当前温度
 double max_alarm_value;                       //上限报警温度
 double min_alarm_value;                       //下限报警温度
 unsigned char reserved[16];             //
}T_SDK_TEMPERATURE_ALARM_PARAM;

//湿度报警参数结构体
typedef struct
{
 unsigned int alarm_enale;               //上下限湿度报警开关， 0:上下限全
部关闭， 1:上限开启，下限关闭，2:上限关闭，下限开启，3:上下限全部开启
 unsigned int humidity_type;        //湿度表示类型保留
 double curr_humidity_value;        //当前湿度
 double max_alarm_value;                       //上限报警湿度
 double min_alarm_value;                       //下限报警湿度
 unsigned char reserved[16];             //
}T_SDK_HUMIDITY_ALARM_PARAM;

//WBGT值报警参数结构体
typedef struct
{
 unsigned int alarm_enale;               //上下限WBGT报警开关， 0:上下限
全部关闭， 1:上限开启，下限关闭，2:上限关闭，下限开启，3:上下限全部开启
 double curr_wbgt_value;            //当前WBGT
 double max_alarm_value;                       //上限报警WBGT
 double min_alarm_value;                       //下限报警WBGT
 unsigned char reserved[16];             //
}T_SDK_WBGT_ALARM_PARAM;

typedef struct
{
```

```
 T_SDK_TEMPERATURE_ALARM_PARAM t_temperature;
 T_SDK_HUMIDITY_ALARM_PARAM t_humidity;
 T_SDK_WBGT_ALARM_PARAM t_wbgt;
}T_SDK_TEMP_HUM_WBGT_ALARM_PARAM;
```

使用范例：

```
             case SDKCMD_GET_TEMP_HUM_WBGT_ALARM_PARAM:
             {
                    T_SDK_TEMP_HUM_WBGT_ALARM_PARAM *param_output =
(T_SDK_TEMP_HUM_WBGT_ALARM_PARAM* )param;
            memset(param_output, 0, sizeof(T_SDK_TEMP_HUM_WBGT_ALARM_PARAM));
            retcode = GOS_SDK_Get_TEMP_HUM_WBGT_Alarm_Param(param_output);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
```

注意事项：无。

5.6 Osd命令

Osd模块包括以下命令：

- SDKCMD_SET_OSD_DEFAULT_PARAM：恢复OSD默认参数
- SDKCMD_GET_OSD_PARAM：获取OSD基本参数
- SDKCMD_SET_OSD_SHOW_SWITCH：设置OSD是否显示
- SDKCMD_SET_OSD_COLOR：设置OSD颜色
- SDKCMD_SET_OSD_POS：移动OSD位置
- SDKCMD_SET_OSD_TITLE：设置OSD标题

5.6.1 恢复OSD默认参数

命令：SDKCMD_SET_OSD_DEFAULT_PARAM
参 数：无。
使用范例：

```
               case SDKCMD_SET_OSD_DEFAULT_PARAM://设置OSD默认参数
            {
                 retcode = GOS_SDK_Set_Osd_DefualtParm();
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
            }
```

注意事项：无。

5.6.2 获取OSD基本参数

命令：SDKCMD_GET_OSD_PARAM

参数：

```
typedef struct
{
 unsigned int    un_show;        //value range(1:display 0:close)
 unsigned int    un_color;       //value range(1:Black 2:Yelloow 3:Red 4:Blue 5:White Other
```

```
default red)
  unsigned int      un_x_coordinate;
  unsigned int      un_y_coordinate;
}T_SDK_SINGLE_OSD_PARAM;

typedef struct
{
  unsigned int      un_encode_channel_id; //(Input Param)
  char      a_title1[64];
  char      a_title2[64];
  T_SDK_SINGLE_OSD_PARAM        t_time;
  T_SDK_SINGLE_OSD_PARAM        t_title1;
  T_SDK_SINGLE_OSD_PARAM        t_title2;
}T_SDK_ALL_OSD_PARAM;
```

使用范例：

```
            case SDKCMD_GET_OSD_PARAM://获取OSD参数
            {
                T_SDK_ALL_OSD_PARAM *param_output =
    (T_SDK_ALL_OSD_PARAM*)param;
                retcode = GOS_SDK_Get_Osd_Parm(param_output);
                if(0 != retcode)
                    return SDK_FAILUR;
               break;
            }
```

注意事项：无。

5.6.3 设置OSD是否显示

命令：SDKCMD_SET_OSD_SHOW_SWITCH

参数：

```
typedef struct
{
  unsigned int      un_encode_channel_id;
  unsigned int      un_show_Time;         //value range(1:display 0:close)
  unsigned int      un_show_Date;         //value range(1:display 0:close)
  unsigned int      un_show_Title1;       //value range(1:display 0:close)
  unsigned int      un_show_Title2;       //value range(1:display 0:close)
}T_SDK_OSD_SWITCH;
```

使用范例：

```
           case SDKCMD_SET_OSD_SHOW_SWITCH://打开或关闭OSD叠加
           {
                T_SDK_OSD_SWITCH *param_input = (T_SDK_OSD_SWITCH*)param;
                retcode = GOS_SDK_OsdShow_Ctrl(param_input);
                if(0 != retcode)
                    return SDK_FAILUR;
               break;
           }
```

注意事项：无。

5.6.4 设置OSD颜色

命令：SDKCMD_SET_OSD_COLOR

参数：

```
typedef struct
{
  unsigned int     un_encode_channel_id;
  unsigned int     un_color;        //value range(1:Black 2:Yelloow 3:Red 4:Blue 5:White Other default red)
}T_SDK_OSD_COLOR;
```

使用范例：

```
case SDKCMD_SET_OSD_COLOR://设置OSD颜色
{
     T_SDK_OSD_COLOR *param_input = (T_SDK_OSD_COLOR*)param;
     retcode = GOS_SDK_OsdColor_Ctrl(param_input);
     if(0 != retcode)
         return SDK_FAILUR;
    break;
}
```

注意事项：无。

5.6.5 移动OSD位置

命令：SDKCMD_SET_OSD_POS

参数：

```
typedef struct
{
  unsigned int     un_encode_channel_id;
  unsigned int     un_x_time;
  unsigned int     un_y_time;
  unsigned int     un_x_title1;
  unsigned int     un_y_title1;
    unsigned int   un_x_title2;
  unsigned int     un_y_title2;
}T_SDK_OSD_POS;
```

使用范例：

```
case SDKCMD_SET_OSD_POS://移动OSD位置
{
     T_SDK_OSD_POS *param_input = (T_SDK_OSD_POS*)param;
     retcode = GOS_SDK_OsdPos_Ctrl(param_input);
     if(0 != retcode)
         return SDK_FAILUR;
    break;
}
```

注意事项：无。

5.6.6 设置OSD标题

命令：SDKCMD_SET_OSD_TITLE

参数：

```
typedef struct
```

```
{
  unsigned int     un_encode_channel_id;
  char     a_title1[64];
  char     a_title2[64];
}T_SDK_OSD_TITLE;
```

使用范例：

```
case SDKCMD_SET_OSD_TITLE://设置标题
{
     T_SDK_OSD_TITLE *param_input = (T_SDK_OSD_TITLE*)param;
     retcode =  GOS_SDK_OSD_SetChNameTitle(param_input);
     if(0 != retcode)
         return SDK_FAILUR;
    break;
}
```

注意事项：无。

5.7 录像命令

录像模块包括以下命令：

- SDKCMD_SET_RECORD_DEFAULT_PARAM：设置录像默认参数
- SDKCMD_GET_RECORD_PARAM：获取录像参数
- SDKCMD_SET_RECORD_SWITCH：启动关闭录像
- SDKCMD_CLEAR_RECORD_FILE：清除所有录像
- SDKCMD_LOCK_UNLOCK_RECORD_FILE：加解锁录像文件
- SDKCMD_GET_MONTH_RECORD_LIST：按月份查找录像文件
- SDKCMD_GET_DAY_RECORD_LIST：按天获取录像文件列表
- SDKCMD_GET_DAY_ASSIGNTIME_RECORD_LIST：获取制定时间录像文件列表
- SDKCMD_GET_RECORD_FILE_FULL_PATH：获取录像文件的绝对路径
- SDKCMD_SET_LOOP_RECORD_SWITCH：开启关闭循环录像功能
- SDKCMD_SET_AUDIO_RECORD_SWITCH：开启关闭是否录制音频
- SDKCMD_SET_RECORD_FILE_DURATION：设置单个录像文件时长
- SDKCMD_GET_RECORD_FILE_DURATION：获取单个录像文件时长
- SDKCMD_SET_RECORD_FILE_TYPE：设置录像音视频格式
- SDKCMD_DEL_RECORD_FILE：删除记录文件
- SDKCMD_MANUAL_RECORD_SWITCH：手动记录开关
- SDKCMD_GET_STORAGE_INFO：获取储存信息
- SDKCMD_FORMAT_STORAGE_REQ：格式化储存

5.7.1 设置录像默认参数

命令：SDKCMD_SET_RECORD_DEFAULT_PARAMS
参 数 ： 无。
使用范例：

```
            case SDKCMD_SET_RECORD_DEFAULT_PARAM:
            {
                retcode = sdk_SetRecordDefaultParam();
                if(SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "sdk_SetRecordDefaultParam
    FAILED!\n");
                    return SDK_FAILUR;
                }
                break;
            }
```

注意事项：无。

5.7.2 获取录像参数

命令：SDKCMD_GET_RECORD_PARAMS

参数：

```
typedef struct
{
  unsigned int    un_switch;
  unsigned int    un_audio_switch;
  unsigned int    un_manual_record_switch;     //手动录像开关
  unsigned int    un_encode_channel_id;
  unsigned int    un_file_duration;
  unsigned int    un_loop;         //value range of auto circulate record(0:on 1:off)
  unsigned int    un_file_type;    //record file type value range (0:ts 1:mp4 2:flv)
}T_SDK_RECORD_PARAM;
```

使用范例：

```
            case SDKCMD_GET_RECORD_PARAM:
            {
                T_SDK_RECORD_PARAM *pData = (T_SDK_RECORD_PARAM*)param;
                retcode = sdk_GetRecordParam(pData);
                if(SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "sdk_getRecordParam FAILED!\n");
                    return SDK_FAILUR;
                }
                break;
            }
```

注意事项：无。

5.7.3 启动关闭录像

命令：SDKCMD_SET_RECORD_SWITCH

参数：unsigned int un_switch;

使用范例：

```
            case SDKCMD_SET_RECORD_SWITCH:
            {
                retcode = sdk_SetRecordSwitch((unsigned int *)param);
                if(SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "sdk_SetRecordSwitch
    FAILED!\n");
                    return SDK_FAILUR;
                }
                break;
            }
```

注意事项：无。

5.7.4 清除所有录像

命令：SDKCMD_CLEAR_RECORD_FILE

参 数 ： 无。

使用范例：

```
            case SDKCMD_CLEAR_RECORD_FILE:
            {
                retcode = sdk_ClearRecordFile();
                if(SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "sdk_ClearRecordFile
    FAILED!\n");
                    return SDK_FAILUR;
                }
                break;
            }
```

注意事项：无。

5.7.5 加解锁录像文件

命令：SDKCMD_LOCK_UNLOCK_RECORD_FILE

参数：

```
typedef struct
{
 unsigned int    un_lock;
 char    a_file_name[64];
}T_SDK_RECORD_LOCK_FILE;
```

使用范例：

```
                case SDKCMD_LOCK_UNLOCK_RECORD_FILE:
            {
                T_SDK_RECORD_LOCK_FILE *pData =
    (T_SDK_RECORD_LOCK_FILE*)param;
                retcode = sdk_LockUnlockRecordFile(pData);
                if(SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "sdk_LockUnlockRecordFile
    FAILED!\n");
                    return SDK_FAILUR;
                }
                break;
            }
```

注意事项：无。

5.7.6 按月份查找录像文件

命令：SDKCMD_GET_MONTH_RECORD_LIST

参数：

typedef struct

{

char a_month[16]; //example:"201401" (input Param. 2014 means year,01 means month)

char* cp_list; //example:"201401013|201401021|201401122|" (Output Param. 2014 means year , in 201401013 01 means month ,3 meas how many files exist)

unsigned int un_list_len;

}T_SDK_RECORD_MONTH_LIST;

使用范例：

```
            case SDKCMD_GET_MONTH_RECORD_LIST:
            {
                T_SDK_RECORD_MONTH_LIST *pData =
    (T_SDK_RECORD_MONTH_LIST*)param;
                retcode = sdk_GetMonthRecordList(pData);
                if(SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "sdk_GetMonthRecordList
    FAILED!\n");
                    return retcode;
                }
                break;
            }
```

注意事项：无。

5.7.7 按天获取录像文件列表

命令：SDKCMD_GET_DAY_RECORD_LIST

参数：

typedef struct

{

char a_day[16]; //example:"20140112"(input Param. 2014 means year,01 means month,12 mens the day)

unsigned int file_type; //请求文件类型 (视频: 0 , 图片: 1)

unsigned int file_counts; //文件数

/*example:"201401121132231bc0120.mp4|201401121133231bc0120.mp4|"

(Output Param. in 201401121132231bc0120z.ts 2014 means year,01 means month ,

12 meas the day,11 means hour, 32 means minute, 23 means second,

1 means the file can not remove when auto circulate record is open,

b means record type (value

frome enum RECORD_TYPE + 'a')

c means alarm type (value frome enum E_SDK_ALARM_TYPE + 'a'))

012 means record time

.mp4 meas record file type*/

```
 char*    cp_list;
 unsigned int     un_list_len;
}T_SDK_RECORD_DAY_LIST;
```

使用范例：

```
            case SDKCMD_GET_DAY_RECORD_LIST:
            {
                T_SDK_RECORD_DAY_LIST *pData =
    (T_SDK_RECORD_DAY_LIST*)param;
                retcode = sdk_GetDayRecordList(pData);
                if(SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "sdk_GetDayRecordList
    FAILED!\n");
                    return retcode;
                }
                break;
            }
```

注意事项：无。

5.7.8 获取指定时间录像文件列表

命令：SDKCMD_GET_DAY_ASSIGNTIME_RECORD_LIST

参数：

```
 typedef struct
{
 char     filename[32];                    //example:"201401121132231bc0120.mp4"
 unsigned int  direction;          //查找方向,0:向上(向左); 1:向下(向右);
 unsigned int  file_counts;                //需要查找的文件数
 unsigned int  Counts;                     //实际文件数

 /*example:"201401121132231bc0120.mp4|201401121133231bc0120.mp4|"
```

(Output Param. in 201401121132231bc0120z.ts 2014 means year,01 means month ,

12 meas the day,11 means hour, 32 means minute, 23 means second,

1 means the file can not remove when auto circulate record is open,

b means record type (value frome enum RECORD_TYPE + 'a')

c means alarm type (value frome enum E_SDK_ALARM_TYPE + 'a'))

012 means record time

.mp4 meas record file type*/

char* cp_list;

unsigned int un_list_len;

}T_SDK_RECORD_ASSIGNTIME_DAY_LIST;

使用范例：

```
            case SDKCMD_GET_DAY_ASSIGNTIME_RECORD_LIST:
            {
                  T_SDK_RECORD_ASSIGNTIME_DAY_LIST *pData =
(T_SDK_RECORD_ASSIGNTIME_DAY_LIST*)param;
                  retcode = sdk_GetDayAssignTimeRecordList(pData);
                  if(SDK_SUCESS != retcode)
                  {
                        Dbg_Trace(GOS_LOG_ERR,
"sdk_GetDayAssignTimeRecordList FAILED!\n");
                        return retcode;
                  }
                  break;
            }
```

注意事项：无。

5.7.9 获取录像文件的绝对路径

命令：SDKCMD_GET_RECORD_FILE_FULL_PATH

参数：

typedef struct

{

char a_file_name[32]; //example:"201401121132231bc0120.mp4"(input param)

char a_path[128]; //example:"/sdcard/ipc/201401/12/201401121132231bc0120.mp4"

}T_SDK_RECORD_FILE_PATH;

使用范例：

```
           case SDKCMD_GET_RECORD_FILE_FULL_PATH:
           {
               T_SDK_RECORD_FILE_PATH *pData =
    (T_SDK_RECORD_FILE_PATH*)param;
               retcode = sdk_GetRecordFileFullPath(pData);
               if(SDK_SUCESS != retcode)
               {
                   Dbg_Trace(GOS_LOG_ERR, "sdk_GetRecordFileFullPath
    FAILED!\n");
                   return SDK_FAILUR;
               }
               break;
           }
```

注意事项：无。

5.7.10 开启关闭循环录像功能

命令：SDKCMD_SET_LOOP_RECORD_SWITCH

参数：unsigned int un_switch;

使用范例：

```
            case SDKCMD_SET_LOOP_RECORD_SWITCH:
            {
                retcode = sdk_SetLoopRecordSwitch((unsigned int *)param);
                if(SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "sdk_SetLoopRecordSwitch
    FAILED!\n");
                    return SDK_FAILUR;
                }
                break;
            }
```

注意事项：无。

5.7.11 开启关闭是否录制音频

命令：SDKCMD_SET_AUDIO_RECORD_SWITCH

参数：unsigned int un_switch;

使用范例：

```
            case SDKCMD_SET_AUDIO_RECORD_SWITCH:
            {
                retcode = sdk_SetAudioRecordSwitch((unsigned int *)param);
                if(SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "sdk_SetAudioRecordSwitch
    FAILED!\n");
                    return SDK_FAILUR;
                }
                break;
            }
```

注意事项：无。

5.7.12 设置单个录像文件时长

命令：SDKCMD_SET_RECORD_FILE_DURATION

参数：int n_file_duration; //单位s

使用范例：

```
            case SDKCMD SET RECORD FILE DURATION:
            {
                retcode = sdk_SetRecordFileDuartion((unsigned int *)param);
                if(SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "sdk_SetRecordFileDuartion
    FAILED!\n");
                    return SDK_FAILUR;
                }
                break;
            }
```

注意事项：无。

5.7.13 获取单个录像文件时长

命令：SDKCMD_GET_RECORD_FILE_DURATION

参数：unsigned int *pData; //单位s

使用范例：

```
            case SDKCMD_GET_RECORD_FILE_DURATION:
            {
                   unsigned int *pData = (unsigned int *)param;
                   retcode = sdk_GetRecordFileDuartion(pData);
                   if(SDK_SUCESS != retcode)
                   {
                          Dbg_Trace(GOS_LOG_ERR, "sdk_GetRecordFileFullPath
FAILED!\n");
                          return SDK_FAILUR;
                   }
                   break;
            }
```

注意事项：无。

5.7.14 设置录像音视频格式

命令：SDKCMD_SET_RECORD_FILE_TYPE

参数：int n_rec_type; //0:ts 1:mp4 2:flv

使用范例：

```
            case SDKCMD SET RECORD FILE TYPE:
            {
                retcode = sdk_SetRecordFileType((unsigned int *)param);
                if(SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "sdk_SetRecordFileType
    FAILED!\n");
                    return SDK_FAILUR;
                }
                break;
            }
```

注意事项：无。

5.7.15 删除记录文件

命令：SDKCMD_DEL_RECORD_FILE

参数：

```
typedef struct
{
 char  a_file_name[32];
//example:"201401121132231bc0120.mp4"(input param)
    char    a_path[128];
//example:"/sdcard/ipc/201401/12/201401121132231bc0120.mp4"
}T_SDK_RECORD_FILE_PATH;
```

使用范例：

```
            case SDKCMD_DEL_RECORD_FILE:
            {
                T_SDK_RECORD_FILE_PATH *pData = (T_SDK_RECORD_FILE_PATH
    *)param;
                retcode = sdk_DeleteRecordFile(pData);
                if(SDK_SUCESS != retcode)
                {
                    Dbg Trace(GOS LOG ERR, "sdk DeleteRecordFile
    FAILED!\n");
                    return SDK_FAILUR;
                }
                break;
            }
```

注意事项：无。

5.7.16 手动记录开关

命令：SDKCMD_MANUAL_RECORD_SWITCH

参数：unsigned int manual_record_switch; // 0:stop 1:start

使用范例：

```
            case SDKCMD MANUAL RECORD SWITCH:
            {
                retcode = sdk_ManualRecordSwitch((unsigned int *)param);
                if(SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "sdk_ManualRecordSwitch
    FAILED!\n");
                    return retcode;
                }
                break;
            }
```

注意事项：无。

5.7.17 获取储存信息

命令：SDKCMD_GET_STORAGE_INFO

参数：

```
typedef struct
{
 unsigned int a_total_size;   //总容量
 unsigned int a_used_size;    //已用容量
 unsigned int a_free_size;    //未用容量
 unsigned int a_reserve[2];   //保留
}T_SDK_STORAGE_INFO;
```

使用范例：

```
            case SDKCMD_GET_STORAGE_INFO:
            {
                T_SDK_STORAGE_INFO *pData = (T_SDK_STORAGE_INFO *)param;
                retcode = sdk_GetStorageInfo(pData);
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\n GET NETWORK INFO
    FAILED!\n");
                    return SDK_FAILUR;
                }
                 break;
            }
```

注意事项：无。

5.7.18 格式化储存

命令：SDKCMD_FORMAT_STORAGE_REQ

参数：无。

使用范例：

```
            case SDKCMD_FORMAT_STORAGE_REQ:
            {
                retcode = CleanAllRecords();
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\n FORMAT FAILED!\n");
                    return SDK_FAILUR;
                }
                 break;
            }
```

注意事项：无。

5.8 抓拍命令

抓拍模块包括以下命令：

- SDKCMD_SET_SNAPSHOT_PATH：设置抓拍路径

5.8.1 设置抓拍路径

命令：SDKCMD_SET_SNAPSHOT_PATH

参数：

```
typedef struct
{
 char pic_path[128];
}T_SDK_PIC_PATH;
```

使用范例：

```
                case SDKCMD_SET_SNAPSHOT_PATH:
             {
                 T_SDK_PIC_PATH *param_input = (T_SDK_PIC_PATH*)param;
                 retcode = SAMPLE_COMM_VENC_SnapProcess(param_input-
    >pic_path,0);
                 if (HI_SUCCESS != retcode)
                 {
                     Dbg_Trace(GOS_LOG_ERR, "\n
    SAMPLE_COMM_VENC_SnapProcess FAILED!\n");
                 }
                 break;
             }
```

```

```

注意事项：无。

5.9 升级命令

升级模块包括以下命令：

- SDKCMD_UPGRADE_BYLOCAL：根据本地升级包升级
- SDKCMD_UPGRADE_BYSELF：下载并且升级

5.9.1 根据本地升级包升级

命令：SDKCMD_UPGRADE_BYLOCAL

参数：

```
typedef struct
{
 unsigned int un_mode;                    //0 -> app; 1 -> fw;    2 ->
kernel;      3 -> uboot
 char a_current_ver[32];                        //current version
 char a_local_path[256];                        //upgrade package path
}T_SDK_UPGRADE_BYLOCAL_PARAMS;
```

使用范例：

```
            case SDKCMD_UPGRADE_BYLOCAL://根据本地升级包升级
            {
           T_SDK_UPGRADE_BYLOCAL_PARAMS* param_input =
(T_SDK_UPGRADE_BYLOCAL_PARAMS*)param;
                  retcode = GOS_SDK_Upgrade_Bylocal(param_input);
                  if (0 != retcode)
                        return SDK_FAILUR;
                  break;
            }
```

注意事项：无。

5.9.2 下载并且升级

命令：SDKCMD_UPGRADE_BYSELF

参数：

```
typedef struct
{
 char  a_ipaddr[128];    //ip address
```

```
 unsigned int      un_port;    //ip port
 unsigned int result;   //0 -> create upgrade thread success; 1-
>malloc failed;  2 -> create thread failed; 3 -> is upgrading
 unsigned int remoteFlag;     //是否是远程升级     0:局域网升级;     1:服务
器升级;
 unsigned int cancelFlag;     //取消升级标志默认为0不取消1时取消升级
}T_SDK_UPGRADE_BYSELF_PARAMS;
```

使用范例：

```
            case SDKCMD_UPGRADE_BYSELF: //下载并且升级
                  {
                        T_SDK_UPGRADE_BYSELF_PARAMS* param_input =
(T_SDK_UPGRADE_BYSELF_PARAMS*)param;
                        retcode = GOS_SDK_Upgrade_Start(param_input);
                        if(0 != retcode)
                              return SDK_FAILUR;
                        break;
                  }
```

注意事项：无。

5.10 网络命令

网络模块包括以下命令：

- SDKCMD_SET_NETWORK_DEFAULT_PARAM：设置网络默认参数
- SDKCMD_SET_WIRELESS_PARAM：设置无线参数
- SDKCMD_SET_NETWORK_DHCP：设置网络是否启用DHCP
- SDKCMD_SET_STATIC_IPADDR_PARAM：设置网络ip address 参数
- SDKCMD_SET_DDNS_PARAM：设置DDNS参数
- SDKCMD_SET_STATIC_DNS_PARAM：设置DNS参数
- SDKCMD_SET_NTP_PARAM：设置NTP参数
- SDKCMD_SET_STATIC_NETGATEWAY_PARAM：设置NETGATEWAY参数
- SDKCMD_SET_STATIC_NETMASK_PARAM：设置静态NETMASK参数
- SDKCMD_SET_MAC_PARAM：设置mac地址
- SDKCMD_SET_HOSTNAME：设置主机名
- SDKCMD_GET_NETWORK_PARAM：获取网络参数
- SDKCMD_GET_NTP_PARAM：获取NTP参数
- SDKCMD_SEARCH_SSID_NEARBY：获取搜索到的SSID信息
- SDKCMD_GET_NVR_IP_ADDR：获取NVR IP地址
- SDKCMD_SET_NVR_IP_ADDR：设置NVR IP地址
- SDKCMD_GET_SERVER_INFO：获取服务器信息

5.10.1 设置网络默认参数

命令：SDKCMD_SET_NETWORK_DEFAULT_PARAM
参 数 ： 无。
使用范例：

```
            case SDKCMD_SET_NETWORK_DEFAULT_PARAM:
            {
                retcode = sdk_networkDefaultParam();
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\n SET network default param
    fail!\n");
                    return SDK_FAILUR;
                }
                 break;
            }
```

注意事项：无。

5.10.2 设置无线参数

命令：SDKCMD_SET_WIRELESS_PARAM

参数：

```
typedef enum _E_SDK_WIFI_MODE
{
 E_SDK_WIFI_AP_MODE = 0x00,
 E_SDK_WIFI_CLIENT_MODE,
 E_SDK_WIFI_AP_CLIENT_MODE,
 E_SDK_WIFI_WPS_MODE,
 E_SDK_WIFI_MODE_NUM
}E_SDK_WIFI_MODE;
typedef enum _E_SDK_NET_STATUS
{
 E_SDK_DISCONNECT,
 E_SDK_CONNECT,
 E_SDK_SETTING,
 E_SDK_CONNECTING,
}E_SDK_NET_STATUS;
typedef struct
{
 E_SDK_WIFI_MODE  e_wifi_mode;
 E_SDK_NET_STATUS e_wifi_connect_status;
 char  a_SSID[64];
```

```
 char  a_passwd[64];
 unsigned int      un_encrypt;
}T_SDK_WIRELESS_PARAMS;
```

使用范例：

```
        case SDKCMD_SET_WIRELESS_PARAM:
            {
                T_SDK_WIRELESS_PARAMS *pData =
    (T_SDK_WIRELESS_PARAMS*)param;
                retcode = sdk_networkWireless(pData);
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK Wireless
    FAILED!\n");
                    return SDK_FAILUR;
                }
                 break;
            }
```

注意事项：无。

5.10.3 设置网络是否启用DHCP

命令：SDKCMD_SET_NETWORK_DHCP

参数：unsigned int dhcp;　　//0:static 1:dhcp

使用范例：

```
            case SDKCMD_SET_NETWORK_DHCP:
            {
                retcode = sdk_networkDHCP((unsigned int *)param);
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\n SET DHCP FAILED!\n");
                    return SDK_FAILUR;
                }
                 break;
            }
```

注意事项：无。

5.10.4 设置网络ip address 参数

命令：SDKCMD_SET_STATIC_IPADDR_PARAM

参数：

typedef struct

{

  char　　a_IP[64];

}T_SDK_IPADDR;

使用范例：

```
            case SDKCMD_SET_STATIC_IPADDR_PARAM:
            {
                T_SDK_IPADDR *pData = (T_SDK_IPADDR*)param;
                retcode = sdk_networkIpAddress(pData);
```

```
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK IP ADDRESS
    FAILED!\n");
                    return SDK_FAILUR;
                }
                 break;
```

注意事项：无。

5.10.5 设置DDNS参数

命令：SDKCMD_SET_DDNS_PARAM

参数：

```
typedef struct
{
    unsigned int  ddns_swicth;    //value range(1:enable 0:disable)
 char  a_username[50];
 char  a_password[50];
 char  a_server[128];
 char  a_domain[128];
 unsigned int     un_protocol_type;      //value range(:0: Oray
protocol 1:DYNDNS protocol)
}T_SDK_DDNS;
```

使用范例：

```
            case SDKCMD SET DDNS PARAM:
            {
               T_SDK_DDNS *pData = (T_SDK_DDNS*)param;
               retcode = sdk_networkDDNS(pData);
               if( SDK_SUCESS != retcode)
               {
                   Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK DDNS
    FAILED!\n");
                   return SDK_FAILUR;
               }
                break;
            }
```

注意事项：无。

5.10.6 设置DNS参数

命令：SDKCMD_SET_STATIC_DNS_PARAM

参数：

```
typedef struct
{
 char    a_DNS[2][64];
}T_SDK_DNS;
```

使用范例：

```
             case SDKCMD_SET_STATIC_DNS_PARAM:
            {
                T_SDK_DNS *pData = (T_SDK_DNS*)param;
                retcode = sdk_networkDNS(pData);
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK DNS FAILED!\n");
                    return SDK_FAILUR;
                }
                 break;
            }
```

注意事项：无。

5.10.7 设置NTP参数

命令：SDKCMD_SET_NTP_PARAM

参数：

```
//ntp 参数结构设置
typedef struct
{
 unsigned int    un_NtpOpen;                                    //ntp校时开关 (1:开启，
0:关闭)
 unsigned int    un_EuroTime;                                   //夏令时开关 (1:开启，0:关闭)
 unsigned int    un_NtpRefTime;                                 //ntp校时间隔 (单位秒)
 int                        un_TimeZone;                        //时区 (-12~11)
 char            a_NtpServer[64];                //ntp校时服务器地址
 unsigned int    un_ntp_port;                                   //ntp校时服务器端口
 unsigned int    un_Res[2];
}T_SDK_NTP_CFG_PARAMS;
```

使用范例：

```
            case SDKCMD_SET_NTP_PARAM:
            {
                #if 0
                T_SDK_NTP *pData = (T_SDK_NTP*)param;
                retcode = sdk_networkNTP(pData);
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK NTP FAILED!\n");
                    return SDK_FAILUR;
                }
                #endif

                T_SDK_NTP_CFG_PARAMS *pData = (T_SDK_NTP_CFG_PARAMS
    *)param;
                retcode = SDK_SetDeviceNtpParam(pData);
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\n SDKCMD_SET_NTP_PARAM
    FAILED!\n");
                    return SDK_FAILUR;
                }

                 break;
            }
```

注意事项：无。

5.10.8 设置NETGATEWAY参数

命令：SDKCMD_SET_STATIC_NETGATEWAY_PARAM

参数：

typedef struct

{

char a_gateway[64];

}T_SDK_GATEWAY;

使用范例：

```
            case SDKCMD_SET_STATIC_NETGATEWAY_PARAM:
            {
                T_SDK_GATEWAY *pData = (T_SDK_GATEWAY*)param;
                retcode = sdk_networkGateway(pData);
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK GATEWAY
    FAILED!\n");
                    return SDK_FAILUR;
                }
                 break;
            }
```

注意事项：无。

5.10.9 设置静态NETMASK参数

命令：SDKCMD_SET_STATIC_NETMASK_PARAM

参数：

typedef struct

{

char a_mask[64];

}T_SDK_NET_MASK;

使用范例：

```
            case SDKCMD_SET_STATIC_NETMASK_PARAM:
            {
                T_SDK_NET_MASK *pData = (T_SDK_NET_MASK*)param;
                retcode = sdk_networkMask(pData);
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK NETMASK
    FAILED!\n");
                    return SDK_FAILUR;
                }
                 break;
            }
```

注意事项：无。

5.10.10 设置mac地址

命令：SDKCMD_SET_MAC_PARAM

参数：

typedef struct

{

char a_wire_MAC[19];

char a_wireless_MAC[19];

}T_SDK_NET_MAC; // not support

使用范例：

```
                case SDKCMD_SET_MAC_PARAM:
            {
                T_SDK_NET_MAC *pData = (T_SDK_NET_MAC*)param;
                retcode = sdk_networkMac(pData);
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK MAC FAILED!\n");
                    return SDK_FAILUR;
                }
                 break;
            }
```

注意事项：无。

5.10.11 设置主机名

命令：SDKCMD_SET_HOSTNAME

参数：

typedef struct

{

char a_hostname[64];

}T_SDK_HOSTNAME;

使用范例：

```
            case SDKCMD_SET_HOSTNAME:
            {
                T_SDK_HOSTNAME *pData = (T_SDK_HOSTNAME*)param;
                retcode = sdk_networkHostName(pData);
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK HOSTNAME
    FAILED!\n");
                    return SDK_FAILUR;
                }
                 break;
            }
```

注意事项：无。

5.10.12 获取网络参数

命令：SDKCMD_GET_NETWORK_PARAM

参数：

```
typedef enum _E_SDK_WIFI_MODE
{
  E_SDK_WIFI_AP_MODE = 0x00,
  E_SDK_WIFI_CLIENT_MODE,
  E_SDK_WIFI_AP_CLIENT_MODE,
  E_SDK_WIFI_WPS_MODE,
  E_SDK_WIFI_MODE_NUM
}E_SDK_WIFI_MODE;
typedef enum _E_SDK_NET_STATUS
{
  E_SDK_DISCONNECT,
  E_SDK_CONNECT,
  E_SDK_SETTING,
  E_SDK_CONNECTING,
}E_SDK_NET_STATUS;
typedef struct
{
  E_SDK_WIFI_MODE   e_wifi_mode;
  E_SDK_NET_STATUS          e_wifi_connect_status;
  char     a_SSID[64];
  char     a_passwd[64];
  unsigned int     un_encrypt;
}T_SDK_WIRELESS_PARAMS;
typedef struct _T_NTP_CONFIG_S
{
  unsigned int     un_switch;                //value range(0:close 1:open)
  unsigned int     un_cycle;         //value range(1~7*24 hour)
  unsigned int     un_timezone;   //value range(0-12: The west 12 zone - 0 zone ;13-24:East 1
zone - East 12 zone)
  char     a_host[128];
  unsigned int     un_daylight;
}T_SDK_NTP;
typedef struct
{
    unsigned int   ddns_swicth;    //value range(1:enable 0:disable)
  char     a_username[50];
  char     a_password[50];
  char     a_server[128];
  char     a_domain[128];
```

```
 unsigned int     un_protocol_type;      //value range(:0: Oray  protocol 1:DYNDNS protocol)
}T_SDK_DDNS;
typedef struct
{
 unsigned int un_protocals_type;
 T_SDK_WIRELESS_PARAMS          t_wireless_param;
 unsigned int     un_dhcp_switch;                    //0:static 1:dhcp
 char     a_hostname[64];
 char     a_IP[64];
 char     a_DNS[2][64];
 T_SDK_NTP    t_NTP;
 char     a_gateway[64];
 char     a_mask[64];
 char     a_wire_MAC[19];
 char     a_wireless_MAC[19];
 T_SDK_DDNS t_DDNS_info;
}T_SDK_NETWORK_PARAMS;
```

使用范例：

```
            case SDKCMD_GET_NETWORK_PARAM:
            {
                T_SDK_NETWORK_PARAMS *pData = (T_SDK_NETWORK_PARAMS*)param;
                retcode = sdk_getNetworkInfo(pData);
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\n GET NETWORK INFO
    FAILED!\n");
                    return SDK_FAILUR;
                }
                 break;
            }
```

注意事项：无。

5.10.13 获取NTP参数

命令：SDKCMD_GET_NTP_PARAM

参数：

```
//ntp 参数结构设置
typedef struct
{
 unsigned int      un_NtpOpen;                                    //ntp校时开关
(1:开启，  0:关闭)
 unsigned int      un_EuroTime;                                   //夏令时开关
(1:开启，  0:关闭)
```

```
 unsigned int      un_NtpRefTime;                          //ntp校时间隔
(单位秒)
 int                    un_TimeZone;                          //时区
(-12~11)
 char              a_NtpServer[64];              //ntp校时服务器地址
 unsigned int      un_ntp_port;                            //ntp校时服务
器端口
 unsigned int      un_Res[2];
}T_SDK_NTP_CFG_PARAMS;
```

使用范例：

```
            case SDKCMD_GET_NTP_PARAM:
            {
                T_SDK_NTP_CFG_PARAMS *pData = (T_SDK_NTP_CFG_PARAMS
    *)param;
                retcode = SDK_GetDeviceNtpParam(pData);
                if( SDK_SUCESS != retcode)
                {
                    Dbg_Trace(GOS_LOG_ERR, "\n SDKCMD_GET_NTP_PARAM
    FAILED!\n");
                    return SDK_FAILUR;
                }
                 break;
            }
```

注意事项：无。

5.10.14 获取搜索到的SSID信息

命令：SDKCMD_SEARCH_SSID_NEARBY

参数：

```
typedef struct
{
 char  a_SSID[64];
 unsigned int      un_signal_level;
}T_SDK_SEARCH_SSID_INFO;


typedef struct
{
 unsigned int      un_ssid_num;
 T_SDK_SEARCH_SSID_INFO t_info[30];
}T_SDK_SEARCH_SSID_NEARBY;
```

使用范例：

```
        case SDKCMD_SEARCH_SSID_NEARBY:
        {
                    T_SDK_SEARCH_SSID_NEARBY *pData =
(T_SDK_SEARCH_SSID_NEARBY*)param;
                    retcode = sdk_getWirelessSSIDInfo(pData);
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n GET Wireless SSID
INFO!\n");
                           return SDK_FAILUR;
                    }
            break;
        }
```

注意事项：无。

5.10.15 获取NVR IP地址

命令：SDKCMD_GET_NVR_IP_ADDR

参数：char *param_output;

使用范例：

```
             case SDKCMD_GET_NVR_IP_ADDR:
             {
                    char *param_output= (char *)param;
                    retcode = SDK_Get_NVR_IP_Addr(param_output);
                    if (HI_SUCCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n SDKCMD_GET_NVR_IP_ADDR
FAILED!\n");
                    }
                    break;
             }
```

注意事项：无。

5.10.16 设置NVR IP地址

命令：SDKCMD_SET_NVR_IP_ADDR

参数：char *param_input;

使用范例：

```
             case SDKCMD_SET_NVR_IP_ADDR:
             {
                    char *param_input= (char *)param;
                    retcode = SDK_Set_NVR_IP_Addr(param_input);
                    if (HI_SUCCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n SDKCMD_SET_NVR_IP_ADDR
FAILED!\n");
                    }
                    break;
             }
```

注意事项：无。

5.10.17 获取服务器信息

命令：SDKCMD_GET_SERVER_INFO

参数：char *param_input;

使用范例：

```
            case SDKCMD_GET_SERVER_INFO:
            {
                  char *param_input= (char *)param;
                  retcode = SDK_Get_Server_Info(param_input);
                  if (HI_SUCCESS != retcode)
                  {
                        Dbg_Trace(GOS_LOG_ERR, "\n SDK_Get_Server_Info
FAILED!\n");
                  }
                  break;
            }
```

注意事项：无。

5.11 云台命令

云台模块包括以下命令：

- SDKCMD_SET_PTZ_TURN_LEFT：设置云台转向左边
- SDKCMD_SET_PTZ_TURN_RIGHT：设置云台转向右边
- SDKCMD_SET_PTZ_TURN_UP：设置云台转向上
- SDKCMD_SET_PTZ_TURN_DOWN：设置云台转向下
- SDKCMD_SET_PTZ_STOP：设置云台停止
- SDKCMD_SET_PTZ_KEEP_LEFT：设置云台继续向左
- SDKCMD_SET_PTZ_KEEP_RIGHT：设置云台继续向右
- SDKCMD_SET_PTZ_KEEP_UP：设置云台继续向上
- SDKCMD_SET_PTZ_KEEP_DOWN：设置云台继续向下

5.11.1 设置云台转向左边

命令：SDKCMD_SET_PTZ_TURN_LEFT

参数：无。

使用范例：

```
          case SDKCMD_SET_PTZ_TURN_LEFT:
            {
                retcode = SetIpncRS232MotorPtz_S(PTZ_TURN_LEFT);
                if(0 != retcode)
                    return SDK_FAILUR;
               break;
            }
```

注意事项：无。

5.11.2 设置云台转向右边

命令：SDKCMD_SET_PTZ_TURN_RIGHT

参数：无。

使用范例：

```
case SDKCMD_SET_PTZ_TURN_RIGHT:
{
     retcode = SetIpncRS232MotorPtz_S(PTZ_TURN_RIGHT);
     if(0 != retcode)
         return SDK_FAILUR;
    break;
}
```

注意事项：无。

5.11.3 设置云台转向上

命令：SDKCMD_SET_PTZ_TURN_UP

参数：无。

使用范例：

```
case SDKCMD_SET_PTZ_TURN_UP:
{
     retcode = SetIpncRS232MotorPtz_S(PTZ_TURN_UP);
     if(0 != retcode)
         return SDK_FAILUR;
    break;
}
```

注意事项：无。

5.11.4 设置云台转向下

命令：SDKCMD_SET_PTZ_TURN_DOWN

参数：无。

使用范例：

```
case SDKCMD SET PTZ TURN DOWN:
{
     retcode = SetIpncRS232MotorPtz_S(PTZ_TURN_DOWN);
     if(0 != retcode)
         return SDK_FAILUR;
    break;
}
```

注意事项：无。

5.11.5 设置云台停止

命令：SDKCMD_SET_PTZ_STOP

参数：无。

使用范例：

```
    case SDKCMD SET PTZ STOP:
{
     retcode = SetIpncRS232MotorPtz_S(PTZ_SCAN_STOP);
     if(0 != retcode)
         return SDK_FAILUR;
    break;
}
```

注意事项：无。

5.11.6 设置云台继续向左

命令：SDKCMD_SET_PTZ_KEEP_LEFT

参数：无。

使用范例：

```
case SDKCMD_SET_PTZ_KEEP_LEFT:
{
     retcode = SetIpncRS232MotorPtz_S(PTZ_KEEP_TURN_LEFT);
     if(0 != retcode)
         return SDK_FAILUR;
    break;
}
```

注意事项：无。

5.11.7 设置云台继续向右

命令：SDKCMD_SET_PTZ_KEEP_RIGHT

参数：无。

使用范例：

```
 case SDKCMD_SET_PTZ_KEEP_RIGHT:
{
     retcode = SetIpncRS232MotorPtz_S(PTZ_KEEP_TURN_RIGHT);
     if(0 != retcode)
         return SDK_FAILUR;
    break;
}
```

注意事项：无。

5.11.8 设置云台继续向上

命令：SDKCMD_SET_PTZ_KEEP_UP

参数：无。

使用范例：

```
case SDKCMD_SET_PTZ_KEEP_UP:
{
     retcode = SetIpncRS232MotorPtz_S(PTZ_KEEP_TURN_UP);
     if(0 != retcode)
         return SDK_FAILUR;
    break;
}
```

注意事项：无。

5.11.9 设置云台继续向下

命令：SDKCMD_SET_PTZ_KEEP_DOWN

参数：无。

使用范例：

```
            case SDKCMD_SET_PTZ_KEEP_DOWN:
           {
               retcode = SetIpncRS232MotorPtz_S(PTZ_KEEP_TURN_DOWN);
               if(0 != retcode)
                   return SDK_FAILUR;
              break;
           }
```

注意事项：无。

5.12 调试命令

调试模块包括以下命令：

- SDKCMD_SET_DEBUG_LEVEL：设置调试日志级别

5.12.1 设置调试日志级别

命令：SDKCMD_SET_DEBUG_LEVEL

参数：

```
typedef enum
{
  FATAL,
  ERROR,
  WARING,
  NOTE,
  None,
}E_SDK_DEBUG_LEVEL;
```

使用范例：

```
            case SDKCMD_SET_DEBUG_LEVEL://设置DEBUG打印等级
            {
                 E_SDK_DEBUG_LEVEL param_input = (E_SDK_DEBUG_LEVEL)param;
                 retcode = GOS_SDK_Set_Debug_Level(param_input);
                 if(0 != retcode)
                     return SDK_FAILUR;
                break;
            }
```

注意事项：无。

6 数据类型

7 错误码

```
#ifndef _SDK_DEFINE_H_
#define _SDK_DEFINE_H_

#ifdef __cplusplus
extern "C"{
#endif

#define SDK_FAILUR -1
#define SDK_SUCESS  0
```

```
#define SDK_FALSE   0
#define SDK_TRUE    1


/*********** error code ********/
#define ERR_MAP            (SDK_SUCESS+ 1)
#define ERR_DEV            (ERR_MAP   + 1)
#define ERR_INI            (ERR_DEV   + 1)
#define ERR_AUDIOINIT      (ERR_INI   + 1)
#define ERR_VIDEOINIT      (ERR_AUDIOINIT + 1)
#define ERR_IMAGEINIT      (ERR_VIDEOINIT + 1)
#define ERR_RCSTART        (ERR_IMAGEINIT + 1)
#define ERR_BCASTINIT      (ERR_RCSTART   + 1)
#define ERR_NETWORK        (ERR_BCASTINIT + 1)
#define ERR_DISCOVER       (ERR_NETWORK   + 1)
#define ERR_AUDIO          (ERR_DISCOVER  + 1)
#define ERR_VIDEO          (ERR_AUDIO     + 1)
#define ERR_MOTION         (ERR_VIDEO     + 1)
#define ERR_OSD            (ERR_MOTION    + 1)
#define ERR_SYSPROC        (ERR_OSD       + 1)
#define ERR_BIPBUF        (ERR_SYSPROC       + 1)


#define ERR_STREAM         4
#define ERR_INVALID_PARAM  5
#define ERR_NULLPTR        6
#define ERR_NOINITSDK      7
#define ERR_NOTPERM        8
#define ERR_MULTINITSDK    9
#define ERR_NOTSUPPORT     10
#define ERR_INVALIDFILE    11
#define ERR_FILESIZE       12
#define ERR_GETSPS         13
#define ERR_READFILE       14
#define ERR_SD_BUSY        15
#define ERR_NO_SD          16
#define ERR_FILE_UPDATE    17
#define ERR_DEV_MODEL      18
#define ERR_SENSOR_TYPE    19
#define ERR_PWD_LENGTH     20

#if defined (__cplusplus)
}
#endif

#endif //_SDK_DEFINE_H_
```

表1 SDK错误码表

8 范例代码

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include "global.h"
#include "gos_log.h"

#include "sdkout_impl.h"
#include "sdk_sys_api.h"
#include "sdk_define.h"
#include "sdk_commonstruct.h"

#include "api_audio.h"
#include "api_debug.h"
#include "api_osd.h"
#include "api_net.h"
#include "api_system.h"
#include "api_alarm.h"
#include "api_video.h"
#include "api_sensor.h"
#include "api_record.h"
#include <sample_venc.h>
#include "gs_tlib.h"
#include "record.h"
#include "upgrade.h"

#include "pet_feeder.h"
#include "gos_uart.h"

#ifdef __cplusplus
#if __cplusplus
extern "C"{
#endif
#endif /* End of #ifdef __cplusplus */


/*
SDK_sysInit(), SDK_sysRun(), SDK_sysExit() 确保设备顺利跑起来
SDK_treatCmdOnFly() 动态修改设备参数，可能会导致重启
*/


F_SDK_Alarm_Callback alarm_callback_fun = NULL;

int SDK_treatCmdOnFly(unsigned int cmd, void* param)
{
	int retcode = -1;

	switch(cmd)
	{
		//*************系统命令(a1)*****************
		case SDKCMD_GET_DEVICE_INFO://获取设备型信息
		{
			T_SDK_DEVICE_INFO *pt_deviceInfo =
(T_SDK_DEVICE_INFO*)param;
			retcode = SDK_GetDeviceInfo(pt_deviceInfo);
			if( SDK_SUCESS != retcode)
			{
				Dbg_Trace(GOS_LOG_ERR, "\nDEVICE TYPE GET
FAILED!\n");
				return SDK_FAILUR;
			}
			break;
		}
		case SDKCMD_SAVE_ALL_PARAM:
		{
			retcode = SDK_SaveAllParam();
```

```
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\nDEVICE SAVE ALL PARAM
FAILED!\n");
                           return SDK_FAILUR;
                    }
                    break;
              }
        case SDKCMD_REBOOT_DEVICE://设备重启
              {
            Gos_Man_De_St.reboot_flag = 1;
            return SDK_SUCESS;
                    break;
              }
        case SDKCMD_RESET_DEVICE:
        {
            retcode = SDK_ResetDevice();
            if( SDK_SUCESS != retcode)
            {
                Dbg_Trace(GOS_LOG_ERR, "\nDEVICE SET RESET FAILED!\n");
                return SDK_FAILUR;
            }
            break;
        }
        case SDKCMD_SET_DEVICE_TIME:
              {
            T_SDK_DEVICE_TIME *param_output = (T_SDK_DEVICE_TIME*)param;
                    retcode = SDK_SetDeviceTime(param_output);
            if(0 != retcode)
            return retcode;
                    break;
              }

        case SDKCMD_SET_NIGHT_VISION:
        {
                    T_SDK_NIGHT_VISION *pData = (T_SDK_NIGHT_VISION*)param;
                    retcode = SDK_SetNightVision(pData);
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\nSet Night Vision
FAILED!\n");
                           return SDK_FAILUR;
                    }
            break;
        }
              case SDKCMD_GET_NIGHT_VISION:
        {
                    retcode = SDK_GetNightVision((T_SDK_NIGHT_VISION
*)param);
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n SDK_GetNightVision
FAILED!\n");
                           return SDK_FAILUR;
                    }
            break;
        }
        case SDKCMD_SET_CONNECTED_PLATFORM_STATUS:
        {
                    retcode = SDK_ConnectedPlatform((unsigned int *)param);

                    if(SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "SDK_ConnectedPlatform
FAILED!\n");
                           return SDK_FAILUR;
                    }
```

```
                    break;
        }
              case SDKCMD_SET_LED_STATUS:
              {
                    retcode = SDK_SetGPIO((T_SDK_LED *)param);
                    if(SDK_SUCESS != retcode)
                    {
                          Dbg_Trace(GOS_LOG_ERR, "SDKCMD_SET_LED_STATUS
FAILED!\n");
                          return SDK_FAILUR;
                    }
                    break;

              }

        case SDKCMD_SET_DEVICE_AUTHENTICATION:
        {
            T_SDK_DEVICE_AUTHENTICATION_INFO *set_param =
(T_SDK_DEVICE_AUTHENTICATION_INFO*)param;
                    retcode = SDK_Set_Device_Authentication(set_param);

                    if(SDK_SUCESS != retcode)
                    {
                          Dbg_Trace(GOS_LOG_ERR,
"SDK_Set_Device_Authentication FAILED!\n");
                          return SDK_FAILUR;
                    }
                    break;
        }
        case SDKCMD_GET_DEVICE_AUTHENTICATION:
        {
            T_SDK_DEVICE_AUTHENTICATION_INFO *get_param =
(T_SDK_DEVICE_AUTHENTICATION_INFO*)param;
                    retcode = SDK_Get_Device_Authentication(get_param);

                    if(SDK_SUCESS != retcode)
                    {
                          Dbg_Trace(GOS_LOG_ERR,
"SDK_Get_Device_Authentication FAILED!\n");
                          return SDK_FAILUR;
                    }
                    break;
        }
              case SDKCMD_GET_DEVICE_ABILITY:
              {
                    T_SDK_DEVICE_ABILITY_INFO *get_param = NULL ;
                    T_SDK_DEVICE_ABILITY_INFO1 *get_param1 = NULL ;
                    T_SDK_DEVICE_ABILITY_INFO2 *get_param2 = NULL ;
                    switch(SDK_DEVICE_ABILITY_VERSION)
                    {
                          case 0:
                         get_param = (T_SDK_DEVICE_ABILITY_INFO*)param;
                                retcode = SDK_Get_Device_Ability(get_param);

                                if(SDK_SUCESS != retcode)
                                {
                                      Dbg_Trace(GOS_LOG_ERR,
"SDK_Get_Device_Authentication ability0 FAILED!\n");
                                      return SDK_FAILUR;
                                }
                                break;
                          case 1:
                         get_param1 = (T_SDK_DEVICE_ABILITY_INFO1*)param;
                                retcode =
SDK_Get_Device_Ability1(get_param1);
                                if(SDK_SUCESS != retcode)
                                {
```

```
                                        Dbg_Trace(GOS_LOG_ERR,
"SDK_Get_Device_Authentication ability1 FAILED!\n");
                                        return SDK_FAILUR;
                                }
                                break;
                          case 2:
                         get_param2 = (T_SDK_DEVICE_ABILITY_INFO2*)param;
                                retcode =
SDK_Get_Device_Ability2(get_param2);
                                if(SDK_SUCESS != retcode)
                                {
                                        Dbg_Trace(GOS_LOG_ERR,
"SDK_Get_Device_Authentication ability2 FAILED!\n");
                                        return SDK_FAILUR;
                                }
                                break;
                                default:
                                        Dbg_Trace(GOS_LOG_ERR, "not support
this ability search version!\n");
                                        break;
                    }
                    break;
             }
             case SDKCMD_SET_LIGHT_SWITCH:
             {
                    unsigned int *param_input = (unsigned int*)param;
                    retcode = SDK_Set_Light_Switch(*param_input);
            if(0 != retcode)
                return retcode;
                    break;
             }
             case SDKCMD_GET_LIGHT_SWITCH:
             {
                    unsigned int *param_output = (unsigned int*)param;
                    retcode = SDK_Get_Light_Switch(param_output);
            if(0 != retcode)
                return retcode;
                    break;
             }
             case SDKCMD_SET_LIGHT_DURATION:
             {
                    T_SDK_DEVICE_LIGHT_DURATION* param_input =
(T_SDK_DEVICE_LIGHT_DURATION*)param;
                    retcode = SDK_Set_Light_Open_Time(param_input-
>un_trigger_time,param_input->un_manual_time);
            if(0 != retcode)
                return retcode;
                    break;
             }
             case SDKCMD_GET_LIGHT_DURATION:
             {
                    T_SDK_DEVICE_LIGHT_DURATION* param_output =
(T_SDK_DEVICE_LIGHT_DURATION*)param;
                    retcode = SDK_Get_Light_Open_Time(&(param_output-
>un_trigger_time),&(param_output->un_manual_time));
            if(0 != retcode)
                return retcode;
            break;
             }
             case SDKCMD_SET_LIGHT_TIMING_INFO:
             {
                    T_SDK_DEVICE_LIGHT_TIMING *param_input =
(T_SDK_DEVICE_LIGHT_TIMING *)param;

                    retcode = SDK_Set_Light_Timing_Info(param_input);
            if(0 != retcode)
                return retcode;
```

```
                    break;
             }
             case SDKCMD_GET_LIGHT_TIMING_INFO:
             {
                    T_SDK_DEVICE_LIGHT_TIMING *param_output =
(T_SDK_DEVICE_LIGHT_TIMING *)param;

                    retcode = SDK_Get_Light_Timing_Info(param_output);
            if(0 != retcode)
                return retcode;
                    break;
             }
             //*************视频相关(a2)*************
             case SDKCMD_SET_VIDEO_ENCODE_DEFAULT_PARAM:
        {
            retcode = GOS_SDK_SetVideoDefaultParm();
            if(0 != retcode)
                return retcode;
            break;
        }
              case SDKCMD_REGISTER_STREAM_DATA_CALLBACK:
              {
            T_SDK_STREAM_REGISTER_CALLBACK* param_input =
(T_SDK_STREAM_REGISTER_CALLBACK *)param;
                    retcode = sdk_RigisterStreamCallback(param_input);
            if(0 != retcode)
                return retcode;
            break;
              }
        case SDKCMD_FORCE_VIDEO_ENCODE_I_FRAME://强制获取I帧
        {
            T_SDK_FORCE_I_FARME *param_input = (T_SDK_FORCE_I_FARME*)param;
            retcode = GOS_SDK_VENC_RequestIFrame(0,param_input-
>un_encode_channel_id,param_input->un_force_num);
            if(0 != retcode)
                return retcode;
            break;
        }

              case SDKCMD_GET_VIDEO_ENCODE_PARAM://获取编码视频参数
              {
            T_SDK_VIDEO_ENCODE_PARAM *param_inout_parm =
(T_SDK_VIDEO_ENCODE_PARAM*)param;
            retcode = GOS_SDK_Get_VENCParam(param_inout_parm-
>un_encode_channel_id, param_inout_parm);
            if(0 != retcode)
                return retcode;
            retcode = GOS_SDK_Get_ViMirrorMode(param_inout_parm-
>un_encode_channel_id, (unsigned int *)&param_inout_parm->un_mirro_type);
            if(0 != retcode)
                return retcode;
                    break;
              }
        case SDKCMD_SET_VIDEO_ENCODE_SWITCH://设置编码算法
              {
            T_SDK_VIDEO_ENCODE_TYPE *param_input =
(T_SDK_VIDEO_ENCODE_TYPE*)param;
            retcode = GOS_SDK_VENC_SetVENCParam_EncType(param_input-
>un_encode_channel_id, param_input->e_type);
            if(0 != retcode)
                return retcode;
            break;
              }
        case SDKCMD_SET_VIDEO_ENCODE_LEVEL: //编码等级 0: baseline; 1:MP; 2:HP
POE暂不允许修改
        {
```

```
            T_SDK_VIDEO_ENCODE_LEVEL *param_input =
(T_SDK_VIDEO_ENCODE_LEVEL*)param;
            retcode = GOS_SDK_VENC_SetVENCParam_Profile(param_input-
>un_encode_channel_id, param_input->un_profile);
            if(0 != retcode)
                return retcode;
            break;
        }

        case SDKCMD_SET_VIDEO_ENCODE_RESOLUTION://设置编码分辨率
             {
            T_SDK_VIDEO_ENCODE_RESOLUTION *param_input =
(T_SDK_VIDEO_ENCODE_RESOLUTION*)param;
            retcode = GOS_SDK_VENC_SetVENCParam_Resolution(param_input-
>un_encode_channel_id, param_input->un_width,param_input->un_height);
            if(0 != retcode)
                return retcode;
            break;
              }
        case SDKCMD_SET_VIDEO_ENCODE_I_FRAME_INTERVAL://设置编码GOP
             {
            T_SDK_VIDEO_ENCODE_I_FRAME_INTERVAL *param_input =
(T_SDK_VIDEO_ENCODE_I_FRAME_INTERVAL*)param;
            retcode = GOS_SDK_VENC_SetVENCParam_KeyInterval(param_input-
>un_encode_channel_id, param_input->un_interval);
            if(0 != retcode)
                return retcode;
            break;
              }
        case SDKCMD_SET_VIDEO_ENCODE_BITRATE://设置编码码率
             {
            T_SDK_VIDEO_ENCODE_BITRATE *param_input =
(T_SDK_VIDEO_ENCODE_BITRATE*)param;
            retcode = GOS_SDK_VENC_SetVENCParam_Bitrate(param_input);
            if(0 != retcode)
                return retcode;
            break;
              }
        case SDKCMD_SET_VIDEO_ENCODE_FRAMERATE://设置编码帧率
             {
            T_SDK_VIDEO_ENCODE_FRAMERATE *param_input =
(T_SDK_VIDEO_ENCODE_FRAMERATE*)param;
            retcode = GOS_SDK_VENC_SetVENCParam_FrameRate(param_input-
>un_encode_channel_id, param_input->un_framerate);
            if(0 != retcode)
                return retcode;
            break;
              }
        case SDKCMD_SET_VIDEO_ENCODE_QUALITY://设置编码图像质量
             {
            T_SDK_VIDEO_ENCODE_QUALITY *param_input =
(T_SDK_VIDEO_ENCODE_QUALITY*)param;
            retcode = GOS_SDK_VENC_SetVENCParam_Qulity(param_input-
>un_encode_channel_id, param_input->un_quality);
            if(0 != retcode)
                return retcode;
            break;
              }
              case SDKCMD_GET_VIDEO_ENCODE_QUALITY: //获取编码图像质量
              {
            T_SDK_VIDEO_ENCODE_QUALITY *param_output =
(T_SDK_VIDEO_ENCODE_QUALITY*)param;
            retcode = GOS_SDK_VENC_GetVENCParam_Qulity(param_output);
            if(0 != retcode)
                return retcode;
            break;
```

```
            }
        case SDKCMD_SET_VIDEO_ENCODE_QP://设置编码qp等级
            {
            T_SDK_VIDEO_ENCODE_QP *param_input =
(T_SDK_VIDEO_ENCODE_QP*)param;
            retcode = GOS_SDK_VENC_SetVENCParam_Qp(param_input-
>un_encode_channel_id, param_input->un_I_frame_max_Qp,param_input-
>un_I_frame_min_Qp);
            if(0 != retcode)
                return retcode;
            break;
            }
            case SDKCMD_SET_CURR_STREAM_QUALITY: //设置当前码流质量
            {
            unsigned int *param_input = (unsigned int *)param;
            retcode = GOS_SDK_Set_CurrStream_Quality(param_input);
            if(0 != retcode)
                return retcode;
            break;
            }
            case SDKCMD_GET_CURR_STREAM_QUALITY: //获取当前码流质量
            {
            unsigned int *param_output = (unsigned int *)param;
            retcode = GOS_SDK_Get_CurrStream_Quality(param_output);
            if(0 != retcode)
                return retcode;
            break;
            }
            case SDKCMD_SET_SNAPSHOT_QUALITY: //设置抓拍分辨率(0主码流 1次码流)
            {
            unsigned int *param_input = (unsigned int *)param;
            retcode = GOS_SDK_SetSnapQuality(param_input);
            if(0 != retcode)
                return retcode;
                    break;
            }
            case SDKCMD_GET_SNAPSHOT_QUALITY: //获取抓拍分辨率(0主码流 1次码流)
            {
            unsigned int *param_output = (unsigned int *)param;
            retcode = GOS_SDK_GetSnapQuality(param_output);
            if(0 != retcode)
                return retcode;
                    break;
            }
            //*************音频相关(a3)*************

            case SDKCMD_GET_AUDIO_ENCODE_PARAM://获取实时音视频流参数
            {
            T_SDK_AUDIO_ENCODE_PARAM *param_output =
(T_SDK_AUDIO_ENCODE_PARAM*)param;
            retcode = GOS_SDK_AENC_GetAENCParam(param_output);
            if(0 != retcode)
                return retcode;
            break;
            }
        case SDKCMD_SET_AUDIO_ENCODE_SWITCH://设置音频开关
            {
            unsigned int *param_input = (unsigned int *)param;
            retcode = GOS_SDK_AENC_SetAENCParam_un_switch(param_input);
            if(0 != retcode)
                return retcode;
            break;
            }
        case SDKCMD_SET_AUDIO_ENCODE_TYPE://设置音频编码类型
            {
```

```
            unsigned int *param_input = (unsigned int *)param;
            retcode = GOS_AI_Setenc_type(param_input);
            if(0 != retcode)
                return retcode;
            break;
              }
        case SDKCMD_SET_AUDIO_ENCODE_BITRATE://设置音频码率
              {
            unsigned int *param_input = (unsigned int *)param;
            retcode = GOS_AI_Setbitrate(param_input);
            if(0 != retcode)
                return retcode;
            break;
              }
        case SDKCMD_SET_AUDIO_ENCODE_SAMPLERATE://设置音频采样率
              {
            unsigned int *param_input = (unsigned int *)param;
            retcode = GOS_SDK_AENC_SetAENCParam_sample_rate(param_input);
            if(0 != retcode)
                return retcode;
            break;
              }
        case SDKCMD_SET_SOUND_MODE://设置音频声道模式
              {
            unsigned int *param_input = (unsigned int *)param;
            retcode = GOS_SDK_AENC_SetAENCParam_sound_mode(param_input);
            if(0 != retcode)
                return retcode;
            break;
              }
        case SDKCMD_SET_AUDIO_INPUT_VOLUME     ://设置音频音量输入大小
              {
            unsigned int *param_input = (unsigned int *)param;
            retcode = GOS_AI_SetInVol(param_input);
            if(0 != retcode)
                return retcode;
            break;
              }
        case SDKCMD_SET_AUDIO_OUTPUT_VOLUME    ://设置音频音量输出声音大小
              {
            unsigned int *param_input = (unsigned int *)param;
            retcode = GOS_AO_SetOutVol(param_input);
            if(0 != retcode)
                return retcode;
            break;
              }
        case SDKCMD_SET_AUDIO_MIC_LINE://设置音频音量输出模式
              {
            unsigned int *param_input = (unsigned int *)param;
            retcode = GOS_AI_Setmic_line_input(param_input);
            if(0 != retcode)
                return retcode;
            break;
              }
        case SDKCMD_SET_AUDIO_ECHO_CANCELL://设置音频回音消除
              {
            unsigned int *param_input = (unsigned int *)param;
            retcode = GOS_AI_SetAEC(param_input);
            if(0 != retcode)
                return retcode;
            break;
              }
        case SDKCMD_SET_INTERCOM_PARAM:    //设置音频对讲参数
        {
            return SDK_SUCESS;
            break;
```

```
        }
        case SDKCMD_INTERCOM_START:  //开启音频对讲
        {
                    GOS_intercome_start();
            return SDK_SUCESS;
            break;
        }
        case SDKCMD_SEND_INTERCOM_DATA:  //传输音频数据
        {
            T_SDK_INTERCOM_DATA *param_input = (T_SDK_INTERCOM_DATA *)param;
            retcode = GOS_SendTackDate2Adec(param_input->cp_data,param_input-
>un_data_len);
            if(0 != retcode)
                return retcode;
            break;
        }
        case SDKCMD_INTERCOM_STOP:   //关闭对讲
        {
                    GOS_intercome_stop();
            return SDK_SUCESS;
            break;
        }
        //*************sensor相关(a4)*************
        case SDKCMD_SET_VIDEO_ENCODE_MIRROR:  //图像镜像翻转
        {

            unsigned int *param_input = (unsigned int*)param;
            retcode = GOS_SDK_Set_ViMirrorMode(0, *param_input, 1);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
        }
              case SDKCMD_GET_VIDEO_ENCODE_MIRROR:
              {
                     unsigned int *param_output = (unsigned int*)param;
                     retcode = GOS_SDK_Get_ViMirrorMode(0, param_output);
                     if(0 != retcode)
                            return SDK_FAILUR;
                     break;
              }
              case SDKCMD_SET_SENSOR_DEFAULT_PARAM://获取输出图像 亮度 对比度 色度
饱和度 锐度
              {
            T_SDK_SENSOR_PARAM *param_output = (T_SDK_SENSOR_PARAM*)param;
            memset(param_output,0,sizeof(T_SDK_SENSOR_PARAM));
            retcode = GOS_SDK_Vi_Get_CSC(param_output);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
        case SDKCMD_SET_SENSOR_SHARP://设置输出锐度
              {
            T_SDK_SENSOR_SHARP *param_input = (T_SDK_SENSOR_SHARP*)param;
            retcode = GOS_SDK_VPSS_SetChnSpParam(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
        case SDKCMD_SET_SENSOR_BRIGHTNESS://设置输出图像亮度
              {
            unsigned int*param_input = (unsigned int*)param;
            retcode = GOS_SDK_Vi_Set_CSC_Brightness(*param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
```

```
        case SDKCMD_SET_SENSOR_CONTRAST://设置输出图像对比度
              {
            unsigned int*param_input = (unsigned int*)param;
            retcode = GOS_SDK_Vi_Set_CSC_Contrast(*param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
        case SDKCMD_SET_SENSOR_HUE://设置输出图像色度
              {
            unsigned int*param_input = (unsigned int*)param;
            retcode = GOS_SDK_Vi_Set_CSC_Hue(*param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
        case SDKCMD_SET_SENSOR_SATURATION://设置输出图像饱和度
              {
            unsigned int*param_input = (unsigned int*)param;
            retcode = GOS_SDK_Vi_Set_CSC_Satu(*param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
        case SDKCMD_GET_SENSOR_NIGHT_DAY_STATUE://获取当前光敏电阻状态
              {
            unsigned int*param_output = (unsigned int*)param;
            retcode = GOS_SDK_Vi_Get_Ircut(param_output);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
        case SDKCMD_SET_SENSOR_GAMMA_LEVEL://设置当前的GAMMA
              {
            unsigned int*param_input = (unsigned int*)param;
            retcode = GOS_SDK_ISP_SetGammaTable(*param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
        case SDKCMD_SET_SENSOR_SHADING_SWITCH://设置当前的暗角补偿属性
              {
            unsigned int*param_input = (unsigned int*)param;
            retcode = GOS_SDK_ISP_SetShading(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
              case SDKCMD_GET_SENSOR_NTSC_PAL://获取当前视频制式 N:30 P:25
              {
            unsigned int *param_input = (unsigned int*)param;
                    retcode = GOS_SDK_VI_Get_Frame(param_input);
                    if(0 != retcode)
                           return SDK_FAILUR;

                    break;
              }
              case SDKCMD_SET_SENSOR_NTSC_PAL://设置当前视频制式 N:30 P:25
              {
            unsigned int *param_input = (unsigned int*)param;
                    retcode = GOS_SDK_VI_Set_Frame(*param_input);
                    if(0 != retcode)
                           return SDK_FAILUR;

                    break;
              }
```

```
        case SDKCMD_GET_SENSOR_AUTO_EXPOSURE://获取AE
             {
            T_SDK_SENSOR_AUTO_EXPOSURE*param_output =
(T_SDK_SENSOR_AUTO_EXPOSURE*)param;
            memset(param_output,0,sizeof(T_SDK_SENSOR_AUTO_EXPOSURE));
            retcode = GOS_SDK_ISP_GetAEAttrEx(param_output);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
        case SDKCMD_SET_SENSOR_AUTO_EXPOSURE://设置AE
             {
            T_SDK_SENSOR_AUTO_EXPOSURE*param_input =
(T_SDK_SENSOR_AUTO_EXPOSURE*)param;
            retcode = GOS_SDK_ISP_SetAEAttrEx(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
        case SDKCMD_SET_SENSOR_3D://3D 降噪
             {
            T_SDK_SENSOR_3D*param_input = (T_SDK_SENSOR_3D*)param;
            retcode = GOS_SDK_Set_VPSS_3DNr(param_input-
>un_chroma_range,param_input->un_space_denoise,param_input->un_time_denoise);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }

        //*************告警相关(a5)*************
        case SDKCMD_RIGISTER_ALARM_CALLBACK://设置告警回调
             {
            //回调实现
            T_SDK_ALARM_REGISTER_CALLBACK *param_input =
(T_SDK_ALARM_REGISTER_CALLBACK* )param;
            retcode = GOS_SDK_ARLAM_Init(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
        case SDKCMD_GET_ALARM_PARAM://获取告警回调参数
             {
            T_SDK_ALARM_PARAM *param_output = (T_SDK_ALARM_PARAM* )param;
            memset(param_output,0,sizeof(T_SDK_ALARM_PARAM));
            retcode = GOS_SDK_Get_Alarm_Param(param_output);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
        case SDKCMD_SET_PIR_ALARM_PARAM://设置PIR告警回调参数
             {
            T_SDK_PIR_ALARM *param_input = (T_SDK_PIR_ALARM* )param;
            retcode = GOS_SDK_Set_Alarm_Pir_Param(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
        case SDKCMD_SET_MOTION_ALARM_PARAM://设置移动侦测告警回调参数
             {
            T_SDK_VIDEO_MOTION_ALARM *param_input =
(T_SDK_VIDEO_MOTION_ALARM* )param;
            retcode = GOS_SDK_Set_Alarm_Motion_Param(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
```

```
        case SDKCMD_SET_AUDIO_ALARM_PARAM://设置音频告警回调参数
              {
            T_SDK_AUDIO_ALARM *param_input = (T_SDK_AUDIO_ALARM*)param;
            retcode = GOS_SDK_Set_Alarm_Audio_Param(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
        case SDKCMD_GET_AUDIO_ALARM_PARAM://获取音频告警回调参数
              {
            T_SDK_AUDIO_ALARM *param_input = (T_SDK_AUDIO_ALARM*)param;
            retcode = GOS_SDK_Get_Alarm_Audio_Param(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
        case SDKCMD_SET_DOORBELL_ALARM_PARAM://设置按门铃回调参数
              {
            T_SDK_DOORBELL_ALARM *param_input = (T_SDK_DOORBELL_ALARM*)param;
            retcode = GOS_SDK_Set_Alarm_Calling_Param(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
        case SDKCMD_SET_IO_ALARM_PARAM://设置IO回调参数 探头报警
              {
            T_SDK_IO_ALARM *param_input = (T_SDK_IO_ALARM*)param;
            retcode = GOS_SDK_Set_Alarm_Io_Param(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
             case SDKCMD_SET_ONEKEY_ALARM_CONTROL_PARAM:
             {
                    T_SDK_ONEKEY_ALARM_CONTROL *param_input =
(T_SDK_ONEKEY_ALARM_CONTROL* )param;
            retcode = GOS_SDK_Set_OneKey_Alarm_Control_Param(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
             case SDKCMD_GET_ONEKEY_ALARM_CONTROL_PARAM:
             {
                    T_SDK_ONEKEY_ALARM_CONTROL *param_output =
(T_SDK_ONEKEY_ALARM_CONTROL* )param;
            memset(param_output, 0, sizeof(T_SDK_ONEKEY_ALARM_CONTROL));
            retcode = GOS_SDK_Get_OneKey_Alarm_Control_Param(param_output);
            if(0 != retcode)
                return SDK_FAILUR;

                    break;
              }
             case SDKCMD_SET_ALARM_RING_PARAM:
             {
                    T_SDK_ALARM_RING_PARAM *param_input =
(T_SDK_ALARM_RING_PARAM* )param;
            retcode = GOS_SDK_Set_Alarm_Ring_Param(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
             case SDKCMD_GET_ALARM_RING_PARAM:
             {
                    T_SDK_ALARM_RING_PARAM *param_output =
(T_SDK_ALARM_RING_PARAM* )param;
            retcode = GOS_SDK_Get_Alarm_Ring_Param(param_output);
```

```
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
              case SDKCMD_PLAY_ALARM_RING_START:
              {
            retcode = GOS_SDK_Play_Alarm_Ring();
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
              case SDKCMD_PLAY_ALARM_RING_STOP:
              {
            retcode = GOS_SDK_Stop_Play_Alarm_Ring();
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
              case SDKCMD_SET_TEMPERATURE_ALARM_PARAM:
              {
                    T_SDK_TEMPERATURE_ALARM_PARAM *param_input =
(T_SDK_TEMPERATURE_ALARM_PARAM* )param;
            retcode = GOS_SDK_Set_Temperature_Alarm_Param(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
              case SDKCMD_GET_TEMPERATURE_ALARM_PARAM:
              {
                    T_SDK_TEMPERATURE_ALARM_PARAM *param_output =
(T_SDK_TEMPERATURE_ALARM_PARAM* )param;
            memset(param_output, 0, sizeof(T_SDK_TEMPERATURE_ALARM_PARAM));
            retcode = GOS_SDK_Get_Temperature_Alarm_Param(param_output);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
              case SDKCMD_SET_HUMIDITY_ALARM_PARAM:
              {
                    T_SDK_HUMIDITY_ALARM_PARAM *param_input =
(T_SDK_HUMIDITY_ALARM_PARAM* )param;
            retcode = GOS_SDK_Set_Humidity_Alarm_Param(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
              case SDKCMD_GET_HUMIDITY_ALARM_PARAM:
              {
                    T_SDK_HUMIDITY_ALARM_PARAM *param_output =
(T_SDK_HUMIDITY_ALARM_PARAM* )param;
            memset(param_output, 0, sizeof(T_SDK_HUMIDITY_ALARM_PARAM));
            retcode = GOS_SDK_Get_Humidity_Alarm_Param(param_output);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
              case SDKCMD_SET_WBGT_ALARM_PARAM:
              {
                    T_SDK_WBGT_ALARM_PARAM *param_input =
(T_SDK_WBGT_ALARM_PARAM* )param;
            retcode = GOS_SDK_Set_WBGT_Alarm_Param(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
              case SDKCMD_GET_WBGT_ALARM_PARAM:
              {
```

```
                    T_SDK_WBGT_ALARM_PARAM *param_output =
(T_SDK_WBGT_ALARM_PARAM* )param;
            memset(param_output, 0, sizeof(T_SDK_WBGT_ALARM_PARAM));
            retcode = GOS_SDK_Get_WBGT_Alarm_Param(param_output);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
             case SDKCMD_SET_TEMP_HUM_WBGT_ALARM_PARAM:
             {
                    T_SDK_TEMP_HUM_WBGT_ALARM_PARAM *param_input =
(T_SDK_TEMP_HUM_WBGT_ALARM_PARAM* )param;
            retcode = GOS_SDK_Set_TEMP_HUM_WBGT_Alarm_Param(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
             case SDKCMD_GET_TEMP_HUM_WBGT_ALARM_PARAM:
             {
                    T_SDK_TEMP_HUM_WBGT_ALARM_PARAM *param_output =
(T_SDK_TEMP_HUM_WBGT_ALARM_PARAM* )param;
            memset(param_output, 0, sizeof(T_SDK_TEMP_HUM_WBGT_ALARM_PARAM));
            retcode = GOS_SDK_Get_TEMP_HUM_WBGT_Alarm_Param(param_output);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }

             //*************osd相关(a6)*************
        case SDKCMD_SET_OSD_DEFAULT_PARAM://设置OSD默认参数
             {
            retcode = GOS_SDK_Set_Osd_DefualtParm();
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
        case SDKCMD_GET_OSD_PARAM://获取OSD参数
             {
            T_SDK_ALL_OSD_PARAM *param_output = (T_SDK_ALL_OSD_PARAM*)param;
            retcode = GOS_SDK_Get_Osd_Parm(param_output);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
        case SDKCMD_SET_OSD_SHOW_SWITCH://打开或关闭OSD叠加
             {
            T_SDK_OSD_SWITCH *param_input = (T_SDK_OSD_SWITCH*)param;
            retcode = GOS_SDK_OsdShow_Ctrl(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
        case SDKCMD_SET_OSD_COLOR://设置OSD颜色
             {
            T_SDK_OSD_COLOR *param_input = (T_SDK_OSD_COLOR*)param;
            retcode = GOS_SDK_OsdColor_Ctrl(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
        case SDKCMD_SET_OSD_POS://移动OSD位置
             {
            T_SDK_OSD_POS *param_input = (T_SDK_OSD_POS*)param;
            retcode = GOS_SDK_OsdPos_Ctrl(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
```

```
                    break;
              }
        case SDKCMD_SET_OSD_TITLE://设置标题
              {
            T_SDK_OSD_TITLE *param_input = (T_SDK_OSD_TITLE*)param;
            retcode = GOS_SDK_OSD_SetChNameTitle(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
              }
        #if GOS_RECORD

      //*****************************************************************
              //**********模块类型: Record相关(a7)*****************************

      //*****************************************************************

              case SDKCMD_SET_RECORD_DEFAULT_PARAM:
              {
                    retcode = sdk_SetRecordDefaultParam();
                    if(SDK_SUCESS != retcode)
                    {
                          Dbg_Trace(GOS_LOG_ERR, "sdk_SetRecordDefaultParam
FAILED!\n");
                          return SDK_FAILUR;
                    }
                    break;
              }
              case SDKCMD_GET_RECORD_PARAM:
              {
                    T_SDK_RECORD_PARAM *pData = (T_SDK_RECORD_PARAM*)param;
                    retcode = sdk_GetRecordParam(pData);
                    if(SDK_SUCESS != retcode)
                    {
                          Dbg_Trace(GOS_LOG_ERR, "sdk_getRecordParam
FAILED!\n");
                          return SDK_FAILUR;
                    }
                    break;
              }
              case SDKCMD_SET_RECORD_SWITCH:
              {
                    retcode = sdk_SetRecordSwitch((unsigned int *)param);
                    if(SDK_SUCESS != retcode)
                    {
                          Dbg_Trace(GOS_LOG_ERR, "sdk_SetRecordSwitch
FAILED!\n");
                          return SDK_FAILUR;
                    }
                    break;
              }
              case SDKCMD_CLEAR_RECORD_FILE:
              {
                    retcode = sdk_ClearRecordFile();
                    if(SDK_SUCESS != retcode)
                    {
                          Dbg_Trace(GOS_LOG_ERR, "sdk_ClearRecordFile
FAILED!\n");
                          return SDK_FAILUR;
                    }
                    break;
              }
              case SDKCMD_LOCK_UNLOCK_RECORD_FILE:
              {
                    T_SDK_RECORD_LOCK_FILE *pData =
(T_SDK_RECORD_LOCK_FILE*)param;
                    retcode = sdk_LockUnlockRecordFile(pData);
```

```
                    if(SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "sdk_LockUnlockRecordFile
FAILED!\n");
                           return SDK_FAILUR;
                    }
                    break;
             }
             case SDKCMD_GET_MONTH_RECORD_LIST:
             {
                    T_SDK_RECORD_MONTH_LIST *pData =
(T_SDK_RECORD_MONTH_LIST*)param;
                    retcode = sdk_GetMonthRecordList(pData);
                    if(SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "sdk_GetMonthRecordList
FAILED!\n");
                           return retcode;
                    }
                    break;
             }
             case SDKCMD_GET_DAY_RECORD_LIST:
             {
                    T_SDK_RECORD_DAY_LIST *pData =
(T_SDK_RECORD_DAY_LIST*)param;
                    retcode = sdk_GetDayRecordList(pData);
                    if(SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "sdk_GetDayRecordList
FAILED!\n");
                           return retcode;
                    }
                    break;
             }
             case SDKCMD_GET_DAY_ASSIGNTIME_RECORD_LIST:
             {
                    T_SDK_RECORD_ASSIGNTIME_DAY_LIST *pData =
(T_SDK_RECORD_ASSIGNTIME_DAY_LIST*)param;
                    retcode = sdk_GetDayAssignTimeRecordList(pData);
                    if(SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR,
"sdk_GetDayAssignTimeRecordList FAILED!\n");
                           return retcode;
                    }
                    break;
             }
             case SDKCMD_GET_RECORD_FILE_FULL_PATH:
             {
                    T_SDK_RECORD_FILE_PATH *pData =
(T_SDK_RECORD_FILE_PATH*)param;
                    retcode = sdk_GetRecordFileFullPath(pData);
                    if(SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "sdk_GetRecordFileFullPath
FAILED!\n");
                           return SDK_FAILUR;
                    }
                    break;
             }
             case SDKCMD_SET_LOOP_RECORD_SWITCH:
             {
                    retcode = sdk_SetLoopRecordSwitch((unsigned int *)param);
                    if(SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "sdk_SetLoopRecordSwitch
FAILED!\n");
```

```
                        return SDK_FAILUR;
                    }
                    break;
                }
                case SDKCMD_SET_AUDIO_RECORD_SWITCH:
                {
                    retcode = sdk_SetAudioRecordSwitch((unsigned int
*)param);
                    if(SDK_SUCESS != retcode)
                    {
                        Dbg_Trace(GOS_LOG_ERR, "sdk_SetAudioRecordSwitch
FAILED!\n");
                        return SDK_FAILUR;
                    }
                    break;
                }
                case SDKCMD_SET_RECORD_FILE_DURATION:
                {
                    retcode = sdk_SetRecordFileDuartion((unsigned int
*)param);
                    if(SDK_SUCESS != retcode)
                    {
                        Dbg_Trace(GOS_LOG_ERR, "sdk_SetRecordFileDuartion
FAILED!\n");
                        return SDK_FAILUR;
                    }
                    break;
                }
                case SDKCMD_GET_RECORD_FILE_DURATION:
                {
                    unsigned int *pData = (unsigned int *)param;
                    retcode = sdk_GetRecordFileDuartion(pData);
                    if(SDK_SUCESS != retcode)
                    {
                        Dbg_Trace(GOS_LOG_ERR, "sdk_GetRecordFileFullPath
FAILED!\n");
                        return SDK_FAILUR;
                    }
                    break;
                }

                case SDKCMD_SET_RECORD_FILE_TYPE:
                {
                    retcode = sdk_SetRecordFileType((unsigned int *)param);
                    if(SDK_SUCESS != retcode)
                    {
                        Dbg_Trace(GOS_LOG_ERR, "sdk_SetRecordFileType
FAILED!\n");
                        return SDK_FAILUR;
                    }
                    break;
                }
                case SDKCMD_DEL_RECORD_FILE:
                {
                    T_SDK_RECORD_FILE_PATH *pData = (T_SDK_RECORD_FILE_PATH
*)param;
                    retcode = sdk_DeleteRecordFile(pData);
                    if(SDK_SUCESS != retcode)
                    {
                        Dbg_Trace(GOS_LOG_ERR, "sdk_DeleteRecordFile
FAILED!\n");
                        return SDK_FAILUR;
                    }
                    break;
                }
                case SDKCMD_MANUAL_RECORD_SWITCH:
                {
                    retcode = sdk_ManualRecordSwitch((unsigned int *)param);
```

```
                    if(SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "sdk_ManualRecordSwitch
FAILED!\n");
                           return retcode;
                    }
                    break;
             }
             case SDKCMD_GET_STORAGE_INFO:
             {
                    T_SDK_STORAGE_INFO *pData = (T_SDK_STORAGE_INFO *)param;
                    retcode = sdk_GetStorageInfo(pData);
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n GET NETWORK INFO
FAILED!\n");
                           return SDK_FAILUR;
                    }
            break;
             }
             case SDKCMD_FORMAT_STORAGE_REQ:
             {
                    retcode = CleanAllRecords();
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n FORMAT FAILED!\n");
                           return SDK_FAILUR;
                    }
            break;
             }

        #endif

//*********************************************************************
        //**********模块类型: network相关(aa)**************************

//*********************************************************************

        //++设置网络默认参数
        case SDKCMD_SET_NETWORK_DEFAULT_PARAM:
        {
                    retcode = sdk_networkDefaultParam();
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n SET network default
param fail!\n");
                           return SDK_FAILUR;
                    }
            break;
        }

        //++获取搜索到的SSID信息
        #if 1 //目前调用这个接口之后wifi会连接不上，所以先去掉
        case SDKCMD_SEARCH_SSID_NEARBY:
        {
                    T_SDK_SEARCH_SSID_NEARBY *pData =
(T_SDK_SEARCH_SSID_NEARBY*)param;
                    retcode = sdk_getWirelessSSIDInfo(pData);
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n GET Wireless SSID
INFO!\n");
                           return SDK_FAILUR;
                    }
            break;
        }
```

```
            #endif
        //++设置无线参数
        case SDKCMD_SET_WIRELESS_PARAM:
        {
                    T_SDK_WIRELESS_PARAMS *pData =
(T_SDK_WIRELESS_PARAMS*)param;
                    retcode = sdk_networkWireless(pData);
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK Wireless
FAILED!\n");
                           return SDK_FAILUR;
                    }
            break;
        }

        //++设置网络是否启用DHCP
        case SDKCMD_SET_NETWORK_DHCP:
        {
                    retcode = sdk_networkDHCP((unsigned int *)param);
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n SET DHCP FAILED!\n");
                           return SDK_FAILUR;
                    }
            break;
        }

        //++设置网络IP ADDRESS参数
        case SDKCMD_SET_STATIC_IPADDR_PARAM:
        {
                    T_SDK_IPADDR *pData = (T_SDK_IPADDR*)param;
                    retcode = sdk_networkIpAddress(pData);
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK IP ADDRESS
FAILED!\n");
                           return SDK_FAILUR;
                    }
            break;
        }

        //++设置DDNS参数
        case SDKCMD_SET_DDNS_PARAM:
        {
                    T_SDK_DDNS *pData = (T_SDK_DDNS*)param;
                    retcode = sdk_networkDDNS(pData);
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK DDNS
FAILED!\n");
                           return SDK_FAILUR;
                    }
            break;
        }

        //++设置DNS参数
        case SDKCMD_SET_STATIC_DNS_PARAM:
        {
                    T_SDK_DNS *pData = (T_SDK_DNS*)param;
                    retcode = sdk_networkDNS(pData);
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK DNS
FAILED!\n");
                           return SDK_FAILUR;
```

```
                    }
            break;
        }

        //++设置NTP参数
        case SDKCMD_SET_NTP_PARAM:
        {
                    #if 0
                    T_SDK_NTP *pData = (T_SDK_NTP*)param;
                    retcode = sdk_networkNTP(pData);
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK NTP
FAILED!\n");
                           return SDK_FAILUR;
                    }
                    #endif

                    T_SDK_NTP_CFG_PARAMS *pData = (T_SDK_NTP_CFG_PARAMS
*)param;
                    retcode = SDK_SetDeviceNtpParam(pData);
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n SDKCMD_SET_NTP_PARAM
FAILED!\n");
                           return SDK_FAILUR;
                    }

            break;
        }
        case SDKCMD_GET_NTP_PARAM:
        {
                    T_SDK_NTP_CFG_PARAMS *pData = (T_SDK_NTP_CFG_PARAMS
*)param;
                    retcode = SDK_GetDeviceNtpParam(pData);
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n SDKCMD_GET_NTP_PARAM
FAILED!\n");
                           return SDK_FAILUR;
                    }
            break;
        }

        //++设置NETGATEWAY参数
        case SDKCMD_SET_STATIC_NETGATEWAY_PARAM:
        {
                    T_SDK_GATEWAY *pData = (T_SDK_GATEWAY*)param;
                    retcode = sdk_networkGateway(pData);
                    if( SDK_SUCESS != retcode)
                    {
                           Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK GATEWAY
FAILED!\n");
                           return SDK_FAILUR;
                    }
            break;
        }

        //++设置NETMASK参数
        case SDKCMD_SET_STATIC_NETMASK_PARAM:
        {
                    T_SDK_NET_MASK *pData = (T_SDK_NET_MASK*)param;
                    retcode = sdk_networkMask(pData);
                    if( SDK_SUCESS != retcode)
                    {
```

```
                            Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK NETMASK
FAILED!\n");
                            return SDK_FAILUR;
                    }
            break;
        }

        //++设置MAC参数
        case SDKCMD_SET_MAC_PARAM:
        {
                    T_SDK_NET_MAC *pData = (T_SDK_NET_MAC*)param;
                    retcode = sdk_networkMac(pData);
                    if( SDK_SUCESS != retcode)
                    {
                            Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK MAC
FAILED!\n");
                            return SDK_FAILUR;
                    }
            break;
        }

        //++设置主机名
        case SDKCMD_SET_HOSTNAME:
        {
                    T_SDK_HOSTNAME *pData = (T_SDK_HOSTNAME*)param;
                    retcode = sdk_networkHostName(pData);
                    if( SDK_SUCESS != retcode)
                    {
                            Dbg_Trace(GOS_LOG_ERR, "\n SET NETWORK HOSTNAME
FAILED!\n");
                            return SDK_FAILUR;
                    }
            break;
        }

        //++获取网络参数
        case SDKCMD_GET_NETWORK_PARAM:
        {
                    T_SDK_NETWORK_PARAMS *pData =
(T_SDK_NETWORK_PARAMS*)param;
                    retcode = sdk_getNetworkInfo(pData);
                    if( SDK_SUCESS != retcode)
                    {
                            Dbg_Trace(GOS_LOG_ERR, "\n GET NETWORK INFO
FAILED!\n");
                            return SDK_FAILUR;
                    }
            break;
        }
        case SDKCMD_SET_SNAPSHOT_PATH:
        {
            T_SDK_PIC_PATH *param_input = (T_SDK_PIC_PATH*)param;
            retcode = SAMPLE_COMM_VENC_SnapProcess(param_input-
>pic_path,2,0);
            if (HI_SUCCESS != retcode)
            {
                Dbg_Trace(GOS_LOG_ERR, "\n SAMPLE_COMM_VENC_SnapProcess
FAILED!\n");
            }
            break;
        }
             case SDKCMD_GET_NVR_IP_ADDR:
             {
                    char *param_output= (char *)param;
                    retcode = SDK_Get_NVR_IP_Addr(param_output);
```

```
                    if (HI_SUCCESS != retcode)
                    {
                          Dbg_Trace(GOS_LOG_ERR, "\n SDKCMD_GET_NVR_IP_ADDR
FAILED!\n");
                    }
                    break;
             }
             case SDKCMD_SET_NVR_IP_ADDR:
             {
                    char *param_input= (char *)param;
                    retcode = SDK_Set_NVR_IP_Addr(param_input);
                    if (HI_SUCCESS != retcode)
                    {
                          Dbg_Trace(GOS_LOG_ERR, "\n SDKCMD_SET_NVR_IP_ADDR
FAILED!\n");
                    }
                    break;
             }
             case SDKCMD_GET_SERVER_INFO:
             {
                    char *param_input= (char *)param;
                    retcode = SDK_Get_Server_Info(param_input);
                    if (HI_SUCCESS != retcode)
                    {
                          Dbg_Trace(GOS_LOG_ERR, "\n SDK_Get_Server_Info
FAILED!\n");
                    }
                    break;
             }
#if COMM
             //*************PTZ相关(a7)*************
        case SDKCMD_SET_PTZ_TURN_LEFT:
             {
            retcode = SetIpncRS232MotorPtz_S(PTZ_TURN_LEFT);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
        case SDKCMD_SET_PTZ_TURN_RIGHT:
             {
            retcode = SetIpncRS232MotorPtz_S(PTZ_TURN_RIGHT);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
        case SDKCMD_SET_PTZ_TURN_UP:
             {
                    retcode = SetIpncRS232MotorPtz_S(PTZ_TURN_UP);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
        case SDKCMD_SET_PTZ_TURN_DOWN:
             {
                    retcode = SetIpncRS232MotorPtz_S(PTZ_TURN_DOWN);

            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
              case SDKCMD_SET_PTZ_STOP:
             {
            retcode = SetIpncRS232MotorPtz_S(PTZ_SCAN_STOP);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
```

```
             case SDKCMD_SET_PTZ_KEEP_LEFT:
             {
            retcode = SetIpncRS232MotorPtz_S(PTZ_KEEP_TURN_LEFT);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
        case SDKCMD_SET_PTZ_KEEP_RIGHT:
             {
            retcode = SetIpncRS232MotorPtz_S(PTZ_KEEP_TURN_RIGHT);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
        case SDKCMD_SET_PTZ_KEEP_UP:
             {
                    retcode = SetIpncRS232MotorPtz_S(PTZ_KEEP_TURN_UP);

            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
        case SDKCMD_SET_PTZ_KEEP_DOWN:
             {
                    retcode = SetIpncRS232MotorPtz_S(PTZ_KEEP_TURN_DOWN);

            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
#endif
             //*************debug相关(a7)*************
        case SDKCMD_SET_DEBUG_LEVEL://设置DEBUG打印等级
             {
            E_SDK_DEBUG_LEVEL param_input = (E_SDK_DEBUG_LEVEL)param;
            retcode = GOS_SDK_Set_Debug_Level(param_input);
            if(0 != retcode)
                return SDK_FAILUR;
                    break;
             }
             case SDKCMD_UPGRADE_BYLOCAL://根据本地升级包升级
             {
            T_SDK_UPGRADE_BYLOCAL_PARAMS* param_input =
(T_SDK_UPGRADE_BYLOCAL_PARAMS*)param;
                    retcode = GOS_SDK_Upgrade_Bylocal(param_input);
                    if (0 != retcode)
                           return SDK_FAILUR;
                    break;
             }
             case SDKCMD_UPGRADE_BYSELF: //下载并且升级
                    {
                           T_SDK_UPGRADE_BYSELF_PARAMS* param_input =
(T_SDK_UPGRADE_BYSELF_PARAMS*)param;
                           retcode = GOS_SDK_Upgrade_Start(param_input);
                           if(0 != retcode)
                                  return SDK_FAILUR;
                           break;
                    }
             //***************宠物喂食器相关*********************
             case SDKCMD_SET_MANUAL_FEED_PARAM:
             {
                    T_SDK_MANUAL_FEED_PARAM *param_input =
(T_SDK_MANUAL_FEED_PARAM* )param;
                    retcode = GOS_SDK_Set_Manual_Feed_Param(param_input);
                    if (0 != retcode)
```

```
                        return SDK_FAILUR;
                break;
        }
        case SDKCMD_GET_MANUAL_FEED_PARAM:
        {
                T_SDK_FEED_STATUS_PARAM *param_output =
(T_SDK_FEED_STATUS_PARAM* )param;
                memset(param_output, 0, sizeof(T_SDK_FEED_STATUS_PARAM));
                retcode = GOS_SDK_Get_Manual_Feed_Param(param_output);
                if(0 != retcode)
                        return SDK_FAILUR;
                break;
        }
        case SDKCMD_SET_AUTO_FEED_PARAM:
        {
                T_SDK_AUTO_FEED_PARAM *param_input =
(T_SDK_AUTO_FEED_PARAM* )param;
                retcode = GOS_SDK_Set_Auto_Feed_Param(param_input);
                if (0 != retcode)
                        return SDK_FAILUR;
                break;
        }
        case SDKCMD_GET_AUTO_FEED_PARAM:
        {
                T_SDK_AUTO_FEED_PARAM *param_output =
(T_SDK_AUTO_FEED_PARAM* )param;
                memset(param_output, 0, sizeof(T_SDK_AUTO_FEED_PARAM));
                retcode = GOS_SDK_Get_Auto_Feed_Param(param_output);
                if(0 != retcode)
                        return SDK_FAILUR;
                break;
        }
        case SDKCMD_SET_FEED_BOWL_SETTING_PARAM:
        {
                T_SDK_FEED_BOWL_SETTING_PARAM *param_input =
(T_SDK_FEED_BOWL_SETTING_PARAM* )param;
                retcode =
GOS_SDK_Set_Feed_Bowl_Setting_Param(param_input);
                if (0 != retcode)
                        return SDK_FAILUR;
                break;
        }
        case SDKCMD_GET_FEED_BOWL_SETTING_PARAM:
        {
                T_SDK_FEED_BOWL_SETTING_PARAM *param_output =
(T_SDK_FEED_BOWL_SETTING_PARAM* )param;
                memset(param_output, 0,
sizeof(T_SDK_FEED_BOWL_SETTING_PARAM));
                retcode =
GOS_SDK_Get_Feed_Bowl_Setting_Param(param_output);
                if(0 != retcode)
                        return SDK_FAILUR;
                break;
        }
        case SDKCMD_SET_DEDUCT_WEIGHT_PARAM:
        {
                retcode = GOS_SDK_Set_Deduct_Weight_Param();
                if (0 != retcode)
                        return SDK_FAILUR;
                break;
        }
        case SDKCMD_SET_FEED_CALIBRATION_PARAM:
        {
                retcode = GOS_SDK_Set_Feed_Calibration_Param();
                if (0 != retcode)
                        return SDK_FAILUR;
                break;
```

```
		}
		case SDKCMD_SET_BOWL_SWITCH_PARAM:
		{
			T_SDK_FEED_BOWL_SWITCH_PARAM *param_input =
(T_SDK_FEED_BOWL_SWITCH_PARAM* )param;
			retcode =
GOS_SDK_Set_Feed_Bowl_Switch_Param(param_input);
			if (0 != retcode)
				return SDK_FAILUR;
			break;
		}
		/*
		case SDKCMD_GET_FEED_REMAIN_PARAM:
		{
			T_SDK_FEED_REMAIN_PARAM *param_output =
(T_SDK_FEED_REMAIN_PARAM* )param;
			memset(param_output, 0, sizeof(T_SDK_FEED_REMAIN_PARAM));
			retcode=GOS_SDK_Get_Feed_Remain_Param(param_output);
			if (0 != retcode)
				return SDK_FAILUR;
			break;
		}
		*/
		default:
		{
			Dbg_Trace(GOS_LOG_ERR, "\nNot support this CMD!\n");
			break;
		}
	}

	return retcode;
}

int SDK_Cmd_Impl(unsigned int un_cmd, void* p_config_param)
{
	int retcode = -1;

	switch(un_cmd)
	{
		case SDKCMD_SYS_INIT:
		{
			retcode = SDK_sysInit();
			if(retcode)
			{
				Dbg_Trace(GOS_LOG_ERR, "\nSDK_sysInit Error,
ERRCODE: %d\n", retcode);
				return retcode;
			}
			break;
		}

		case SDKCMD_SYS_EXIT:
		{
			retcode = SDK_sysExit();
			if(retcode)
			{
				Dbg_Trace(GOS_LOG_ERR, "\nSDK_sysExit Error,
ERRCODE: %d\n", retcode);
				return retcode;
			}
			break;
		}

		case SDKCMD_SYS_RUN:
		{
			retcode = SDK_sysRun();
```

```
                    if(retcode)
                    {
                            Dbg_Trace(GOS_LOG_ERR, "\nSDK_sysRun Error,
ERRCODE: %d\n", retcode);
                            return retcode;
                    }
                    break;
            }

            default:
            {
                    retcode = SDK_treatCmdOnFly(un_cmd,
(void*)p_config_param);
                    if(retcode)
                    {
                            Dbg_Trace(GOS_LOG_ERR, "SDK_treatCmdOnFly Error,
ERRCODE: %d\n", retcode);
                            return retcode;
                    }
                    break;
            }
      }

      return retcode;

}

#ifdef __cplusplus
#if __cplusplus
}
#endif
#endif /* __cplusplus */
```

9 参考

9.1 分辨率

__encode_res[] =

{

{"4kx3k",4016,3016},

{"4kx2k",4096,2160},

{"4k",4096,2160},

{"qfhd",3840,2160},

{"5m", 2592, 1944},

{"5m_169", 2976, 1674},

{"3m", 2048, 1536},

{"3m_169", 2304, 1296},

{"1080p", 1920, 1080},

{"720p", 1280, 720},

{"480p", 720, 480},

{"576p", 720, 576},

{"4sif", 704, 480},

{"4cif", 704, 576},

```
{"xga", 1024, 768},
{"vga", 640, 480},
{"wvga", 800, 480},
{"fwvga", 854, 480},        //a 16:9 style
{"cif", 352, 288},
{"sif", 352, 240},
{"qvga", 320, 240},
{"qwvga", 400, 240},
{"qcif", 176, 144},
{"qsif", 176, 120},
{"qqvga", 160, 120},
{"svga", 800, 600},
{"sxga", 1280, 1024},
{"480i", 720, 480},
{"576i", 720, 576},
{"1080i", 1920, 1080},

{"", 0, 0},

{"1920x1080", 1920, 1080},
{"1600x1200", 1600, 1200},
{"1440x1080", 1440, 1080},
{"1366x768", 1366, 768},
{"1280x1024", 1280, 1024},
{"1280x960", 1280, 960},
{"1280x720", 1280, 720},
{"1024x768", 1024, 768},
{"720x480", 720, 480},
{"720x576", 720, 576},

{"", 0, 0},

{"704x480", 704, 480},
{"704x576", 704, 576},
{"640x480", 640, 480},
{"352x288", 352, 288},
{"352x256", 352, 256},    //used for interlaced MJPEG 352x256 encoding ( crop to 352x240 by app)
{"352x240", 352, 240},
{"320x240", 320, 240},
{"176x144", 176, 144},
```

```
        {"176x120", 176, 120},
        {"160x120", 160, 120},

        //vertical video resolution
        {"480x640", 480, 640},
        {"480x854", 480, 854},

        //for preview size only to keep aspect ratio in preview image for different VIN aspect ratio
        {"16_9_vin_ntsc_preview", 720, 360},
        {"16_9_vin_pal_preview", 720, 432},
        {"4_3_vin_ntsc_preview", 720, 480},
        {"4_3_vin_pal_preview", 720, 576},
        {"5_4_vin_ntsc_preview", 672, 480},
        {"5_4_vin_pal_preview", 672, 576 },
        {"ntsc_vin_ntsc_preview", 720, 480},
        {"pal_vin_pal_preview", 720, 576},
};
```

9.2 音频采样率

```
static unsigned const samplingFrequencyTable[16] =
{
  96000, 88200, 64000, 48000,
  44100, 32000, 24000, 22050,
  16000, 12000, 11025, 8000,
  7350, 0, 0, 0
};
```