高德红外SGP_SDK开发手册(C++版)

版权声明
技术术支持
武汉高德红外股份有限公司 咨询热线：4008 822 866　（周一至周五 8:40-17:30） 售后服务专线：027-81298738　（周一至周五 8:40-17:30） 微信公众号：GuideSensmart 官方微博：@高德红外 E-mail：marketing@guide-infrared.com 地址：湖北省武汉市东湖开发区黄龙山南路6号 邮编：430205

概述

欢迎使用SGP_SDK开发手册，本文档详细描述了开发包中各个函数实现功能、接口及其函数之间的调用关系和示例实现。

本开发包主要包含业务操作和设备管理两大部分：

业务操作

实时预览、字符叠加、拍照、视频录制、温度获取、报警等功能。

设备管理

设备重启、密码管理、设备参数配置(系统通用参数配置、报警参数配置、图像配置、视频配置、网络配置、电机配置、融合配置)等功能。

Windows下，SDK包括的文件有

功能库	SgpApi.h	SDK对外头文件
	SgpParam.h	结构体定义头文件
	SgpApi.lib	lib功能库
	SgpApi.dll	功能库
依赖库	PocoJSON.dll	
	PocoFoundation.dll	
	avcodec-58.dll	
	avformat-58.dll	
	avutil-56.dll	
	libcrypto-1_1.dll	
	swresample-3.dll	

	swscale-5.dll	
	PocoNet.dll	
	PocoUtil.dll	
	PocoXML.dll	
配置文件	log_conf.properties	
Linux下，SDK包括的文件有		
功能库	SgpApi.h	SDK对外头文件
	SgpParam.h	结构体定义头文件
	libSgpApi.so	功能库
依赖库	libavcodec.so	
	libva.so	
	libva-x11.so	
	libavdevice.so	
	libavfilter.so	
	libavformat.so	
	libavutil.so	
	libva-drm.so	
	ibcrypto.so	
	libPocoFoundation.so	
	libPocoJSON.so	
	libPocoNet.so	
	libPocoUtil.so	
	libPocoXML.so	
	libpostproc.so	
	libc.so	
	libssl.so	
	libswresample.so	

	libswscale.so	
	libx264.so.164	
配置文件	log_conf.properties	
	sgp_sdk.log	SDK生成的日志文件、日志文件按照文件大小自动备份 定期自动删除
本SDK的功能库和依赖库都是必须，缺少依赖库会导致某些功能运行异常。sgp_sdk.log为SDK日志文件。		

SDK支持系统
Windows 32/64位网络SDK： Windows 10/Windows 8/Windows 7以及Windows Server 2012/2008 x86 Linux 32/64位设备网络SDK： 已测系统：CentOS 7、Redhat、Ubuntu 12、Ubuntu 14、Ubuntu 16、 Ubuntu 18、Ubuntu 20 Arm Linux 32/64位设备网络SDK： 需提供交叉编译环境，定制化编译

开发说明	
Window开发环境	SDK： 根据平台选择32位或者64位SDK，SDK包可单放在某个文件夹下或者将SDK 包内容拷贝到应用程序exe同级别目录下。 Demo： GuiderApiTest_IPT工程文件可用QT或者VS2015打开。
Linux开发环境	SDK： 根据平台选择32位或者64位SDK，解压命令tar -xzf lib.tar.gz。开使用的库在SDK\lib\目录下。SDK包可单放在某个文件夹下或者将SDK包内容拷贝到应用程序同级别目录下。 Demo： main.cpp为C语言编写测试示例，可供参考。 GuiderApiTest为QT开发的DEMO，为了能够正常运行，运行Demo需安装QT5.7，

	从QT官网下载qt-opensource-linux-x64-5.7.0.run。安装完成后设置环境变量。QTDIR改成实际安装路径。 export QTDIR=/home/Qt5.7.0 export PATH=$QTDIR/5.7/gcc_64/bin:$PATH export PATH=$QTDIR/Tools/QtCreator/bin:$PATH export LD_LIBRARY_PATH=$QTDIR/5.7/gcc_64/lib:$LD_LIBRARY_PATH 运行Demo 1、拷贝lib.tar.gz到linux某目录，解压tar -xzf lib.tar.gz。 2、设置共享库路径。 用root权限vi /etc/ld.so.conf，在"include ld.so.conf.d/*.conf"下方增加lib路径。 再执行/sbin/ldconfig -v使之生效。 3、关闭防火墙。 systemctl stop firewalld.service 4、切换到普通用户执行./GuiderApiTest或者双击GuiderApiTest运行程序。

修订记录

发布时间	版本号	修订内容
2022.03	V1.1.11	新增
2022.04	V1.1.14	增加电子变倍接口
2022.05	V1.1.15	修改SGP_PORT_INFO等几个结构体声明
2022.06	V1.1.16	增加注册外部告警回调函数、恢复出厂设置函数，增加部分结构体变量
2022.07	V1.1.19	在字符串叠加函数中增加IPT640M机芯显示左上、左下选项
2022.08	V1.2.1	1、增加三个拍照函数 SGP_GetScreenCaptureCache、 SGP_GetHeatMapCache、 SGP_GetFirHeatMapCache
2023.07	V1.2.9	1、增加通过SDK获取Y16数据（SGP_GetY16）进行测温SGP_GetTempMatrixEx 2.拍照等接口缺少对文件路径的校验 3.增加获取测温点数组对应温度值接口 SGP_GetTempPoints 4.增加火灾报警回调函数 SGP_RegisterFireAlarmCallback 5.增加自动调焦回调函数 SGP_RegisterAutoFocusCallback
2023.09.25	V1.2.10	1、增加alarm_interal字段到分析对象结构体SGP_RULE中
2023.10.11	V1.2.14	1、增加show_type字段到分析对象结构体

		SGP_RULE中
2023.12.22	V1.2.15	1、增加注册对象温差告警回调函数SGP_RegisterObjTempAlarmCallback，增加对象温差结构体SGP_OBJTEMPALARMNOTIFY 2、对SGP_GetTempPoints、SGP_SetTempPoints、SGP_GetMatrixTempPoints、SGP_GetPointTemp增加备注，指出接口的使用注意事项
2024.01.26	V1.2.16	1、更新 SGP_SetTempPoints 接口使用备注信息

流程说明：

创建实例会返回实例序号值，之后的所有函数都需要用到这个序号值。

操作任何业务之前首先要登陆平台

创建实例（SGP_InitDevice）:生成一个SGPSDK的独立实例，之后所有函数操作都作用于这个实例。可多次调用此函数生成多个实例，多个实例之间的操作互不影响。

登陆平台（SGP_Login）：实现用户登录服务器功能。平台中设置用户为复用时，在可以同时间多次登陆。

报警业务：布控之后如果有报警触发，则会有报警事件触发。详见报警业务。

登出平台（SGP_Logout）:登出平台

销毁实例（SGP_UnInitDevice）：

流程说明：

创建实例会返回实例序号值，之后的所有函数都需要用到这个序号值。

操作任何业务之前首先要登陆到平台

创建实例（SGP_InitDevice）:生成一个SGPSDK的独立实例，之后所有函数操作都作用于这个实例。可多次调用此函数生成多个实例，多个实例之间的操作互不影响。

登陆平台（SGP_Login）：实现用户登录服务器功能。平台中设置用户为复用时，在可以同时间多次登陆。

拍照业务：调用SGP_GetHeatMap函数获得jpeg国网格式热图，调用SGP_GetFirHeatMap获得FIR格式国网格式热图。

登出平台（SGP_Logout）:登出平台

销毁实例（SGP_UnInitDevice）：

连接设备接口

接口描述	功能描述	功能详细描述
SGP_InitDevice	初始化一个设备对象	
SGP_UnInitDevice	释放设备对象	
SGP_Login	用户登录	
SGP_Logout	用户登出	
SGP_GetGeneralInfo	获取通用信息	
SGP_ChangePassword	修改密码	
SGP_ResetPassword	重置密码	

初始化一个设备对象SGP_InitDevice

选项：	说明
描述：	初始化一个设备对象
详细描述：	
函数：	SGP_HANDLE SGP_InitDevice();
参数：	[in] 无
返回值：	成功 返回设备对象句柄
备注：	首次调用的函数，与SGP_UnInitDevice成对使用。同一台设备最多支持20路访问
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象句柄。 **/ void Init() { SGP_HANDLE handle = 0;

```
      handle =
SGP_InitDevice();
      if (handle)
      {
          //成功，TODO......

      }
      else
      {
          //失败，TODO......
      }
 }
```

释放设备对象SGP_UnInitDevice

选项：	说明
描述：	释放设备对象
详细描述：	
函数：	void SGP_UnInitDevice(SGP_HANDLE handle);
参数：	handle [in] 设备对象句柄
返回值：	无
备注：	与SGP_InitDevice成对使用

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象句柄。
	**/
	void Uninit()
	{
	SGP_UnInitDevice(handle);
	}

用户登录SGP_Login

选项：	说明
描述：	用户登录
详细描述：	需要登录成功以后才能访问其他接口
函数：	int SGP_Login(SGP_HANDLE handle, const char *server, const char *username, const char *password, int port);
参数：	handle [in] 设备对象句柄 server [in] 设备ip地址，设备ip初始默认为“192.168.1.168” username [in] 用户名，管理员账号默认为“admin” password [in] 密码，admin账号密码默认为“admin123”，明文方式 port [int] 端口，端口默认为80
返回值：	成功返回SGP_OK,失败返回错误码

备注：	需要登录成功以后才能访问其他接口，与SGP_Logout成对使用

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle　设备对象句柄。
	**/
	void Init()
	{
	SGP_HANDLE handle = 0;
	handle = SGP_InitDevice();
	if (handle)
	{
	const char *server = "192.168.1.168";
	const char *username = "admin";
	const char *password = "admin123";
	int port = 80;
	int ret = SGP_Login(handle, server, username, password, port);
	if (ret == SGP_OK)
	{
	//登录成功，TODO......
	}
	else
	{
	SGP_UnInitDevice(handle)
	//登录失败，TODO......

```
            }
        }
        else
        {
            //失败，TODO......
        }
}
```

用户登出SGP_Logout

选项：	说明
描述：	用户登出
详细描述：	
函数：	int SGP_Logout(SGP_HANDLE handle);
参数：	handle [in] 设备对象句柄
返回值：	成功返回SGP_OK,失败返回错误码
备注：	与SGP_Login成对使用

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象句柄。
	**/
	void Init()
	{
	int ret =
	SGP_Logout(handle);
	if (ret == SGP_OK)
	{
	//成功，TODO.....
	}
	else
	{
	//失败，TODO.....

```
        }
}
```

修改密码SGP_ChangePassword

选项：	说明
描述：	修改密码
详细描述：	
函数：	int SGP_ChangePassword(SGP_HANDLE handle, const char *username, const char *oldpassword, const char *newpassword);
参数：	handle [in] 设备对象句柄 username [in] 登录用户名 oldpassword [in] 旧密码 明文 newpassword [in] 新密码 明文
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象句柄。 **/ void Init() { const char* username = "admin"; const char* oldpassword = "admin123"; const char* newpassword = "admin567";

```
      int ret =
SGP_ChangePassword(m_handle,username,oldpassword,newpassword);
      if (ret ==SGP_OK）
      {
          //成功，TODO......
      }
      else
      {
          //失败，TODO......

      }
 }
```

获取通用信息SGP_GetGeneralInfo

选项:	说明
描述:	获取通用信息
详细描述:	
函数:	int SGP_GetGeneralInfo(SGP_HANDLE handle, SGP_GENERAL_INFO *output);
参数:	handle [in] 设备对象句柄 output [out] 输出信息，获取的通用信息
返回值:	成功返回SGP_OK，失败返回错误码
备注:	结构体变量在使用前先初始化
使用示例:	/** 示例中部分类、变量、函数的解释: 1，handle 设备对象句柄。 **/ void Init() {

```
     SGP_GENERAL_INFO info;
     memset(&info, 0x00,
sizeof(SGP_GENERAL_INFO));
     int ret =
SGP_GetGeneralInfo(handle,&info);
     if (ret ==SGP_OK）
     {
        //成功，TODO......
     }
     else
     {
        //失败，TODO......
     }
 }
```

重置密码SGP_ResetPassword

选项：	说明
描述：	重置密码
详细描述：	
函数：	int SGP_ResetPassword(SGP_HANDLE handle, const char *username);
参数：	handle [in] 设备对象句柄 username [in] 登录用户名
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象句柄。 **/ void Init() {

```
        const char* username =
"admin";
        int ret =
SGP_ResetPassword(m_handle,username);
        if (ret ==SGP_OK）
        {
            //成功，TODO......
        }
        else
        {
            //失败，TODO......
        }
 }
```

测温接口

接口描述	功能描述	功能详细描述
SGP_GetPointTemp	获取单点温度	
SGP_GetAnalyticObjectsTemp	获取分析对象温度	
SGP_GetImageTemps	获取温度矩阵	
SGP_GetTempMatrix	获取温度矩阵（医疗机芯有效）	从设备端获取温度

		矩阵
SGP_SetTempPoints	设置测温点索引数组	
SGP_GetTempPoints	获取测温点温度数组	
SGP_GetMatrixTempPoints	获取指定矩形区域温度矩阵	
SGP_GetTempMatriRotation	Y16旋转	
SGP_GetTempMatriRotationEx	温度矩阵旋转	
SGP_GetY16	获取Y16	
SGP_StopY16	停止获取Y16	
SGP_GetTempMatrixEx	Y16输出温度矩阵	
SGP_GetMeasureTempInfo	获取校温	

	信息	

获取单点温度SGP_GetPointTemp

选项：	说明
描述：	获取单点温度
详细描述：	
函数：	int SGP_GetPointTemp(SGP_HANDLE handle, int x, int y, float *output);
参数：	handle [in] 传入设备对象 x [in] 横坐标，范围在1到图像宽度之间 y [in] 纵坐标，范围在1到图像高度之间 output [out] 点温度
返回值：	成功返回SGP_OK,失败返回错误码
备注：	1、要确保传入的动态数组output对象不为空，否则可能会引发异常

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle　设备对象。
	**/
	void Init()
	{
	　　int x = 100;
	　　int y = 100;
	　　float output=0.0f;
	　　int ret = SGP_GetPointTemp(handle,x,y,&output);
	　　if (ret == SGP_OK)
	　　{
	　　　　//成功，TODO......
	　　}
	　　else
	　　{
	　　　　//失败，TODO......
	　　}
	}

获取分析对象温度
SGP_GetAnalyticObjectsTemp

选项：	说明
描述：	获取分析对象温度
详细描述：	
函数：	int SGP_GetAnalyticObjectsTemp(SGP_HANDLE handle, SGP_ANALYTIC_TEMPS *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
     SGP_ANALYTIC_TEMPS array;
     memset(&array, 0x00,
sizeof(SGP_ANALYTIC_TEMPS));
     int ret =
SGP_GetAnalyticObjectsTemp(handle,&array);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取温度矩阵SGP_GetImageTemps

选项：	说明
描述：	获取温度矩阵
详细描述：	
函数：	int SGP_GetImageTemps(SGP_HANDLE handle, float *output, int length, int type);
参数：	handle [in] 传入设备对象 output [out] 输出温度矩阵 length [in] output大小 type [in] 0为推流红外分辨率，1为设备红外原始分辨率
返回值：	成功返回SGP_OK,失败返回错误码
备注：	1、温度矩阵值为float类型，4个字节长度，调用SGP_GetGeneralInfo函数获取红外模组宽和

	红外模组高，第三个参数length值传入红外模组宽*红外模组高*4

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle　设备对象。
	**/
	void Init()
	{
	SGP_GENERAL_INFO info;
	memset(&info, 0x00, sizeof(info));
	int ret = SGP_GetGeneralInfo(handle,&info);
	if (ret == SGP_OK)
	{
	int heigth = info.ir_model_h;
	int width = info.ir_model_w;
	int length = heigth*width;
	int type = 1;
	float *output = (float *)calloc(length, sizeof(float));
	if(output!=NULL)
	{
	ret = SGP_GetImageTemps(handle,output,length*4,type);
	if (ret == SGP_OK)
	{
	//成功，TODO......
	}
	else
	{
	//失败，TODO......
	}
	}
	free(output);

```
        output=NULL;
      }
}
```

获取温度矩阵（医疗机芯有效-仅ZU13A支持）
SGP_GetTempMatrix

选项：	说明
描述：	获取温度矩阵（医疗机芯有效）
详细描述：	医疗机芯有效
函数：	int SGP_GetTempMatrix(SGP_HANDLE handle, SGP_TEMPCALLBACK callback, void *pUser);
参数：	handle [in] 传入设备对象 callback [in] 注册温度矩阵回调函数（Short数据，温度*100倍） pUser [in] 回调函数入参
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释：

```
 1，handle  设备对象。
 **/
 static void TempCall(short
*temp, int w, int h, void
*pUser)
 {
     MainWindow *pDlg =
(MainWindow *)pUser;
     //TODO......
 }
 void MainWindow::Init()
 {
     int ret =
SGP_GetTempMatrix(handle,
TempCall, this);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }

 }
```

回调函数描述	
函数	typedef

名称	void(*SGP_TEMPCALLBACK)(short *temp, int w, int h, void *pUser);	
功能描述	温度矩阵回调函数	
参数说明	temp	输出参数，温度矩阵数据
	w	输出参数，温度矩阵宽
	h	输出参数，温度矩阵高
	pUser	输出参数
返回值	无	

设置测温点索引数组 SGP_SetTempPoints

选项：	说明
描述：	设置测温点索引数组(该接口当前仅ZU08D支持)
详细描述：	
函数：	int SGP_SetTempPoints (SGP_HANDLE handle, int *index , int length, int type);
参数：	handle [in] 传入设备对象 index [in] 传入测温点索引数组 length [in] index大小 type [in] 0为推流红外分辨率，1为设备红外原始分辨率
返回值：	成功返回SGP_OK,失败返回错误码
备注：	1、传入测温点的索引（将二维图像矩阵转换为一维矩阵），测温索引数组大小即为测温点个数。调用此接口后，设备端会缓存此数组。 2、实际传入的测温点不能超过当前设备的分辨率宽高乘积的最大值，否则，得到的数据也没有实际意义的。

	3、调用SGP_GetTempPoints即可返回对应索引的温度值，无需反复设置。 4、要确保传入的测温点索引数组的长度和length的大小一致，否则可能会引发异常 5、如果需要请求全图温度矩阵，建议调用SGP_GetImageTemps接口。

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle　设备对象。
	**/
	const int pointNum = 100;
	void Init()
	{
	//开辟长度为100的数组，并填充100个点的数组索引进行赋值
	int *index = (int *)malloc(pointNum * sizeof(int));
	for (int i = 0; i < pointNum; i++)
	{
	index[i] = i * 100;
	}
	
	int ret = SGP_SetTempPoints(m_handle, index, pointNum, 1);
	
	if (ret == SGP_OK)
	{
	//成功，TODO.....
	}
	else

	{
	//失败，TODO.....
	}
	
	free(index);
	index =NULL;
	
	}

获取测温点温度数组 SGP_GetTempPoints

选项：	说明
描述：	获取测温点温度数组（该接口当前仅ZU08D支持）
详细描述：	
函数：	int SGP_GetTempPoints (SGP_HANDLE handle, float *output , int length, int type);
参数：	handle [in] 传入设备对象 output [out] 输出测温点数组对应的温度值数组 length [in] output大小 type [in] 0为推流红外分辨率，1为设备红外原始分辨率
返回值：	成功返回SGP_OK,失败返回错误码
备注：	1、温度矩阵值为float类型，4个字节长度 2、第三个参数length值传入SGP_SetTempPoints接口的索引数组长度 3、要确保传入的测温点数组的长度和length的大小一致，否则可能会引发异常
使用示例：	/** 示例中部分类、变量、函数的解释：

```
 1，handle   设备对象。
 **/
const int pointNum = 100;
void Init()
{
    //开辟长度为100的数组，并填充100个点的数组索引进行赋值
    int *index = (int *)malloc(pointNum * sizeof(int));
    for (int i = 0; i < pointNum; i++)
    {
        index[i] = i * 100;
    }

    int ret = SGP_SetTempPoints(m_handle, index, pointNum, 1);
     if (ret == SGP_OK )
     {
        float *temp = (float *)malloc(pointNum * sizeof(float));
        memset(temp, 0, pointNum * sizeof(float));
        if(temp != NULL)
        {
           int ret = SGP_GetTempPoints(m_handle, temp, pointNum, 1 );
           if (ret == SGP_OK )
           {
               //成功，TODO......
           }
           else
           {
               //失败，TODO......
           }
```

	}
	free(temp);
	temp =NULL;
	}
	free(index);
	index = nullptr;
	}

获取指定矩形区域温度矩阵 SGP_GetMatrixTempPoints

选项：	说明
描述：	获取指定矩形区域温度矩阵（该接口当前仅ZU08D支持）
详细描述：	
函数：	int SGP_GetMatrixTempPoints(SGP_HANDLE handle, float *output , int length, const SGP_RECT &input);
参数：	handle [in] 传入设备对象 output [out] 输出测温点数组对应的温度值数组 length [in] output大小 input [in] 输入参数，指定的矩形区域
返回值：	成功返回SGP_OK,失败返回错误码
备注：	1、要确保测温点数组对应数组的长度和length的大小一致，否则可能会引发异常
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/

```
 void Init()
 {
    float *output = new float[100];
    SGP_RECT rect;
    rect.x = 0;
    rect.y = 0;
    rect.w = 20;
    rect.h = 5;
    const int length = 100;
    int ret =
SGP_GetMatrixTempPoints(m_handle, output,
length, rect);
    if (ret == SGP_OK )
    {
        //成功，TODO......
    }
    else
    {
        //失败，TODO......
    }

    delete[] output;
    output = nullptr;
 }
```

温度矩阵旋转 SGP_GetTempMatriRotation

选项：	说明
描述：	温度矩阵旋转
详细描述：	从设备端获取温度矩阵
函数：	int SGP_GetTempMatriRotation(SGP_HANDLE handle, short *dst, short *src, int w, int h, int rotation);
参数：	handle [in] 传入设备对象 dst [out] 输出旋转后的温度矩阵 src [in] 输入需要旋转的温度矩阵 w [in] 输入src的宽 h [in] 输入src的高 rotation [in] 0：旋转90，1：旋转180°，2：旋转270°
返回值：	成功返回SGP_OK,失败返回错误码
备注：	温度矩阵值为short类型，2个字节长度。
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。

```
 **/
 void Init()
 {
       //通过SGP_GetTempMatrix回调的温度矩阵传入接口
SGP_GetTempMatriRotation
       获取旋转后的温度矩阵。
       int rotation = 1;
       ret =
SGP_GetTempMatriRotation(handle,dst,src,width,height,rotation);
       if (ret == SGP_OK )
       {
           //成功，TODO......
       }
       else
       {
           //失败，TODO......
       }
 }
```

温度矩阵旋转 SGP_GetTempMatriRotationEx

选项：	说明
描述：	温度矩阵旋转
详细描述：	从设备端获取温度矩阵
函数：	int SGP_GetTempMatriRotationEx(SGP_HANDLE handle, float* dst, float* src, int w, int h, int rotation);
参数：	handle [in] 传入设备对象 dst [out] 输出旋转后的温度矩阵 src [in] 输入需要旋转的温度矩阵 w [in] 输入src的宽 h [in] 输入src的高 rotation [in] 0：旋转90，1：旋转180°，2：旋转270°
返回值：	成功返回SGP_OK,失败返回错误码
备注：	温度矩阵值为short类型，2个字节长度。
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/

```
void Init()
{
	//通过SGP_GetTempMatrix回调的温度矩阵传入接口
SGP_GetTempMatriRotationEx
	获取旋转后的温度矩阵。
	int rotation = 1;
	ret =
SGP_GetTempMatriRotationEx(handle,dst,src,width,height,rotation);
	if (ret == SGP_OK )
	{
	    //成功，TODO......
	}
	else
	{
	    //失败，TODO......
	}
}
```

获取Y16 SGP_GetY16

选项：	说明
描述：	获取Y16数据
详细描述：	设备端传输Y16数据进行回调
函数：	int SGP_GetY16(SGP_HANDLE handle, SGP_Y16CALLBACK callback, void *pUser);
参数：	handle [in] 传入设备对象 callback [in] 回调函数地址 pUser [in] 回调函数入参
返回值：	成功返回SGP_OK，失败返回错误码
备注：	1、要先打开视频流 2、单台机芯设备建议仅拉取一路y16数据，否则会影响性能、帧率等
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 2，以QT界面库为例

```
 **/
 static void GetY16Data(short
*y16,int length, void *pUser)
 {
   MainWindow *pDlg =
(MainWindow *)pUser;
   //TODO......
 }
 void MainWindow::Init()
 {
    SGP_GetY16(handle,
GetY16Data, this);
    //TODO......
 }
```

回调函数描述		
函数名称	typedef void(*SGP_Y16CALLBACK)(short *y16, int length, void *pUser);	
功能描述	传输Y16数据的回调函数	
参数说明	y16	输出参数，Y16和参数行数据
参数说	length	输出参数，Y16和参数

明		行数据的字节数
参数说明	pUser	输出参数
返回值	无	

停止获取Y16 SGP_StopY16

选项：	说明
描述：	停止获取Y16数据
详细描述：	设备端停止传输Y16数据
函数：	int SGP_StopY16(SGP_HANDLE handle);
参数：	handle [in] 传入设备对象
返回值：	无
备注：	

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	2,以QT界面库为例
	**/
	
	void MainWindow::Init()
	{
	SGP_StopY16(handle);
	//TODO.....
	}

Y16输出温度矩阵SGP_GetTempMatrixEx

选项：	说明
描述：	Y16输出温度矩阵
详细描述：	设备端传输Y16数据，由SDK调用测温库进行测温
函数：	int SGP_GetTempMatrixEx(SGP_HANDLE handle, float*dst, short *src, int w, int h);
参数：	handle [in] 传入设备对象 dst [out] 输出温度矩阵 src [in] 输入需要的Y16数据 w [in] 输入src的宽 h [in] 输入src的高
返回值：	成功返回SGP_OK,失败返回错误码

备注：	温度矩阵值为short类型，2个字节长度。

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle　设备对象。
	**/
	void Init()
	{
	//通过SGP_GetY16回调的Y16传入接口SGP_GetTempMatrixEx
	获取温度矩阵。
	ret = SGP_GetTempMatrixEx(handle,dst,src,width,height);
	if (ret == SGP_OK)
	{
	//成功，TODO......
	}
	else
	{
	//失败，TODO......
	}
	}

获取校温信息 SGP_GetMeasureTempInfo

选项：	说明
描述：	获取校温信息
详细描述：	
函数：	int SGP_GetMeasureTempInfo(SGP_HANDLE handle, SGP_MEASURE_TEMP_INFO &output);
参数：	handle [in] 传入设备对象 output [out] 输出校温信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
        // 获取校温信息
        SGP_MEASURE_TEMP_INFO output;
        ret =
SGP_GetMeasureTempInfo(handle,output);
        if (ret == SGP_OK )
        {
            //成功，TODO......
        }
        else
        {
            //失败，TODO......
        }
 }
```

拍照接口

接口描述	功能描述	功能详细描述
SGP_GetScreenCapture	获取屏幕截图	
SGP_GetScreenCaptureCache	获取屏幕截图(非文件)	
SGP_GetHeatMap	获取热图	
SGP_GetHeatMapCache	获取热图(非文件)	
SGP_GetFirHeatMap	获取高压热图	
SGP_GetFirHeatMapCache	获取高压热图(非文件)	

获取屏幕截图SGP_GetScreenCapture

选项：	说明
描述：	获取屏幕截图
详细描述：	图片不带温度
函数：	int SGP_GetScreenCapture(SGP_HANDLE handle, SGP_IMAGE_TYPE type, const char *input);
参数：	handle [in] 传入设备对象 type [in] 图片类型，1可见光，2红外图片 input [in] 保存文件路径+文件名+.jpg
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示	/** 示例中部分类、变量、函数的解释：

例：

```
 1，handle  设备对象。
 **/
 void Init()
 {
    SGP_IMAGE_TYPE type =
SGP_IR_IMAGE;
    char path[] =
"./screenpic.jpg";
    int ret =
SGP_GetScreenCapture(handle,type
,path);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取屏幕截图(非文件)SGP_GetScreenCaptureCache

选项：	说明
描述：	获取屏幕截图(非文件)
详细描述：	图片不带温度
函数：	int SGP_GetScreenCaptureCache(SGP_HANDLE handle, SGP_IMAGE_TYPE type, char *input, int input_length, int *output_length);
参数：	handle [in] 传入设备对象 type [in] 图片类型，1可见光，2红外图片 input [in] 外部分配，获取图片二进制数据 input_length [in] input大小 output_length [out] 图片二进制数据值大小
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init()

```
 {
     int input_length = 1024*1024*10;
     int output_length = 0;
     SGP_IMAGE_TYPE type = SGP_IR_IMAGE;

     char *input= (char *)calloc(input_length, sizeof(char));
     if(input!=NULL)
     {
        int
ret=SGP_GetScreenCaptureCache(handle,type,input,input_length,
&output_length);
        if (ret == SGP_OK )
        {
            //成功，TODO......
        }
        else
        {
            //失败，TODO......
        }
      }
     free(input);
     input=NULL;
 }
```

获取热图SGP_GetHeatMap

选项：	说明
描述：	获取热图
详细描述：	热图文件格式为国网格式
函数：	int SGP_GetHeatMap(SGP_HANDLE handle, const char *input);
参数：	handle [in] 传入设备对象 input [in] 保存文件路径+文件名+.jpg
返回值：	成功返回SGP_OK,失败返回错误码
备注：	获取的红外热图满足DLT 664-2016带电红外设备诊断应用规范对jpeg格式要求， 格式参考表1
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { const char path[] = "./screenpic.jpg";

```
    int ret = SGP_GetHeatMap
(handle,path);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

表1

获取热图(非文件)SGP_GetHeatMapCache

选项：	说明
描述：	获取热图(非文件)
详细描述：	热图文件格式为国网格式
函数：	int SGP_GetHeatMapCache(SGP_HANDLE handle, char *input, int input_length, int *output_length);
参数：	handle [in] 传入设备对象 input [in] 外部分配，获取图片二进制数据 input_length [in] input大小 output_length [out] 图片二进制数据大小
返回值：	成功返回SGP_OK,失败返回错误码
备注：	获取的红外热图满足DLT 664-2016带电红外设备诊断应用规范对jpeg格式要求，格式参考表1

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	**/
	void Init()
	{
	int input_length = 1024*1024*10;
	int output_length = 0;

```
    char *input= (char *)calloc(input_length, sizeof(char));
    if(input!=NULL)
    {
       int ret =
SGP_GetHeatMapCache(handle,input,input_length,&output_length);
       if (ret == SGP_OK )
       {
           //成功，TODO......
       }
       else
       {
           //失败，TODO......
       }
     }
    free(input);
    input=NULL;
}
```

获取高压热图SGP_GetFirHeatMap

选项：	说明
描述：	获取高压热图
详细描述：	
函数：	int SGP_GetFirHeatMap(SGP_HANDLE handle, const char *input);
参数：	handle [in] 传入设备对象 input [in] 保存文件路径+文件名+.fir
返回值：	成功返回SGP_OK,失败返回错误码
备注：	高压红外热图满足Q/GDW 12164-2021 变电站远程智能巡视系统技术规范中附录A 文件格式定义 A.1 红外图谱文件，格式参考表1
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init()

```
 {
     const char path[] =
"./screenpic.fir";
    int ret =
SGP_GetFirHeatMap(handle,path);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

表1

获取高压热图(非文件)SGP_GetFirHeatMapCache

选项：	说明
描述：	获取高压热图(非文件)
详细描述：	热图文件格式为高压国网格式
函数：	int SGP_GetFirHeatMapCache (SGP_HANDLE handle, char *input, int input_length, int *output_length);
参数：	handle [in] 传入设备对象 input [in] 外部分配，获取图片二进制数据 input_length [in] input大小 output_length [out] 图片二进制数据大小
返回值：	成功返回SGP_OK,失败返回错误码
备	高压红外热图满足Q/GDW 12164-2021 变电站远程智能巡视系统技术规范中附录A 文件格式定义 A.1 红外图

注：	谱文件，格式参考表1

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle　设备对象。
	**/
	void Init()
	{
	int input_length = 1024*1024*10;
	int output_length = 0;
	
	char *input= (char *)calloc(input_length, sizeof(char));
	if(input!=NULL)
	{
	int ret = SGP_GetFirHeatMapCache (handle,input,input_length,&output_length);
	if (ret == SGP_OK)
	{
	//成功，TODO......
	}
	else
	{
	//失败，TODO......
	}
	}
	free(input);
	input=NULL;
	}

成像接口

接口描述	功能描述	功能详细描述
SGP_OpenIrVideo	开启红外视频	
SGP_OpenVlVideo	开启可见光视频	
SGP_CloseIrVideo	关闭红外视频	
SGP_CloseVlVideo	关闭可见光视频	
SGP_GetIrImageInfo	获取成像参数	
SGP_SetIrImageInfo	设置成像参数	

开启红外视频SGP_OpenIrVideo

选项：	说明
描述：	开启红外视频
详细描述：	
函数：	int SGP_OpenIrVideo(SGP_HANDLE handle, SGP_RTSPCALLBACK callback, void *pUser);
参数：	handle [in] 传入设备对象 callback [in] 注册图像回调函数（RGB24数据） pUser [in] void* 可传入窗口句柄指针
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。

```
 **/
 /**
 示例中部分类、变量、函数的解释：
 1，handle   设备对象。
 2,以QT界面库为例
 **/
 static void
GetIrRtsp(unsigned char
*outdata, int w, int h, void
*pUser)
 {
   MainWindow *pDlg =
(MainWindow *)pUser;
   //TODO......
 }
 void MainWindow::Init()
 {
    SGP_OpenIrVideo(handle,
GetIrRtsp, this);
    //TODO......
 }
```

回调函数描述	

函数名称	typedef void(*SGP_RTSPCALLBACK)(unsigned char *outdata, int w, int h, void *pUser);	
功能描述	图像数据回调函数	
参数说明	outdata	输出参数 图像数据
	w	输出参数 图像宽度
	h	输出参数 图像高度
	pUser	输出参数
返回值	无	

开启可见光视频SGP_OpenVlVideo

选项：	说明
描述：	开启可见光视频
详细描述：	
函数：	int SGP_OpenVlVideo(SGP_HANDLE handle, SGP_RTSPCALLBACK callback, void *pUser);
参数：	handle [in] 传入设备对象 callback [in] 注册图像回调函数（RGB24数据） pUser [in] void* 可传入窗口句柄指针
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。

```
 **/
 /**
 示例中部分类、变量、函数的解释：
 1，handle   设备对象。
 2,以QT界面库为例
 **/
 static void
GetVIRtsp(unsigned char
*outdata, int w, int h, void
*ptr)
 {
   MainWindow *pDlg =
(MainWindow *)ptr;
   //TODO......
 }
 void MainWindow::Init()
 {
    SGP_OpenVlVideo(handle,
GetVIRtsp, this);
    //TODO......
 }
```

关闭红外视频SGP_CloseIrVideo

选项：	说明
描述：	关闭红外视频
详细描述：	
函数：	void SGP_CloseIrVideo(SGP_HANDLE handle);
参数：	handle [in] 传入设备对象
返回值：	无
备注：	退出登录会自动关闭视频流

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	**/
	void Init()
	{
	SGP_CloseIrVideo(handle);
	//TODO.....
	}

关闭可见光视频SGP_CloseVlVideo

选项：	说明
描述：	关闭可见光视频
详细描述：	
函数：	void SGP_CloseVlVideo (SGP_HANDLE handle);
参数：	handle [in] 传入设备对象
返回值：	无
备注：	退出登录会自动关闭视频流

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	**/
	void Init()
	{
	SGP_CloseVlVideo (handle);
	//TODO.....
	}

获取成像参数 SGP_GetIrImageInfo

选项：	说明
描述：	获取成像参数
详细描述：	
函数：	int SGP_GetIrImageInfo(SGP_HANDLE handle, SGP_IR_IMAGE_INFO * output);
参数：	handle [in] 传入设备对象 output [out] 传出降噪参数信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ /** 示例中部分类、变量、函数的解释： 1，handle 设备对象。

```
 2,以QT界面库为例
 **/
  void MainWindow::Init()
 {
    SGP_IR_IMAGE_INFO output;
    SGP_GetIrImageInfo(handle,
&output);
    //TODO......
 }
```

设置成像参数 SGP_SetIrImageInfo

选项：	说明
描述：	设置成像参数
详细描述：	
函数：	int SGP_SetIrImageInfo(SGP_HANDLE handle, SGP_IR_IMAGE_INFO_ENUM input, int value);
参数：	handle [in] 传入设备对象 input [in] 传入成像参数枚举类型 value [in] 传入参数值
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。

```
 **/
 /**
 示例中部分类、变量、函数的解释：
 1，handle   设备对象。
 2,以QT界面库为例
 **/
  void MainWindow::Init()
 {
    SGP_SetIrImageInfo(handle,
SGP_3D_FLAG, 4);
    //TODO......
 }
```

录像接口

接口描述	功能描述	功能详细描述
SGP_Record	录制视频	控制设备端录像，文件录制在红外设备上
SGP_StartRecord	开始录制	文件录制在本地电脑
SGP_StopRecord	停止录制	文件录制在本地电脑

录制视频SGP_Record

选项：	说明
描述：	录制视频
详细描述：	控制设备端录像,录制的视频文件存放在红外设备中
函数：	int SGP_Record(SGP_HANDLE handle, int subtype, int record_stream);
参数：	handle [in] 传入设备对象 subtype [in] 1:开始录制 2：停止录制 record_stream [in] 1:单光可见光； 2:单光红外； 3:双光，同时录制
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示	/** 示例中部分类、变量、函数的解释：

例：

```
1，handle  设备对象。
**/
void Init()
{
    int record = 1;
    int record_stream = 2;
    int ret =
SGP_Record(handle,subtype,record_stream);
    if (ret == SGP_OK )
    {
        //成功，TODO......
    }
    else
    {
        //失败，TODO......
    }
}
```

开始录制SGP_StartRecord

选项：	说明
描述：	开始录制
详细描述：	录制文件存放在本地
函数：	int SGP_StartRecord(SGP_HANDLE handle, SGP_VIDEO_TYPE type, const char *input, SGP_RECORDCALLBACK callback, void *pUser);
参数：	handle [in] 传入设备对象 type [in] 录像类型,1可见光录像，2红外录像 input [in] 保存文件路径+文件名+.mp4 callback [in] 录像状态回调函数，例如自动停止录像 pUser [in] 回调函数入参
返回值：	成功返回SGP_OK,失败返回错误码
备	与SGP_StopRecord成对使用，如果是红外录像，需要先调用接口

注：	SGP_OpenIrVideo打开红外视频，才能录像，如果是可见光本地录像，需要先调用接口SGP_OpenVlVideo打开可见光视频，才能进行可见光本地录像。

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle　设备对象。
	**/
	static void RecordCall(int state, void *pUser)
	{
	MainWindow *pDlg = (MainWindow *)pUser;
	//TODO......
	}
	static void GetIrRtsp(unsigned char *outdata, int w, int h, void *pUser)
	{
	MainWindow *pDlg = (MainWindow *)pUser;
	//TODO......
	}
	
	void Init()
	{
	SGP_VIDEO_TYPE type = SGP_IR_VIDEO;
	const char path[] = "./record.mp4";
	int ret = SGP_OpenIrVideo(handle, GetIrRtsp, this);
	if (ret == SGP_OK)
	{
	Sleep(5000);//打开视频后，间隔5s后开始录制视频
	int ret = SGP_StartRecord(handle,type,path,RecordCall,this);
	if (ret == SGP_OK)
	{

```
            //成功，TODO......
        }
        else
        {
            //失败，TODO......
        }
        SGP_StopRecord(handle,type);
    }
    SGP_CloseIrVideo(handle);
}
```

回调函数描述		
函数名称	typedef void(*SGP_RECORDCALLBACK)(int state, void *pUser);	
功能描述	本地录像回调函数	
参数说明	state	输出参数,1：开始录制 2：录制中 3：停止录制
	pUser	输出参数
返回值	无	

停止录制SGP_StopRecord

选项：	说明
描述：	停止录制
详细描述：	停止本地录制
函数：	void SGP_StopRecord(SGP_HANDLE handle, SGP_VIDEO_TYPE type);
参数：	handle [in] 传入设备对象 type [in] 录像类型,1可见光录像，2红外录像
返回值：	成功返回SGP_OK,失败返回错误码
备注：	与SGP_StartRecord成对使用

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	**/
	static void RecordCall(int state, void *pUser)
	{
	MainWindow *pDlg = (MainWindow *)pUser;
	//TODO......

```
 }
 static void GetIrRtsp(unsigned char *outdata, int
w, int h, void *pUser)
 {
   MainWindow *pDlg = (MainWindow *)pUser;
   //TODO......
 }
 void Init()
 {
     SGP_VIDEO_TYPE type = SGP_IR_VIDEO;
     const char path[] = "./record.mp4";
     int ret = SGP_OpenIrVideo(handle, GetIrRtsp,
this);
     if (ret == SGP_OK )
     {
        sleep(5000);//打开视频后，间隔5s后开始录制视频
        int ret =
SGP_StartRecord(handle,type,path,RecordCall,this);
        if (ret == SGP_OK )
        {
            //成功，TODO......
        }
        else
        {
            //失败，TODO......
        }
        SGP_StopRecord(handle,type);
     }
    SGP_CloseIrVideo(handle);
 }
```

操控设备接口

接口描述	功能描述	功能详细描述
SGP_SynchroTime	同步系统时间	
SGP_RebootSystem	系统重启	
SGP_ClearData	清理数据	
SGP_DoShutter	快门操作	
SGP_SetThermometryFlag	设置全局测温开关	
SGP_GetThermometryParam	获取全局测温参数	
SGP_SetThermometryParam	设置全局测温参数	
SGP_SetColorBar	设置色带号	
SGP_SetColorBarShow	设置色带显示	
SGP_SetTempShowMode	设置温度显示类型	
SGP_SetFocus	调焦	
SGP_GetMotorPosition	获取电机位置	
SGP_SetRange	切换测温范围	

SGP_SetStringShow	设置字符串叠加	
SGP_GetThermometryRule	获取分析对象	
SGP_AddThermometryRule	添加分析对象	
SGP_UpdateThermometryRule	更新分析对象	
SGP_DeleteThermometryRule	删除分析对象	
SGP_DeleteAllThermometryRule	删除全部分析对象	
SGP_SetThermometryRuleShowMode	设置分析对象温度显示类型	
SGP_GetIrImageEffectParam	获取红外图像效果参数	
SGP_SetIrImageEffectParam	设置红外图像效果参数	
SGP_GetVlImageEffectParam	获取可见光图像效果参数	
SGP_SetVlImageEffectParam	设置可见光图像效果参数	
SGP_GetImageFusion	获取图像融合	

SGP_SetImageFusion	设置图像融合	
SGP_GetNetInfo	获取网络信息	
SGP_SetNetInfo	设置网络信息	
SGP_GetPortInfo	获取端口信息	
SGP_SetPortInfo	设置端口信息	
SGP_GetShieldArea	获取屏蔽区域	
SGP_SetShieldArea	设置屏蔽区域	
SGP_GetColdHotTrace	获取全局温度告警	
SGP_SetColdHotTrace	设置全局温度告警	
SGP_GetTempAlarm	获取分析对象告警	
SGP_SetTempAlarm	设置分析对象告警	
SGP_GetVideoParam	获取视频参数	
SGP_SetVideoParam	设置视频参数	
SGP_GetVersionInfo	获取系统	

	版本信息	
SGP_GetNetException	获取网络异常	
SGP_SetNetException	设置网络异常	
SGP_GetAccessViolation	获取非法访问	
SGP_SetAccessViolation	设置非法访问	
SGP_GetEmilInfo	获取邮件信息	
SGP_SetEmilInfo	设置邮件信息	
SGP_GetFillLight	获取补光灯信息	
SGP_SetFillLight	设置补光灯信息	
SGP_GetInfraredMode	获取融合状态	
SGP_SetInfraredMode	设置融合状态	
SGP_GetSilentMode	获取蜂鸣器状态	
SGP_SetSilentMode	设置蜂鸣器状态	
SGP_GetRecordInfo	获取录制信息	

SGP_SetRecordInfo	设置录制信息	
SGP_SetElectronicMagnification	设置电子变倍	
SGP_GetAlarmInput	获取报警输入	
SGP_SetAlarmInput	设置报警输入	
SGP_FactoryReset	恢复出厂设置	
SGP_CommandSend	透传RS485数据查询	

同步系统时间SGP_SynchroTime

选项：	说明
描述：	同步系统时间
详细描述：	
函数：	int SGP_SynchroTime(SGP_HANDLE handle, const char *datetime);
参数：	handle [in] 传入设备对象 datetime [in] 同步时间,时间格式为"2020-05-21 12:22:33"
返回值：	成功返回SGP_OK,失败返回错误码
备注：	红外设备自身不带电池，此函数实现将电脑时间同步到红外设备上
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init()

```
 {
     const char* datetime =
"2022-03-18 12:22:33";
     int ret =
SGP_SynchroTime(handle,datetime);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

系统重启SGP_RebootSystem

选项：	说明
描述：	系统重启
详细描述：	
函数：	int SGP_RebootSystem(SGP_HANDLE handle);
参数：	handle [in] 传入设备对象
返回值：	成功返回SGP_OK,失败返回错误码
备注：	

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	**/
	void Init()
	{
	int ret = SGP_RebootSystem(handle);
	if (ret == SGP_OK)
	{
	//成功，TODO.....
	}
	else

```
        {
            //失败，TODO......
        }
}
```

清理数据SGP_ClearData

选项：	说明
描述：	清理数据
详细描述：	用于清理红外设备缓存数据
函数：	int SGP_ClearData(SGP_HANDLE handle);
参数：	handle [in] 传入设备对象
返回值：	成功返回SGP_OK,失败返回错误码
备注：	

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	**/
	void Init()
	{
	int ret = SGP_ClearData(handle);
	if (ret == SGP_OK)
	{
	//成功，TODO.....
	}
	else
	{
	//失败，TODO.....
	}

```
}
```

快门操作SGP_DoShutter

选项：	说明
描述：	快门操作
详细描述：	
函数：	int SGP_DoShutter(SGP_HANDLE handle, SGP_SHUTTER_ENUM type);
参数：	handle [in] 传入设备对象 type [in] 快门操作类型
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_SHUTTER_ENUM type = SGP_SHUTTER;

```
    int ret =
SGP_DoShutter(handle,type);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置全局测温开关SGP_SetThermometryFlag

选项：	说明
描述：	设置全局测温开关
详细描述：	
函数：	int SGP_SetThermometryFlag(SGP_HANDLE handle, int input);
参数：	handle [in] 传入设备对象 input [in] 0关闭 1开启
返回值：	成功返回SGP_OK,失败返回错误码
备注：	

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	**/
	void Init()
	{
	int flag = 0;

```
    int ret =
SGP_SetThermometryFlag(handle,flag);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取全局测温参数
SGP_GetThermometryParam

选项：	说明
描述：	获取全局测温参数
详细描述：	
函数：	int SGP_GetThermometryParam(SGP_HANDLE handle, SGP_THERMOMETRY_PARAM *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
     SGP_THERMOMETRY_PARAM info;
     memset(&info, 0x00,
sizeof(SGP_THERMOMETRY_PARAM));
    int ret =
SGP_GetThermometryParam(handle,&info);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置全局测温参数
SGP_SetThermometryParam

选项：	说明
描述：	设置全局测温参数
详细描述：	
函数：	int SGP_SetThermometryParam(SGP_HANDLE handle, SGP_THERMOMETRY_PARAM input);
参数：	handle [in] 传入设备对象 input [in] 输入信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	使用前先调用SGP_GetThermometryParam获取全局测温参数，然后再修改测温参数， 结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/

```
 void Init()
 {
     //先获取全局测温参数，再设置参数。
     SGP_THERMOMETRY_PARAM info;
     memset(&info, 0x00,
sizeof(SGP_THERMOMETRY_PARAM));
     int ret =
SGP_GetThermometryParam(handle,&info);
     if (ret == SGP_OK )
     {
           parm.dist = 5; //修改测温距离
为5米
           ret =
SGP_SetThermometryParam(handle,info);
           if (ret == SGP_OK )
           {
               //成功，TODO......
           }
           else
           {
               //失败，TODO......
           }

      }
 }
```

设置色带号SGP_SetColorBar

选项：	说明
描述：	设置色带号
详细描述：	
函数：	int SGP_SetColorBar(SGP_HANDLE handle, int input);
参数：	handle [in] 传入设备对象 input [in] 色带号1~26
返回值：	成功返回SGP_OK,失败返回错误码
备注：	

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	**/
	void Init()
	{
	int colorbar = 2;

```
   int ret =
SGP_SetColorBar(handle,colorbar);
   if (ret == SGP_OK )
   {
    //success, TODO......
   }
   else
   {
    //fail, TODO ......
   }
}
```

设置色带显示SGP_SetColorBarShow

选项：	说明
描述：	设置色带显示
详细描述：	
函数：	int SGP_SetColorBarShow(SGP_HANDLE handle, int input);
参数：	handle [in] 传入设备对象 input [in] 0关闭 1开启
返回值：	成功返回SGP_OK,失败返回错误码
备注：	

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	**/
	void Init()
	{
	int showcolorbar = 1;

```
   int ret =
SGP_SetColorBarShow(handle,showcolorbar);
   if (ret == SGP_OK )
   {
    //success, TODO......
   }
   else
   {
    //fail, TODO ......
   }
}
```

设置温度显示类型SGP_SetTempShowMode

选项：	说明
描述：	设置温度显示类型
详细描述：	
函数：	int SGP_SetTempShowMode(SGP_HANDLE handle, int input);
参数：	handle [in]　传入设备对象 input [in]　温度显示方式：1 最高温 2 最低温 3 平均温 4 最高温 + 最低温 5 最高温 + 平均温 6 平均温 + 最低温 7 最高温 + 最低温 + 平均温 8不显示
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle　设备对象。

```
 **/
 void Init()
 {
     int showtype = 1;
    int ret =
SGP_SetTempShowMode(handle,showtype);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

调焦SGP_SetFocus

选项：	说明
描述：	调焦
详细描述：	
函数：	int SGP_SetFocus(SGP_HANDLE handle, SGP_FOCUS_TYPE type, int value);
参数：	handle [in] 传入设备对象 type [in] 操作类型 value [in] 电机位置值0~750，当type传入SGP_FOCUS_PLACE有效
返回值：	成功返回SGP_OK,失败返回错误码
备注：	电机位置值范围0~750
使用示	/** 示例中部分类、变量、函数的解释：

例：

```
 1，handle  设备对象。
 **/
 void Init()
 {
     SGP_FOCUS_TYPE type =
SGP_FOCUS_AUTO;
     int value =0;
    int ret =
SGP_SetFocus(handle,type,value);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取电机位置SGP_GetMotorPosition

选项：	说明
描述：	获取电机位置
详细描述：	
函数：	int SGP_GetMotorPosition(SGP_HANDLE handle, int *output);
参数：	handle [in] 传入设备对象 output [out] 电机位置
返回值：	成功返回SGP_OK,失败返回错误码
备注：	

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	**/
	void Init()
	{
	int value =0;

```
    int ret =
SGP_GetMotorPosition(handle,&value);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

切换测温范围SGP_SetRange

选项：	说明
描述：	切换测温范围
详细描述：	
函数：	int SGP_SetRange(SGP_HANDLE handle, int input);
参数：	handle [in] 传入设备对象 input [in] 0~2，部分设备只有1个档位，目前最多有3个档位
返回值：	成功返回SGP_OK,失败返回错误码
备注：	调用SGP_GetGeneralInfo获取设备支持的档位，例如对IPT640M，0表示-20℃~150℃,1表示100℃~350℃，2表示100℃~550℃（如果设备支持则包含2）
使用示	/** 示例中部分类、变量、函数的解释：

例：

```
 1，handle  设备对象。
 **/
 void Init()
 {
     SGP_GENERAL_INFO info;
     memset(&info, 0x00,
sizeof(SGP_GENERAL_INFO));
     int ret =
SGP_GetGeneralInfo(handle,&info);
     if (ret ==SGP_OK）
     {
         int range =
info.range_num;
        ret =
SGP_SetRange(handle,range -1);
         if (ret == SGP_OK )
         {
              //成功，TODO......
         }
         else
         {
              //失败，TODO......
         }
 }
```

设置字符串叠加SGP_SetStringShow

选项：	说明
描述：	设置字符串叠加
详细描述：	
函数：	int SGP_SetStringShow(SGP_HANDLE handle, int type, const char *input);
参数：	handle [in] 传入设备对象 type [in] 是否使用字符叠加 其他机型 1:关闭; 2，4，5:右下; 3:右上 IPT640M 1:关闭; 2:左上; 3:右上; 4:左下; 5:右下 IPM630 1:关闭; 5:右下 IPT430M 1:关闭; 5:右下 input [in] 需要显示的字符串

返回值：	成功返回SGP_OK,失败返回错误码
备注：	

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle　设备对象。
	**/
	void Init()
	{
	int type = 3;
	const char* dateshow= "1号设备终端";
	int ret = SGP_SetStringShow(handle,type,dateshow);
	if (ret == SGP_OK)
	{
	//成功，TODO.....
	}
	else
	{
	//失败，TODO.....
	}
	}

获取分析对象SGP_GetThermometryRule

选项：	说明
描述：	获取分析对象
详细描述：	
函数：	int SGP_GetThermometryRule(SGP_HANDLE handle, SGP_RULE_ARRAY *output);
参数：	handle [in] 传入设备对象 output [out] 全部分析对象
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_RULE_ARRAY array;

```
     memset(&array, 0x00,
sizeof(SGP_RULE_ARRAY));
    int ret =
SGP_GetThermometryRule(handle,&array);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }

 }
```

添加分析对象SGP_AddThermometryRule

选项：	说明
描述：	添加分析对象
详细描述：	
函数：	int SGP_AddThermometryRule(SGP_HANDLE handle, SGP_RULE input);
参数：	handle [in] 传入设备对象 input [in] 分析对象类型
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_RULE rulePoint;

```
     memset(&rule, 0x00,
sizeof(SGP_RULE));
     rulePoint.alarm_condition = 1;
     rulePoint.alarm_flag = 1;
     rulePoint.alarm_time = 30;
     rulePoint.alarm_type = 1;
     rulePoint.avg_temp = 30;
     rulePoint.flag = 1;
     rulePoint.high_temp = 35;
     rulePoint.low_temp = 28;
     rulePoint.points_num = 1; //点个数是1
     rulePoint.points[0].x = 200;
     rulePoint.points[0].y = 200;
     strcpy(rulePoint.rule_name, "点1");
     rulePoint.show_location = 1;
     rulePoint.temp_mod = 1;
     rulePoint.type = 1;  //类型是1
     rulePoint.atmo_trans = 0.9;
     rulePoint.dist = 2;
     rulePoint.emiss = 0.8;
     rulePoint.emiss_mode = 1;
     rulePoint.humi = 80;
     rulePoint.opti_trans = 1;
     rulePoint.ref_temp = 25;

    int ret =
SGP_AddThermometryRule(handle,rulePoint);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

更新分析对象SGP_UpdateThermometryRule

选项：	说明
描述：	更新分析对象
详细描述：	
函数：	int SGP_UpdateThermometryRule(SGP_HANDLE handle, SGP_RULE input);
参数：	handle [in] 传入设备对象 input [in] 分析对象
返回值：	成功返回SGP_OK,失败返回错误码
备注：	先调用SGP_GetThermometryRule函数获取,再更新，结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_RULE_ARRAY array;

```
     memset(&array, 0x00,
sizeof(SGP_RULE_ARRAY));
    int ret =
SGP_GetThermometryRule(handle,&array);
     if (ret == SGP_OK )
     {
          if(array.rule_num>0)
          {
              SGP_RULE rule;
              memset(&rule, 0,
sizeof(SGP_RULE));

memcpy(&rule,&array.rule[0],sizeof(SGP_RULE));//
取第一组分析对象值

              rule.alarm_condition = 1;
              rule.alarm_flag =1;
              rule.alarm_time = 10;
             int ret =
SGP_UpdateThermometryRule(handle,rule);
              if (ret == SGP_OK )
              {
                  //成功，TODO......
              }
              else
              {
                  //失败，TODO......
              }
          }
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

删除分析对象SGP_DeleteThermometryRule

选项：	说明
描述：	删除分析对象
详细描述：	
函数：	int SGP_DeleteThermometryRule(SGP_HANDLE handle, int input);
参数：	handle [in] 传入设备对象 input [in] 分析对象id
返回值：	成功返回SGP_OK,失败返回错误码
备注：	

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	**/
	void Init()
	{
	SGP_RULE_ARRAY array;
	memset(&array, 0x00, sizeof(SGP_RULE_ARRAY));
	int ret =
	SGP_GetThermometryRule(handle,&array);

```
     if (ret == SGP_OK )
     {
          if(array.rule_num>0)
          {
             int ret =
SGP_DeleteThermometryRule(handle,array.rule[0].id);
              if (ret == SGP_OK )
              {
                  //成功，TODO......
              }
              else
              {
                  //失败，TODO......
              }
          }
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

删除全部分析对象
SGP_DeleteAllThermometryRule

选项：	说明
描述：	删除全部分析对象
详细描述：	
函数：	int SGP_DeleteAllThermometryRule(SGP_HANDLE handle);
参数：	handle [in] 传入设备对象
返回值：	成功返回SGP_OK,失败返回错误码
备注：	

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	**/
	void Init()
	{
	int ret =
	SGP_DeleteAllThermometryRule(handle);
	if (ret == SGP_OK)

```
        {
            //成功，TODO......
        }
        else
        {
            //失败，TODO......
        }
}
```

设置分析对象温度显示类型
SGP_SetThermometryRuleShowMode

选项：	说明
描述：	设置分析对象温度显示类型
详细描述：	
函数：	int SGP_SetThermometryRuleShowMode(SGP_HANDLE handle, int input);
参数：	handle [in] 传入设备对象 input [in] 对象温度显示:1最高温;2最低温;3平均温;4仅名称;5不显示
返回值：	成功返回SGP_OK,失败返回错误码
备注：	前提是有分析对象，设置分析对象温度显示类型才会生效。
使用示例：	/** 示例中部分类、变量、函数的解释: 1，handle 设备对象。 **/ void Init()

```
 {
     int showtype = 1;
    int ret =
SGP_SetThermometryRuleShowMode(handle,showtype);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取红外图像效果参数
SGP_GetIrImageEffectParam

选项：	说明
描述：	获取红外图像效果参数
详细描述：	
函数：	int SGP_GetIrImageEffectParam(SGP_HANDLE handle, SGP_IAMGE_EFFECT_PARAM_IR_CONFIG *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
     SGP_IAMGE_EFFECT_PARAM_IR_CONFIG info;
     memset(&info, 0x00, sizeof(info));
    int ret =
SGP_GetIrImageEffectParam(handle,&info);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置红外图像效果参数
SGP_SetIrImageEffectParam

选项：	说明
描述：	设置红外图像效果参数
详细描述：	
函数：	int SGP_SetIrImageEffectParam(SGP_HANDLE handle, SGP_IR_IMAGE_EFFECT_ENUM type, int value);
参数：	handle [in] 传入设备对象 type [in] 参数类型 value [in] 参数值
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示	/** 示例中部分类、变量、函数的解释：

例：

```
 1，handle  设备对象。
 **/
 void Init()
 {
     SGP_IR_IMAGE_EFFECT_ENUM type =
SGP_IR_ROTATE;
     int value = 1;
     int ret =
SGP_SetIrImageEffectParam(handle,type,value);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取可见光图像效果参数
SGP_GetVlImageEffectParam

选项：	说明
描述：	获取可见光图像效果参数
详细描述：	此函数适用于带可见光的双光设备
函数：	int SGP_GetVlImageEffectParam(SGP_HANDLE handle, SGP_IAMGE_EFFECT_PARAM_VL_CONFIG *output);
参数：	handle [in] 传入设备对象 output [out] 输出值
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
     SGP_IAMGE_EFFECT_PARAM_VL_CONFIG info;
     memset(&info, 0x00,
sizeof(SGP_IAMGE_EFFECT_PARAM_VL_CONFIG));
    int ret =
SGP_GetVlImageEffectParam(handle,&info);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置可见光图像效果参数
SGP_SetVlImageEffectParam

选项：	说明
描述：	设置可见光图像效果参数
详细描述：	此函数适用于带可见光的双光设备
函数：	int SGP_SetVlImageEffectParam(SGP_HANDLE handle, SGP_VL_IMAGE_EFFECT_ENUM type, int value);
参数：	handle [in] 传入设备对象 type [in] 参数类型 value [in] 参数值
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示	/** 示例中部分类、变量、函数的解释：

例：

```
 1，handle  设备对象。
 **/
 void Init()
 {
     SGP_VL_IMAGE_EFFECT_ENUM  type =
SGP_VL_BRIGHTNESS;
     int value = 50;
    int ret =
SGP_SetVlImageEffectParam(handle,type,value
);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取图像融合SGP_GetImageFusion

选项：	说明
描述：	获取图像融合
详细描述：	此函数适用于带可见光的双光设备
函数：	int SGP_GetImageFusion(SGP_HANDLE handle, SGP_IMAGE_FUSION *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_IMAGE_FUSION info;

```
     memset(&info, 0x00,
sizeof(SGP_IMAGE_FUSION));
     int ret =
SGP_GetImageFusion(handle,&info);
     if (ret == SGP_OK )
     {

          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置图像融合SGP_SetImageFusion

选项：	说明
描述：	设置图像融合
详细描述：	此函数适用于带可见光的双光设备
函数：	int SGP_SetImageFusion(SGP_HANDLE handle, SGP_IMAGE_FUSION input);
参数：	handle [in] 传入设备对象 input [in] 输入信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	先调用SGP_GetImageFusion函数获取，再设置，结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
     SGP_IMAGE_FUSION info;
     memset(&info, 0x00,
sizeof(SGP_IMAGE_FUSION));
     int ret =
SGP_GetImageFusion(handle,&info);
     if (ret == SGP_OK )
     {
          info.percent  = 50;
          ret =
SGP_SetImageFusion(handle,info);
          if (ret == SGP_OK )
          {
               //成功，TODO......
          }
          else
          {
               //失败，TODO......
          }

     }
     else
     {
          //失败，TODO......
     }
 }
```

获取网络信息SGP_GetNetInfo

选项：	说明
描述：	获取网络信息
详细描述：	
函数：	int SGP_GetNetInfo(SGP_HANDLE handle, SGP_NET_INFO *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在定义后先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_NET_INFO info;

```
     memset(&info, 0x00,
sizeof(SGP_NET_INFO));
     int ret =
SGP_GetNetInfo(handle,&info);
     if (ret == SGP_OK )
     {

          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置网络信息SGP_SetNetInfo

选项：	说明
描述：	设置网络信息
详细描述：	
函数：	int SGP_SetNetInfo(SGP_HANDLE handle, SGP_NET_INFO input);
参数：	handle [in] 传入设备对象 input [in] 输入信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	先调用SGP_GetNetInfo函数,再设置,结构体变量在定义后需要先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
     SGP_NET_INFO info;
     memset(&info, 0x00,
sizeof(SGP_NET_INFO));
     int ret =
SGP_GetNetInfo(handle,&info);
     if (ret == SGP_OK )
     {
          info.card = 1;
          ret =
SGP_SetNetInfo(handle,info);
          if (ret == SGP_OK )
          {
               //成功,
TODO......
          }
          else
          {
               //失败,
TODO......
          }

     }
     else
     {
          //失败，TODO......
     }
 }
```

获取端口信息SGP_GetPortInfo

选项：	说明
描述：	获取端口信息
详细描述：	
函数：	int SGP_GetPortInfo(SGP_HANDLE handle, SGP_PORT_INFO *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_PORT_INFO info;

```
     memset(&info, 0x00,
sizeof(SGP_PORT_INFO));
     int ret =
SGP_GetPortInfo(handle,&info);
     if (ret == SGP_OK )
     {

          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }

 }
```

设置端口信息SGP_SetPortInfo

选项：	说明
描述：	设置端口信息
详细描述：	
函数：	int SGP_SetPortInfo(SGP_HANDLE handle, SGP_PORT_INFO input);
参数：	handle [in] 传入设备对象 input [in] 输入信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	先调用SGP_GetPortInfo函数再设置，结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_PORT_INFO info;

```
     memset(&info, 0x00,
sizeof(SGP_PORT_INFO));
     int ret =
SGP_GetPortInfo(handle,&info);
     if (ret == SGP_OK )
     {
          info.max_connectios =
10;
          ret =
SGP_SetPortInfo(handle,info);
          if (ret == SGP_OK )
          {
               //成功,
TODO......
          }
          else
          {
               //失败,
TODO......
          }
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取屏蔽区域SGP_GetShieldArea

选项：	说明
描述：	获取屏蔽区域
详细描述：	
函数：	int SGP_GetShieldArea(SGP_HANDLE handle, SGP_SHIELD_AREA_INFO *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_SHIELD_AREA_INFO info;

```
     memset(&info, 0x00,
sizeof(SGP_SHIELD_AREA_INFO));
     int ret =
SGP_GetShieldArea(handle,&info);
     if (ret == SGP_OK )
     {

          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }

 }
```

设置屏蔽区域SGP_SetShieldArea

选项：	说明
描述：	设置屏蔽区域
详细描述：	
函数：	int SGP_SetShieldArea(SGP_HANDLE handle, SGP_SHIELD_AREA_INFO input);
参数：	handle [in] 传入设备对象 input [in] 输入信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	先调用SGP_GetShieldArea函数,再设置，结构体变量在使用前先初始化

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	**/
	void Init()
	{
	SGP_SHIELD_AREA_INFO info;
	memset(&info, 0x00, sizeof(SGP_SHIELD_AREA_INFO));
	int ret = SGP_GetShieldArea(handle,&info);
	if (ret == SGP_OK)
	{

```
            info.rect_num = 2;

           info.rect[0].x = 100;
           info.rect[0].y = 100;
           info.rect[0].w = 50;
           info.rect[0].h = 50;
           info.rect[1].x = 200;
           info.rect[1].y = 200;
           info.rect[1].w = 50;
           info.rect[1].h = 50;

            ret =
SGP_SetShieldArea(handle,info);
           if (ret == SGP_OK )
           {
                //成功，TODO......
           }
           else
           {
                //失败，TODO......
           }
     }
     else
     {
           //失败，TODO......
     }
 }
```

获取全局温度告警SGP_GetColdHotTrace

选项：	说明
描述：	获取全局温度告警
详细描述：	
函数：	int SGP_GetColdHotTrace(SGP_HANDLE handle, SGP_COLD_HOT_TRACE_INFO *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_COLD_HOT_TRACE_INFO info;

```
     memset(&info, 0x00,
sizeof(SGP_COLD_HOT_TRACE_INFO));
     int ret =
SGP_GetColdHotTrace(handle,&info);
     if (ret == SGP_OK )
     {

          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置全局温度告警SGP_SetColdHotTrace

选项：	说明
描述：	设置全局温度告警
详细描述：	
函数：	int SGP_SetColdHotTrace(SGP_HANDLE handle, SGP_COLD_HOT_TRACE_INFO input);
参数：	handle [in] 传入设备对象 input [in] 输入信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	先调用SGP_GetColdHotTrace函数,再设置，结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
     SGP_COLD_HOT_TRACE_INFO info;
     memset(&info, 0x00,
sizeof(SGP_COLD_HOT_TRACE_INFO));
     int ret =
SGP_GetColdHotTrace(handle,&info);
     if (ret == SGP_OK )
     {
          info.alarm_out_delay =
10;
          ret =
SGP_SetColdHotTrace(handle,info);
          if (ret == SGP_OK )
          {
               //成功，TODO......
          }
          else
          {
               //失败，TODO......
          }
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取分析对象告警SGP_GetTempAlarm

选项：	说明
描述：	获取分析对象告警
详细描述：	
函数：	int SGP_GetTempAlarm(SGP_HANDLE handle, SGP_TEMP_ALARM_INFO *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_TEMP_ALARM_INFO info;

```
     memset(&info, 0x00,
sizeof(SGP_TEMP_ALARM_INFO));
     int ret =
SGP_GetTempAlarm(handle,&info);
     if (ret == SGP_OK )
     {

          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }

 }
```

设置分析对象告警SGP_SetTempAlarm

选项：	说明
描述：	设置分析对象告警
详细描述：	
函数：	int SGP_SetTempAlarm(SGP_HANDLE handle, SGP_TEMP_ALARM_INFO input);
参数：	handle [in] 传入设备对象 input [in] 输入信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	先调用SGP_GetTempAlarm函数,再设置，结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
     SGP_TEMP_ALARM_INFO info;
     memset(&info, 0x00,
sizeof(SGP_TEMP_ALARM_INF));
     int ret =
SGP_GetTempAlarm(handle,&info);
     if (ret == SGP_OK )
     {
          info.audio_flag = 1;
          ret =
SGP_SetTempAlarm(handle,info);
          if (ret == SGP_OK )
          {
               //成功,
TODO......
          }
          else
          {
               //失败,
TODO......
          }
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取视频参数SGP_GetVideoParam

选项：	说明
描述：	获取视频参数
详细描述：	
函数：	int SGP_GetVideoParam(SGP_HANDLE handle, SGP_VIDEO_PARAM_ENUM type, SGP_VIDEO_PARAM *output);
参数：	handle [in] 传入设备对象 type [in] 视频类别 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。

```
 **/
 void Init()
 {
     SGP_VIDEO_PARAM_ENUM type =
SGP_IR;
     SGP_VIDEO_PARAM info;
     memset(&info, 0x00,
sizeof(SGP_VIDEO_PARAM));
     int ret =
SGP_GetVideoParam(handle,type,&info);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }

 }
```

设置视频参数SGP_SetVideoParam

选项：	说明
描述：	设置视频参数
详细描述：	
函数：	int SGP_SetVideoParam(SGP_HANDLE handle, SGP_VIDEO_PARAM_ENUM type, SGP_VIDEO_PARAM input);
参数：	handle [in] 传入设备对象 type [in] 视频类别 input [in] 参数值
返回值：	成功返回SGP_OK,失败返回错误码
备注：	调用SGP_GetVideoParam函数,再设置，结构体变量在使用前先初始化。设置帧率后必须把i帧间隔和码流同步设置，否则可能会设置失败。建议添加i帧间隔和码流设置
使用示	/** 示例中部分类、变量、函数的解释：

例：

```
 1，handle  设备对象。
 **/
 void Init()
 {
     SGP_GENERAL_INFO info;
     memset(&info, 0x00,
sizeof(SGP_GENERAL_INFO));
     int ret =
SGP_GetGeneralInfo(handle,&info);

     if (ret == SGP_OK )
     {
          SGP_VIDEO_PARAM_ENUM type =
SGP_IR;
          SGP_VIDEO_PARAM param;
          memset(&param, 0x00,
sizeof(SGP_VIDEO_PARAM));
          ret =
SGP_GetVideoParam(handle,type,&param);
          param.fps = 25;
          param.bit_size  =
info.ir_model_w *ir_model_h
*1.5*info.fps*8/18;
          param.gop_size  =
info.fps*2;
          ret =
SGP_SetVideoParam(handle,type,param);
          if (ret == SGP_OK )
          {
               //成功，TODO......
          }
          else
          {
               //失败，TODO......
          }
     }
```

```
    else
    {
        //失败，TODO......
    }
}
```

获取系统版本信息SGP_GetVersionInfo

选项：	说明
描述：	获取系统版本信息
详细描述：	
函数：	int SGP_GetVersionInfo(SGP_HANDLE handle, SGP_VERSION_INFO *output);
参数：	handle [in] 传入设备对象 output [out] 版本信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_VERSION_INFO info;

```
     memset(&info, 0x00,
sizeof(SGP_VERSION_INFO));
     int ret =
GetVersionInfo(handle,&info);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取网络异常SGP_GetNetException

选项：	说明
描述：	获取网络异常
详细描述：	
函数：	int SGP_GetNetException(SGP_HANDLE handle, SGP_NET_EXCEPTION_INFO *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_NET_EXCEPTION_INFO info;

```
     memset(&info, 0x00,
sizeof(SGP_NET_EXCEPTION_INFO));
     int ret =
SGP_GetNetException(handle,&info);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置网络异常SGP_SetNetException

选项：	说明
描述：	设置网络异常
详细描述：	
函数：	int SGP_SetNetException(SGP_HANDLE handle, SGP_NET_EXCEPTION_INFO input);
参数：	handle [in] 传入设备对象 input [in] 输入信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	先调用SGP_GetNetException函数,再设置,结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
     SGP_NET_EXCEPTION_INFO info;
     memset(&info, 0x00,
sizeof(SGP_NET_EXCEPTION_INFO));
     int ret =
SGP_GetNetException(handle,&info);
     if (ret == SGP_OK )
     {
          info.audio_flag = 1;
          ret =
SGP_SetNetException(handle,info);
          if (ret == SGP_OK )
          {
               //成功，TODO......
          }
          else
          {
               //失败，TODO......
          }
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取非法访问SGP_GetAccessViolation

选项：	说明
描述：	获取非法访问
详细描述：	
函数：	int SGP_GetAccessViolation(SGP_HANDLE handle, SGP_ACCESS_VIOLATION_INFO *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_ACCESS_VIOLATION_INFO info;

```
     memset(&info, 0x00,
sizeof(SGP_ACCESS_VIOLATION_INFO));
     int ret =
SGP_GetAccessViolation(handle,&info);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置非法访问SGP_SetAccessViolation

选项：	说明
描述：	设置非法访问
详细描述：	
函数：	int SGP_SetAccessViolation(SGP_HANDLE handle, SGP_ACCESS_VIOLATION_INFO input);
参数：	handle [in] 传入设备对象 input [in] 输入信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	先调用SGP_GetAccessViolation函数,再设置,结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
     SGP_ACCESS_VIOLATION_INFO info;
     memset(&info, 0x00,
sizeof(SGP_ACCESS_VIOLATION_INF));
     int ret =
GetAccessViolation(handle,&info);
     if (ret == SGP_OK )
     {
          info.audio_flag = 0;
          ret =
SGP_SetAccessViolation(handle,info);
          if (ret == SGP_OK )
          {
               //成功，TODO......
          }
          else
          {
               //失败，TODO......
          }
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取邮件信息SGP_GetEmilInfo

选项：	说明
描述：	获取邮件信息
详细描述：	
函数：	int SGP_GetEmilInfo(SGP_HANDLE handle, SGP_EMAIL_INFO *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_EMAIL_INFO info;

```
     memset(&info, 0x00,
sizeof(SGP_EMAIL_INFO));
     int ret =
SGP_GetEmilInfo(handle,&info);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置邮件信息SGP_SetEmilInfo

选项：	说明
描述：	设置邮件信息
详细描述：	
函数：	int SGP_SetEmilInfo(SGP_HANDLE handle, SGP_EMAIL_INFO input);
参数：	handle [in] 传入设备对象 input [in] 输入信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	先调用SGP_GetEmilInfo函数,再设置,结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
     SGP_EMAIL_INFO info;
     memset(&info, 0x00,
sizeof(SGP_EMAIL_INFO));
     int ret =
SGP_GetEmilInfo(handle,&info);
     if (ret == SGP_OK )
     {
          info.alarm = 0;
          ret =
SGP_SetEmilInfo(handle,info);
          if (ret == SGP_OK )
          {
               //成功，
TODO......
          }
          else
          {
               //失败，
TODO......
          }
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取补光灯信息SGP_GetFillLight

选项：	说明
描述：	获取补光灯信息
详细描述：	此函数适用于带可见光的双光设备
函数：	int SGP_GetFillLight(SGP_HANDLE handle, SGP_FILL_LIGHT_INFO *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_FILL_LIGHT_INFO info;

```
     memset(&info, 0x00,
sizeof(SGP_FILL_LIGHT_INFO));
     int ret =
SGP_GetFillLight(handle,&info);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置补光灯信息SGP_SetFillLight

选项：	说明
描述：	设置补光灯信息
详细描述：	此函数适用于带可见光的双光设备
函数：	int SGP_SetFillLight(SGP_HANDLE handle, SGP_FILL_LIGHT_INFO input);
参数：	handle [in] 传入设备对象 input [in] 输入信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	先调用SGP_GetFillLight函数,再设置,结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
     SGP_FILL_LIGHT_INFO info;
     memset(&info, 0x00,
sizeof(SGP_FILL_LIGHT_INFO));
     int ret =
SGP_GetFillLight(handle,&info);
     if (ret == SGP_OK )
     {
          info.brightness= 50;
          ret =
SGP_SetFillLight(handle,info);
          if (ret == SGP_OK )
          {
               //成功,
TODO......
          }
          else
          {
               //失败,
TODO......
          }
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取融合状态SGP_GetInfraredMode

选项：	说明
描述：	获取融合状态
详细描述：	此函数适用于带可见光的双光设备
函数：	int SGP_GetInfraredMode(SGP_HANDLE handle, int *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息，mode红外模式： 0:单光红外； 1:双光红外
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
     int mode = 0;
     int ret =
SGP_GetInfraredMode(handle,&mode);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置融合状态SGP_SetInfraredMode

选项：	说明
描述：	设置融合状态
详细描述：	此函数适用于带可见光的双光设备
函数：	int SGP_SetInfraredMode(SGP_HANDLE handle, int input);
参数：	handle [in] 传入设备对象 input [in] mode红外模式： 0:单光红外; 1:双光红外
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释: 1，handle 设备对象。 **/ void Init() {

```
     int mode = 1;
     int ret =
SGP_SetInfraredMode(handle,mode);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取蜂鸣器状态SGP_GetSilentMode

选项：	说明
描述：	获取蜂鸣器状态
详细描述：	
函数：	int SGP_GetSilentMode(SGP_HANDLE handle, int *output);
参数：	handle [in] 传入设备对象 output [out] 0:非静音; 1:静音
返回值：	成功返回SGP_OK,失败返回错误码
备注：	

使用示例：	/**
	示例中部分类、变量、函数的解释:
	1，handle 设备对象。
	**/
	void Init()
	{
	int silent= 0;

```
     int ret =
SGP_GetSilentMode(handle,&silent);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置蜂鸣器状态SGP_GetSilentMode

选项：	说明
描述：	设置蜂鸣器状态
详细描述：	
函数：	int SGP_SetSilentMode(SGP_HANDLE handle, int input);
参数：	handle [in] 传入设备对象 input [in] 0:非静音; 1:静音
返回值：	成功返回SGP_OK,失败返回错误码
备注：	

使用示例：	/**
	示例中部分类、变量、函数的解释:
	1，handle 设备对象。
	**/
	void Init()
	{
	int input= 0;

```
     int ret =
SGP_SetSilentMode(handle,input);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取录制信息SGP_GetRecordInfo

选项：	说明
描述：	获取录制信息
详细描述：	
函数：	int SGP_GetRecordInfo(SGP_HANDLE handle, SGP_RECORD_INFO *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_RECORD_INFO info;

```
     memset(&info, 0x00,
sizeof(SGP_RECORD_INFO));
     int ret =
SGP_GetRecordInfo(handle,&info);
     if (ret == SGP_OK )
     {

          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置录制信息SGP_SetRecordInfo

选项：	说明
描述：	设置录制信息
详细描述：	
函数：	int SGP_SetRecordInfo(SGP_HANDLE handle, SGP_RECORD_INFO input);
参数：	handle [in] 传入设备对象 input [in] 录制信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	先调用SGP_GetRecordInfo函数,再设置,结构体变量在使用前先初始化
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
     SGP_RECORD_INFO  info;
     memset(&info, 0x00,
sizeof(SGP_RECORD_INFO));
     int ret =
SGP_GetRecordInfo(handle,&info);
     if (ret == SGP_OK )
     {

info.record_interval  = 1;
          ret =
SGP_SetRecordInfo(handle,info);
          if (ret == SGP_OK )
          {
               //成功，TODO......
          }
          else
          {
               //失败，TODO......
          }
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置电子变倍 SGP_SetElectronicMagnification

选项：	说明
描述：	设置电子变倍，只对主码流有效
详细描述：	
函数：	int SGP_SetElectronicMagnification(SGP_HANDLE handle, SGP_VIDEO_PARAM_ENUM type, int magnification);
参数：	handle [in] 传入设备对象 type [in] 视频类型值 input [in] 1：红外原始，可见光原始 2：红外2倍，可见光4倍 3：红外3倍，可见光16倍
返回值：	成功返回SGP_OK,失败返回错误码
备注：	

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	**/
	void Init()
	{
	int magnification = 2;
	SGP_VIDEO_PARAM_ENUM type = 3;

```
     int ret =
SGP_SetElectronicMagnification(handle,type,magnification);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

获取报警输入 SGP_GetAlarmInput

选项：	说明
描述：	获取报警输入
详细描述：	
函数：	int SGP_GetAlarmInput(SGP_HANDLE handle, SGP_ALARM_INPUT_INFO *output);
参数：	handle [in] 传入设备对象 output [out] 输出信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_ALARM_INPUT_INFO info;

```
     memset(&info, 0x00,
sizeof(SGP_ALARM_INPUT_INFO));
     int ret =
SGP_GetAlarmInput(handle,&info);
     if (ret == SGP_OK )
     {

          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

设置报警输入 SGP_SetAlarmInput

选项：	说明
描述：	设置报警输入
详细描述：	
函数：	int SGP_SetAlarmInput(SGP_HANDLE handle, SGP_ALARM_INPUT_INFO input);
参数：	handle [in] 传入设备对象 input [in] 报警信息
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() { SGP_ALARM_INPUT_INFO info;

```
     memset(&info, 0x00,
sizeof(SGP_ALARM_INPUT_INFO));
     int ret =
SGP_GetAlarmInput(handle,&info);
     if (ret == SGP_OK )
     {
          info.flag   = 1;
          ret =
SGP_SetAlarmInput(handle,info);
          if (ret == SGP_OK )
          {
               //成功，TODO......
          }
          else
          {
               //失败，TODO......
          }
     }
     else
     {
          //失败，TODO......
     }
 }
```

恢复出厂设置 SGP_FactoryReset

选项：	说明
描述：	恢复出厂设置
详细描述：	
函数：	int SGP_FactoryReset(SGP_HANDLE handle);
参数：	handle [in] 传入设备对象
返回值：	成功返回SGP_OK,失败返回错误码
备注：	

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle 设备对象。
	**/
	void Init()
	{
	int ret = SGP_FactoryReset(handle);
	if (ret == SGP_OK)
	{
	//成功，TODO.....
	}

```
        else
        {
            //失败，TODO......
        }
}
```

透传RS485数据查询 SGP_CommandSend

选项：	说明
描述：	透传RS485数据查询
详细描述：	
函数：	int SGP_CommandSend(SGP_HANDLE handle, const char *data);
参数：	handle [in] 传入设备对象 data [in] 传入查询指令 例如："05030000006705A4"
返回值：	成功返回SGP_OK,失败返回错误码
备注：	
使用示例：	/** 示例中部分类、变量、函数的解释： 1，handle 设备对象。 **/ void Init() {

```
     const char* data=
"05030000006705A4";
     int ret =
SGP_CommandSend(handle,data);
     if (ret == SGP_OK )
     {
          //成功，TODO......
     }
     else
     {
          //失败，TODO......
     }
 }
```

注册温度告警回调函数
SGP_RegisterTempAlarmCallback

选项：	说明
描述：	注册温度告警回调函数
详细描述：	
函数：	void SGP_RegisterTempAlarmCallback(SGP_HANDLE handle, SGP_TEMPALARMCALLBACK callback, void *pUser);
参数：	handle [in] 传入设备对象 callback [in] 回调函数地址 pUser [in] 回调函数传入参数,例如QT，可以传入this指针
返回值：	无
备注：	
使	/**

用示例：	示例中部分类、变量、函数的解释：
	1，handle　设备对象。
	2,以QT界面库为例
	**/
	static void TempAlarm(SGP_TEMPALARMNOTIFY notify, void *pUser)
	{
	MainWindow *pDlg = (MainWindow *)pUser;
	printf("获取的高温温度是%f\n", notify.high_temp);
	printf("获取的低温温度是%f\n", notify.low_temp);
	printf("获取的平均温度是%f\n", notify.avg_temp);
	printf("获取的报警类型是%d\n", notify.temp_flag);
	//TODO......
	}
	void MainWindow::Init()
	{
	SGP_RegisterTempAlarmCallback(handle, TempAlarm, this);
	//TODO......
	}

回调函数描述		
函数名称	typedef void(*SGP_TEMPALARMCALLBACK)(SGP_TEMPALARMNOTIFY notify, void *pUser);	
功能描述	温度告警回调函数	
参数说明	notify	输出参数
	pUser	输出参数
返回值	无	

注册对象温差告警回调函数
SGP_RegisterObjTempAlarmCallback

选项：	说明
描述：	注册对象温差告警回调函数
详细描述：	
函数：	void SGP_RegisterObjTempAlarmCallback(SGP_HANDLE handle, SGP_OBJTEMPALARMCALLBACK callback, void *pUser);
参数：	handle [in] 传入设备对象 callback [in] 回调函数地址 pUser [in] 回调函数传入参数,例如QT，可以传入this指针
返回值：	无
备注：	
使	/**

用示例：	示例中部分类、变量、函数的解释：
	1，handle　设备对象。
	2,以QT界面库为例
	**/
	static void TempAlarm(SGP_OBJTEMPALARMNOTIFY notify, void *pUser)
	{
	MainWindow *pDlg = (MainWindow *)pUser;
	printf("获取的分析对象1的温度是%f\n", notify.fTemp1);
	printf("获取的分析对象2的温度是%f\n", notify.fTemp2);
	printf("获取的分析对象1、2的温差值是%f\n", notify.fTempDiff);
	printf("获取的对象温差比较的判断条件是%d\n", notify.iTempFlag);
	//TODO......
	}
	void MainWindow::Init()
	{
	SGP_RegisterObjTempAlarmCallback(handle, TempAlarm, this);
	//TODO......
	}

回调函数描述		
函数名称	typedef void(*SGP_OBJTEMPALARMCALLBACK)(SGP_OBJTEMPALARMNOTIFY notify, void *pUser);	
功能描述	对象温差告警回调函数	
参数说明	notify	输出参数
	pUser	输出参数
返回值	无	

注册内存已满回调函数
SGP_RegisterMemoryFullCallback

选项：	说明
描述：	注册内存已满回调函数
详细描述：	
函数：	void SGP_RegisterMemoryFullCallback(SGP_HANDLE handle, SGP_MEMORYFULLCALLBACK callback, void *pUser);
参数：	handle [in] 传入设备对象 callback [in] 回调函数地址 pUser [in] 回调函数入参
返回值：	无
备注：	
使用示	/** 示例中部分类、变量、函数的解释：

例：

```
 1，handle  设备对象。
 2,以QT界面库为例
 **/
 static void
MemoryFull(SGP_MEMORYFULLNOTIFY notify
,void *pUser)
 {
   MainWindow *pDlg = (MainWindow
*)pUser;
   printf("总存储是%dM\n",
notify.total);
   printf("可用大小%dM\n",
notify.free);
   printf("报警阈值%dM\n",
notify.limit);

   //TODO......
 }
 void MainWindow::Init()
 {

SGP_RegisterMemoryFullCallback(handle,
MemoryFull, this);
    //TODO......
 }
```

回调函数描述

函数名称	typedef void(*SGP_MEMORYFULLCALLBACK)(SGP_MEMORYFULLNOTIFY notify, void *pUser);	
功能描述	内存已满回调函数	
参数说明	notify	输出参数
	pUser	输出参数
返回值	无	

注册存储故障回调函数
SGP_RegisterStorageErrorCallback

选项：	说明
描述：	注册存储故障回调函数
详细描述：	
函数：	void SGP_RegisterStorageErrorCallback(SGP_HANDLE handle, SGP_STORAGEERRORCALLBACK callback, void *pUser);
参数：	handle [in] 传入设备对象 callback [in] 回调函数地址 pUser [in] 回调函数入参
返回值：	无
备注：	
使用示	/** 示例中部分类、变量、函数的解释：

例：	1，handle　设备对象。
	2,以QT界面库为例
	**/
	static void StorageError(void *pUser)
	{
	MainWindow *pDlg = (MainWindow *)pUser;
	printf("Storage Error\n");
	//TODO......
	}
	void MainWindow::Init()
	{
	SGP_RegisterStorageErrorCallback(handle, StorageError, this);
	//TODO......
	}

回调函数描述	
函数名称	typedef void(*SGP_STORAGEERRORCALLBACK)(void *pUser);
功能	存储故障回调函数

描述		
参数说明	pUser	输出参数
返回值	无	

注册推流异常回调函数
SGP_RegisterRtspErrorCallback

选项：	说明
描述：	注册推流异常回调函数
详细描述：	
函数：	void SGP_RegisterRtspErrorCallback(SGP_HANDLE handle, SGP_RTSPERRORCALLBACK callback, void *pUser);
参数：	handle [in] 传入设备对象 callback [in] 回调函数地址 pUser [in] 回调函数入参
返回值：	无
备注：	
使用示	/** 示例中部分类、变量、函数的解释：

例：

```
 1，handle   设备对象。
 2,以QT界面库为例
 **/
 static void RtspError(int type,void
*pUser)
 {
   MainWindow *pDlg = (MainWindow
*)pUser;
   printf("类型是%d\n", type);
   //TODO......
 }
 void MainWindow::Init()
 {

SGP_RegisterRtspErrorCallback(handle,
RtspError, this);
    //TODO......
 }
```

回调函数描述	
函数名称	typedef void(*SGP_RTSPERRORCALLBACK)(int type, void *pUser);

功能描述	推流异常回调函数	
参数说明	type	输出参数
参数说明	pUser	输出参数
返回值	无	

注册非法访问回调函数
SGP_RegisterAccessViolationCallback

选项：	说明
描述：	注册非法访问回调函数
详细描述：	
函数：	void SGP_RegisterAccessViolationCallback(SGP_HANDLE handle, SGP_ACCESSVIOLATIONCALLBACK callback, void *pUser);
参数：	handle [in] 传入设备对象 callback [in] 回调函数地址 pUser [in] 回调函数入参
返回值：	无
备注：	
使用示	/** 示例中部分类、变量、函数的解释：

例：

```
 1，handle   设备对象。
 **/
 static void
AccessViolation(SGP_ACCESSVIOLATIONNOTIFY
notify, void *pUser)
 {
   MainWindow *pDlg = (MainWindow *)pUser;
   printf("异常登录用户名是%s\n",
notify.user);
   printf("异常登录IP是%s\n", notify.ip);
   printf("异常登录时间是%s\n", notify.time);

   //TODO......
 }
 void MainWindow::Init()
 {

SGP_RegisterAccessViolationCallback(handle,
AccessViolation, this);
    //TODO......

 }
```

回调函数描述	
函数	typedef void(*SGP_ACCESSVIOLATIONCALLBACK)

名称	(SGP_ACCESSVIOLATIONNOTIFY notify, void *pUser);	
功能描述	非法访问回调函数	
参数说明	notify	输出参数
	pUser	输出参数
返回值	无	

注册网络异常回调函数
SGP_RegisterNetworkErrorCallback

选项：	说明
描述：	注册网络异常回调函数
详细描述：	
函数：	void SGP_RegisterNetworkErrorCallback(SGP_HANDLE handle, SGP_NETWORKERRORCALLBACK callback, void *pUser);
参数：	handle [in] 传入设备对象 callback [in] 回调函数地址 pUser [in] 回调函数入参
返回值：	无
备注：	
使用示	/** 示例中部分类、变量、函数的解释：

例：

```
 1，handle  设备对象。
 **/
 static void
NetworkError(SGP_NETWORKERRORNOTIFY
notify, void *pUser)
 {
   MainWindow *pDlg = (MainWindow
*)pUser;
   printf("类型是%d\n", notify.type);
   printf("IP是%s\n", notify.ip);

   //TODO......
 }
 void MainWindow::Init()
 {

SGP_RegisterNetworkErrorCallback(handle,
NetworkError, this);
    //TODO......

 }
```

回调函数描述	
函	typedef

数名称	void(*SGP_NETWORKERRORCALLBACK) (SGP_NETWORKERRORNOTIFY notify, void *pUser);	
功能描述	网络异常回调函数	
参数说明	notify	
	pUser	
返回值	无	

注册外部告警回调函数
SGP_RegisterAlarmInputCallback

选项：	说明
描述：	注册外部告警回调函数
详细描述：	
函数：	void SGP_RegisterAlarmInputCallback(SGP_HANDLE handle, SGP_ALARMINPUTCALLBACK callback, void *pUser);
参数：	handle [in] 传入设备对象 callback [in] 回调函数地址 pUser [in] 回调函数入参
返回值：	无
备注：	
使用示	/** 示例中部分类、变量、函数的解释：

例：

```
 1，handle  设备对象。
 **/
 static void
AlarmInput(SGP_ALARMINPUTCALLBACK
notify, void *ptr)
 {
   MainWindow *pDlg = (MainWindow
*)ptr;
   printf("报警时间是%s\n",
notify.time);
   printf("红外JPEG图片得BASE64格式
是%s\n", notify.ir_image_content);
   printf("可见光录像地址是%s\n",
notify.vl_video_url);
   printf("红外录像地址是%s\n",
notify.ir_video_url);

   //TODO......
 }
 void MainWindow::Init()
 {

SGP_RegisterAlarmInputCallback(handle,
AlarmInput, this);
    //TODO......

 }
```

回调函数描述		
函数名称	typedef void(*SGP_ALARMINPUTCALLBACK)(SGP_ALARMINPUTNOTIFY notify, void *pUser);	
功能描述	外部告警回调函数	
参数说明	notify	
	pUser	
返回值	无	

注册火警报警回调函数
SGP_RegisterFireAlarmCallback

选项：	说明
描述：	注册火警报警回调函数
详细描述：	
函数：	void SGP_RegisterFireAlarmCallback(SGP_HANDLE handle, SGP_FIREALARMCALLBACK callback, void *pUser);
参数：	handle [in] 传入设备对象 callback [in] 回调函数地址 pUser [in] 回调函数入参
返回值：	无
备注：	
使用示	/** 示例中部分类、变量、函数的解释：

例：

```
 1，handle   设备对象。
 **/
 static void
AlarmInput(SGP_FIRE_ALARM  notify,
void *ptr)
 {
   MainWindow *pDlg = (MainWindow
*)ptr;
   printf("报警时间是%s\n",
notify.time);
   printf("红外JPEG图片得BASE64格式
是%s\n", notify.ir_image_url );
   printf("可见光录像地址是%s\n",
notify.vl_video_url);
   printf("红外录像地址是%s\n",
notify.ir_video_url);

   //TODO......
 }
 void MainWindow::Init()
 {

SGP_RegisterFireAlarmCallback(handle,
AlarmInput, this);
    //TODO......

 }
```

回调函数描述		
函数名称	typedef void(*SGP_FIREALARMCALLBACK)(SGP_FIRE_ALARM notify, void *pUser);	
功能描述	火灾报警回调函数	
参数说明	notify	
	pUser	
返回值	无	

注册自动调焦回调函数
SGP_RegisterAutoFocusCallback

选项：	说明
描述：	注册自动调焦回调函数
详细描述：	
函数：	void SGP_RegisterAutoFocusCallback (SGP_HANDLE handle, SGP_AUTOFOCUSCALLBACK callback, void *pUser);
参数：	handle [in] 传入设备对象 callback [in] 回调函数地址 pUser [in] 回调函数入参
返回值：	无
备注：	

使用示例：	/**
	示例中部分类、变量、函数的解释：
	1，handle　设备对象。
	**/
	static void GetFocusResult (int result , void *ptr)
	{
	　printf("Focus result is %d\n", result);
	　//TODO......
	}
	void MainWindow::Init()
	{
	SGP_RegisterAutoFocusCallback (handle, GetFocusResult , this);
	　//TODO......
	}

回调函数描述	
函数名	typedef void(*SGP_AUTOFOCUSCALLBACK)(

称	int result, void *pUser);	
功能描述	自动调焦回调函数	
参数说明	result	0:调焦完成，结果不清晰 1:调焦完成，结果清晰
	pUser	
返回值	无	

结构体定义描述

```
struct SGP_ACCESS_VIOLATION_INFO
{
    int audio_flag;//是否音频联动 0:否; 1:是
    int audio_index;//音频文件索引0-2
    int audio_mode;//音频模式 0:持续时间; 1:播放次数
    int audio_value;//音频模式值 0-100（次/秒）
    int allow_count;//允许登录次数3-10次
    int flag;//是否开启 0:不开启; 1:开启
    int sendmail;//是否发送邮件 0:否; 1:是
    int light_flag;//是否闪光灯 0:否; 1:是
    int light_hold;//闪光灯持续时间10-300s
    int output_flag;//是否外部输出 0:不输出 1:输出
    int output_hold;//外部输出持续时间10-300s
};
```

结构体定义描述

```
struct SGP_ACCESSVIOLATIONNOTIFY
{
    char user[STRING_LENGH];//异常登录用户
    char ip[STRING_LENGH];  //异常登录IP
    char time[STRING_LENGH];//异常登录时间
};
```

结构体定义描述

```
struct SGP_ALARM_INPUT_INFO
{
    int flag;//是否开启 0 不开启 1 开启
    int alarm_shake;//报警抖动0-100s
    int type;//输入类型：0 常开型  1 常闭型
    int record_delay;//录制延时 10-300
    int record_flag;//是否录制 0:不录制; 1:录制
    int record_stream;//录制类型 0:不录制; 1:只录制可见光; 2:只录制红外; 3:录制红外和可见光
    int capture_flag;//是否截图 0:否; 1:是
    int capture_stream;//截图类型 0:不截图; 1:只截图可见光; 2:只截图红外; 3:截图红外和可见光
    int sendmail;//是否发送邮件 0:不发送; 1:发送
    int light_flag;//是否开启闪光灯 0:否; 1:是
    int light_hold;//闪光灯持续时间，10-300s
    int output_flag;//是否外部输出 0:不输出 1:输出
    int output_hold;//外部输出持续时间10-300s
    int audio_flag;//是否音乐提醒 1:是; 0:否
    int audio_index;//音乐文件索引，0-2
    int audio_mode;//音乐播放模式 1:播放次数; 2:持续时间
```

```
    int audio_value;//音乐播放值,随模式定义:(持续时间:秒数)(播放次数:播放次数0-100)
    int effect_day_num;//时间数组数量
    SGP_EFFECT_DAY effect_day[7];//时间数组
};
```

结构体定义描述

```
struct SGP_ALARMINPUTNOTIFY
{
    char time[STRING_LENGH];//报警时间，格式为2020-05-21 12:22:33
    char vl_image_url[STRING_LENGH];//报警记录可见光截图，http jpeg路径
    char ir_image_url[STRING_LENGH];//报警记录红外截图，http jpeg路径
    char vl_image_content[STRING_LENGH];//可见光JPEG图片得BASE64格式
    char ir_image_content[STRING_LENGH];//红外JPEG图片得BASE64格式
    char vl_video_url[STRING_LENGH];//可见光录像地址
    char ir_video_url[STRING_LENGH];//红外录像地址
};
```

结构体定义描述

struct SGP_FIRE_ALARM
{
float high_temp;//高温温度,高温报警时有效
float low_temp;//低温温度，低温报警时有效
float avg_temp;//平均温度，平均温报警时有效
char time[STRING_LENGH];//报警时间，格式为2020-05-21 12:22:33
char capture_time[STRING_LENGH];//报警抓图时间，格式为2020-05-21 12:22:33
char vl_image_url[STRING_LENGH];//报警记录可见光截图，http jpeg路径
char ir_image_url[STRING_LENGH];//报警记录红外截图，http jpeg路径
char vl_video_url[STRING_LENGH];//可见光录像地址
char ir_video_url[STRING_LENGH];//红外录像地址
};

结构体定义描述

```
struct SGP_ANALYTIC_TEMP
{
    int rule_id;     //规则id
    char rule_name[STRING_LENGH];   //规则名称 32字符以内
    int type;//对象类型 1点; 2线; 3矩形; 4多边形; 5圆形
    float max_temp;//最高温度值
    float min_temp;//最低温度值
    float avg_temp;//平均温度值
};
```

结构体定义描述

```
struct SGP_ANALYTIC_TEMPS
{
    int analytic_num;
    SGP_ANALYTIC_TEMP analytic[ANALYTIC_MAX_NUM];
    float global_max_temp;//全局最高温度值
    float global_min_temp;//全局最低温度值
    float global_avg_temp;//全局平均温度值
};
```

结构体定义描述

```
struct SGP_COLD_HOT_TRACE_INFO
{
    int light_hold;//闪光灯持续时间 10-300s
    int light_flag;//是否开启闪光灯 0:否; 1:是
    int alarm_shake;//报警抖动,单位s,0-100
    int capture_flag;//是否截图 0:否; 1:是
    int capture_stream;//截图类型 1:只截图可见光; 2:只截图红外; 3:截图红外和可见光 (web2.0 截图类型 0:不截图)
    char high_color[STRING_LENGH];//高温颜色:0xRGB
    int high_flag;//高温是否检测 0:不检测; 1:检测
    float high_temp;//高温温度，-40~2000
    char low_color[STRING_LENGH];//低温颜色:0xRGB
    int low_flag;//低温是否检测 0:不检测; 1:检测
    float low_temp;//低温温度，-40~2000
    int record_delay;//录像时间, 10~300s
    int record_flag;//是否录制 0:不录制; 1:录制
    int record_stream;//录制类型 1:只录制可见光; 2:只录制红外; 3:录制红外和可见光 (web2.0 录制类型 0:不录制)
    int sendmail;//是否发送邮件 0:不发送; 1:发送
    int trace_flag;//是否开启 0:不开启; 1:开启
```

```
    int effect_day_num;//时间数组数量
    int output_flag;//是否外部输出 0:不输出
1:输出
    int output_hold;//外部输出持续时间 10-
300s
    int audio_flag;//是否音乐提醒 1:是; 0:
否
    int audio_index;//音乐文件索引，0-2
    int audio_mode;//音乐播放模式 1:播放次
数; 2:持续时间
    int audio_value;//音乐播放值,随模式定
义:(持续时间:秒数)(播放次数:播放次数0-100)
    SGP_EFFECT_DAY effect_day[7];//时间数
组
    int high_condition;//全局最高温对应的控
制条件, 1:大于，0:小于
    int low_condition;//全局最低温对应的控
制条件, 1:大于，0:小于
};
```

结构体定义描述

```
struct SGP_CONFIG
{
    int type;//报警类型 1:高温报警; 2:低温报警; 3:平均温报警; 4:高低温报警
    int condition;//条件 1:高于; 2:低于 3:匹配;
    float high_temp;//配置高温
    float low_temp;//配置低温
    float avg_temp;//配置平均温
    int objtype;//类型 0:冷热点; 1:点; 2:线; 3:矩形; 4:多边形;5:圆形
    SGP_POINT points[7];
};
```

结构体定义描述

```
struct SGP_EFFECT_DAY
{
    int day;//1-7,星期几
    int period_num;//时间段数量
    SGP_PERIOD period[6];//时间段
};
```

结构体定义描述

```
struct SGP_EMAIL_INFO
{
    int alarm;//是否使用报警邮件 0:否;1:是
    int alarm_value;//报警邮件间隔 1-3600
秒
    int enclosure;//是否带附件 0:否; 1:是
    int encry_type;//加密方式 0:none;
1:tls; 2:ssl
    char from[STRING_LENGH];//发件人
    int health;//是否使用健康邮件 0:否; 1:
是
    int health_value;//健康邮件间隔 1-3600
秒
    int is_anon;//是否匿名 0:否; 1:是
    char password[STRING_LENGH];//登录密
码，密文传输
    int smtp_port;//smtp服务端口，默认25
    char
smtp_server[STRING_LENGH];//smtp服务器,默
认空xxx.xxx.xxx.xxx
    char subject[STRING_LENGH];//主题
    char username[STRING_LENGH];//登录服
务器用户名
    int mailto_num;//收件人数量
    char mailto[5][STRING_LENGH];//收件人
列表
};
```

结构体定义描述

struct SGP_FILL_LIGHT_INFO
{
int brightness;/*亮度 0-100,0 - 20一档;21 - 40二档;41 - 60三档;61 - 80四档;81 - 100五档*/
int light;//灯开启状态 0:关闭; 1:开启
int mode;//灯模式 0:手动; 1:自动
};

结构体定义描述

```
enum SGP_FOCUS_TYPE
{
    SGP_FOCUS_STOP = 0,         //电机停止
    SGP_FOCUS_FAR = 1,          //远焦
    SGP_FOCUS_NEAR = 2,         //近焦
    SGP_FOCUS_FAR_FINE = 3,     //远焦微调
    SGP_FOCUS_NEAR_FINE = 4,    //近焦微调
    SGP_FOCUS_AUTO = 5,         //自动聚焦
    SGP_FOCUS_PLACE = 6,        //设置位置
};
```

结构体定义描述

```
struct SGP_GENERAL_INFO
{
        char datetime[STRING_LENGH];//系统时间，格式为2020-05-21 12:55:12
        char ir_rtsp_url[STRING_LENGH];//红外主码流rtsp地址
        char ir_sub_rtsp_url[STRING_LENGH];//红外辅码流rtsp地址
        int ir_model_w;//红外模组宽
        int ir_model_h;//红外模组高
        int ir_output_w;//红外通道输出宽
        int ir_output_h;//红外通道输出高
        int range_num;//测温范围数量
        SGP_RANGE range[RANGE_MAX_NUM];//测温范围
        char vl_rtsp_url[STRING_LENGH];//可见光主码流rtsp地址(双光产品支持)
        char vl_sub_rtsp_url[STRING_LENGH];//可见光辅码流rtsp地址(双光产品支持)
};
```

结构体定义描述

```
struct SGP_IAMGE_EFFECT_PARAM_IR_CONFIG
{
    int auto_shutter;//快门自动补偿时间1-20(单位分钟)
    int brightness;//亮度，取值范围0-100
    int contrast;//对比度，取值范围0-100
    int reverse;//是否反转，0:不反转 1 反转
    int time_flag;//降噪时域滤波开关:0关闭;1开启
    int time_value;//降噪时域滤波值 0-100
    int space_flag;//降噪空域滤波开关:0关闭;1开启
    int space_value;//降噪空域滤波值 0-100
    int iee_flag;//细节增强开关:0关闭;1开启
    int iee_value;//细节增强值0-100
    int saturation;//饱和度，取值范围0-100(红外设备不支持)
    int sharpness;//锐度，取值范围0-100
     int rotate;//旋转参数（顺时针，0:0°，1:90°，2:  180°，3:270°）
};
```

结构体定义描述

```
struct SGP_IAMGE_EFFECT_PARAM_VL_CONFIG
{
    int blc;//背光补偿:0关闭; 1上; 2下; 3
左; 4右; 5中; 6自动
    int brightness;//亮度，取值范围0-100
    int contrast;//对比度，取值范围0-100
    int exp;//曝光补偿: 0-100
    int hlc;//强光抑制:0关闭;1开启
    int reverse;//是否反转，0:不反转 1 反转
    int saturation;//饱和度，取值范围0-100
    int sharpness;//锐度，取值范围0-100
    int wdr;//宽动态 0:关闭; 1:20%;
2:40%; 3:60%; 4:80%; 5:100%
};
```

结构体定义描述

```
struct SGP_IMAGE_FUSION
{
    int percent;//融合比例值0-100
    int ir_left;//红外图像左边裁剪像素值0～50
    int ir_right;//红外图像右边裁剪像素值0～50
    int ir_top;//红外图像上边裁剪像素值0～50
    int ir_botton;//红外图像下边裁剪像素值0～50
    int vl_left;//可见光图像左边裁剪像素值0～1000
    int vl_right;//可见光图像右边裁剪像素值0～1000
    int vl_top;//可见光图像上边裁剪像素值0～1000
    int vl_botton;//可见光图像下边裁剪像素值0～1000
    SGP_IMAGE_FUSION_MATCH_POINTS ir_match_points;//
红外校准点
    SGP_IMAGE_FUSION_MATCH_POINTS vl_match_points;//
可见光校准点
};
```

结构体定义描述

```
struct SGP_IMAGE_FUSION_MATCH_POINTS
{
    SGP_POINT point1;//第1个校准点
    SGP_POINT point2;//第2个校准点
    SGP_POINT point3;//第3个校准点
    SGP_POINT point4;//第4个校准点
    SGP_POINT point5;//第5个校准点
};
```

结构体定义描述

enum SGP_IMAGE_TYPE
{
SGP_VL_IMAGE = 1,//可见光图片
SGP_IR_IMAGE = 2,//红外图片
};

结构体定义描述

enum SGP_IR_IMAGE_EFFECT_ENUM
{
SGP_IR_AUTO_SHUTTER = 1,//快门自动补偿时间1-20(单位分钟)
SGP_IR_BRIGHTNESS = 2,//亮度，取值范围0-100
SGP_IR_CONTRAST = 3,//对比度，取值范围0-100
SGP_IR_REVERSE = 4,//是否反转，0:不反转 1 反转
SGP_IR_TIME_FLAG = 5,//降噪时域滤波开关:0关闭;1开启
SGP_IR_TIME_VALUE = 6,//降噪时域滤波值0-100
SGP_IR_SPACE_FLAG = 7,//降噪空域滤波开关:0关闭;1开启
SGP_IR_SPACE_VALUE = 8,//降噪空域滤波值 0-100
SGP_IR_IEE_FLAG = 9,//细节增强开关:0关闭;1开启
SGP_IR_IEE_VALUE = 10,//细节增强值0-100
SGP_IR_SATURATION = 11,//饱和度，取值范围0-100
SGP_IR_SHARPNESS = 12,//锐度，取值范围0-100
SGP_IR_ROTATE = 13,//旋转
};

结构体定义描述

struct SGP_MEMORYFULLNOTIFY
{
int total;//总存储，单位M
int free;//可用大小，单位M
int limit;//报警阈值，可用小于报警阈值时报警，单位M
};

结构体定义描述

```
struct SGP_NET_EXCEPTION_INFO
{
    int audio_flag;//是否音频联动 0:否; 1:是
    int audio_index;//音频文件索引0-2
    int audio_mode;//音频模式 0:持续时间; 1:播放次数
    int audio_value;//音频模式值 0-100（次/秒）
    int flag;//是否开启 0:不开启; 1:开启
    int light_flag;//是否闪光灯 0:否; 1:是
    int light_hold;//闪光灯持续时间10-300s
    int output_flag;//是否外部输出 0:不输出 1:输出
    int output_hold;//外部输出持续时间1-300s
};
```

结构体定义描述

```
struct SGP_NET_INFO
{
    int card;//网卡类型:0有线网卡
    char dns1[STRING_LENGH];//dns服务器
xxx.xxx.xxx.xxx
    char dns2[STRING_LENGH];//dns服务器
xxx.xxx.xxx.xxx
    char gateway[STRING_LENGH];//网关
xxx.xxx.xxx.xxx
    char host_name[STRING_LENGH];//主机名
    int ip_version;//版本 0:ipv4; 1:ipv6
    char ipaddr[STRING_LENGH];//网络ip地
址xxx.xxx.xxx.xxx
    char mac[STRING_LENGH];//Mac地址
    int mode;//模式 0:静态; 1:动态
    char netmask[STRING_LENGH];//子网掩码
xxx.xxx.xxx.xxx
};
```

结构体定义描述

struct SGP_NETWORKERRORNOTIFY
{
int type;//类型 1:ip冲突; 2:ping不通网关，ip为网关
char ip[STRING_LENGH];//ip地址
};

结构体定义描述

struct SGP_PERIOD
{
char start[STRING_LENGH];//开始时间，格式 HH:mm:ss
char end[STRING_LENGH];//结束时间，格式 HH:mm:ss
};

结构体定义描述

struct SGP_POINT
{
int x;//x坐标 范围 参照红外图像
int y;//y坐标 范围 参照红外图像
};

结构体定义描述

struct SGP_PORT_INFO
{
int http_port;//http服务器端口，默认端口80保留设置
int max_connectios;//最大web连接数，1-20
int onvif_check;//Onvif登录校验 0:不校验; 1:校验
int rtsp_port;//红外rtsp端口，1024-65535，端口用于rtsp流服务，默认端口554保留设置
int tcp_port;//web消息交互端口，不可设置
};

结构体定义描述

struct SGP_RANGE
{
int id;//测温档位类型 (低温:0,高温:1,其他:2)
int min;//测温范围最低温
int max;//测温范围最高温
};

结构体定义描述

```
struct SGP_RECORD_INFO
{
    int record_interval;//延时录制时间1-3600秒
    int record_max_size;//录制文件最大大小，单位M，1-1000
    int record_time;//录制时长，单位秒，1-3600分钟
};
```

结构体定义描述

struct SGP_RECT
{
int x;//x坐标， 1-图像宽
int y;//y坐标， 1-图像高
int w;//区域宽，与坐标共同作用，取值范围1-图像宽
int h;//区域高，与坐标共同作用，取值范围1-图像高
};

结构体定义描述

```
struct SGP_RULE
{
    int id;//分析对象id，内部分配，无需设置。
    int alarm_condition;//报警条件:1高于;2低于;3匹配;4高于和低于(web2.0支持1和2)
    int alarm_flag;//是否报警:0不需要;1需要
    int alarm_time;//去抖动时间,0-10秒
    int alarm_interal;//报警间隔时间，单位：秒。允许设置的数据为：30， 60， 300，600，900,1800，3600
   int alarm_type;//报警类型:1高温报警;2低温报警;3平均温报警;4最高温+最低温报警警(web2.0只支持1、2、3)
    float avg_temp;//平均温(基于设备的测温范围)
    int flag;//是否启用配置:0不启用;1启用
    float high_temp;//报警高温阈值(基于设备的测温范围)
    float low_temp;//报警低温阈值(基于设备的测温范围)
    int points_num;
    SGP_POINT points[7];//矩形，圆是四个点，顺时针顺序
    char rule_name[STRING_LENGH];//规则名称，支持50字符
    int show_location;//名称显示位置:1上方;2下方;3左方;4右方;5中间
    float temp_mod;//温度误差
    int type;//对象类型:1点;2线;3矩形;4多边形;5圆
```

```
    float atmo_trans;//大气透过率0.01-1
    float dist;//距离，单位米，0.1-20.0
    float emiss;//发射率 0.1-1.0
    int emiss_mode;//发射率类型:1标准;2自定义
    int humi;//湿度,范围1-100
    float opti_trans;//光学透过率0.01-1
    float ref_temp;//反射温度-20~550,单位摄氏度(web2.0 -40到2000)
    int show_type;//显示内容，范围1~8,1~5(最高温，最低温，平均温度，仅名称，不显示),6~8属于预留部分
};
```

结构体定义描述

struct SGP_RULE_ARRAY
{
int rule_num;
SGP_RULE rule[ANALYTIC_MAX_NUM];//规则列表
};

结构体定义描述

struct SGP_SHIELD_AREA_INFO
{
int rect_num;
SGP_RECT rect[SHIELD_AREA_MAX_NUM];//区域数组,左上角坐标0, 0标准,最多支持两个
};

结构体定义描述

```
enum SGP_SHUTTER_ENUM
{
    SGP_SHUTTER = 1,//快门操作
    SGP_SHUTTER_OPEN = 2,//快门常开
    SGP_SHUTTER_CLOSE = 3,//快门常闭
    SGP_SHUTTER_AUTO = 4,//自动快门
};
```

结构体定义描述

```
struct SGP_TEMP_ALARM_INFO
{
    int audio_flag;//是否音乐提醒 1:是; 0:否
    int audio_index;//音乐文件索引，0-2
    int audio_mode;//音乐播放模式 1:播放次数; 2:持续时间
    int audio_value;//音乐播放值,随模式定义:(持续时间:秒数)(播放次数:播放次数0-100)
    int alarm_flag;//是否开启报警 1:开启; 0:不开启（新web上该字段弃用）
    int light_hold;//闪光灯持续时间，10-300s
    int light_flag;//是否开启闪光灯 0:否; 1:是
    int alarm_shake;//报警抖动0-100s(新web上仅在全局温度-报警参数设置-去抖动，这个功能上使用)
    int capture_flag;//是否截图 0:否; 1:是（web2.0上该字段弃用）
    int capture_stream;//截图类型 0:不截图; 1:只截图可见光; 2:只截图红外; 3:截图红外和可见光
    int record_delay;//录制时间 10-300s
    int record_flag;//是否录制 0:不录制; 1:录制（web2.0上该字段弃用）
    int record_stream;//录制类型 0:不录制; 1:只录制可见光; 2:只录制红外; 3:录制红外和可见光
    int sendmail;//是否发送邮件 0:不发送; 1:发送
```

```
    int effect_day_num;//时间数组数量
    int output_flag;//是否外部输出 0:不输出
1:输出
    int output_hold;//外部输出持续时间10-
300s
    SGP_EFFECT_DAY effect_day[7];//时间数
组
};
```

结构体定义描述

```
struct SGP_TEMPALARMNOTIFY
{
    char vl_image_url[STRING_LENGH];//报警记录可见光截图
    char vl_video_url[STRING_LENGH];//可见光视频地址
    char ir_image_url[STRING_LENGH];//报警记录红外截图
    char ir_video_url[STRING_LENGH];//红外视频地址
    float high_temp;//高温温度,高温报警时有效
    float low_temp;//低温温度,低温报警时有效
    float avg_temp;//平均温度,平均温报警时有效
    int temp_flag;//报警类型，0代表平均温，1代表高温报警，2代表低温报警，3代表高低温报警(web2.0 不支持)
    int type;//1:温度报警; 2:热点报警; 3:冷点报警(web2.0 不支持)
    char name[STRING_LENGH];//名称
    char time[STRING_LENGH];//报警时间，格式为2020-05-21 12:22:33
    SGP_CONFIG config;//配置
};
```

结构体定义描述

```
struct SGP_OBJTEMPALARMNOTIFY
{
    char vl_image_url[200];//报警记录可见光截图地址
    char vl_video_url[200];//可见光视频地址
    char ir_image_url[200];//报警记录红外截图地址
    char ir_video_url[200];//红外视频地址

    char obj_name1[STRING_LENGH];//分析对象1的名称
    char obj_name2[STRING_LENGH];//分析对象2的名称
    float fTemp1;//分析对象1的温度
    float fTemp2;//分析对象2的温度
    float fTempDiff;//分析对象1、2的温差值
    float fTempThreshold;//分析对象1、2的温差阈值
    int iTempFlag;//判断条件:  0:温差大于阈值; 1:温差小于阈值
    float iTempType1;//分析对象1的温度类型，0: 最高温，1:最低温， 2:平均温
    float iTempType2;//分析对象2的温度类型，0: 最高温，1:最低温， 2:平均温
    char name[STRING_LENGH];//名称
    char time[STRING_LENGH];//报警时间，格式为2020-05-21 12:22:33
};
```

结构体定义描述

```
struct SGP_THERMOMETRY_PARAM
{
    int color_bar;//色带1-26
    int color_show;//色带显示0~1
    int flag;//测温开关0~1
    float mod_temp;//温度修正
    int show_mode;//温度显示方式: 1 最高温 2 最低温 3 平均温 4 最高温 + 最低温
                          5 最高温 + 平均温 6 平均温 + 最低温 7 最高温 + 最低温 + 平均温 8不显示
    int gear;//测温范围
    int show_string; //是否使用字符叠加 其他机型 1:关闭; 2，4，5:右下; 3:右上
                          //IPM630 1:关闭; 5:右下
                          //IPT640M 1:关闭; 2:左上; 3:右上; 4:左下; 5:右下
    char show_desc[STRING_LENGH];//显示字符串
    float atmo_trans;//大气透过率0.01-1
    float dist;//距离，单位米，0.1-20.0
    float emiss;//发射率 0.1-1.0
    int emiss_mode;//发射率类型:1标准;2自定义
    int humi;//湿度,范围1-100
    float opti_trans;//光学透过率0.01-1
    float ref_temp;//反射温度-20~550,单位摄氏度(web2.0的反射温度范围: -40-2000)
    int isot_flag;//等温线开关0:关闭;1开启(工业机芯支持)
```

```
    float isot_high;//高温阈值0~400
    char
isot_high_color[STRING_LENGH];//高温颜色,
十六进制值,如红色:0xff0000
    int isot_low;//低温阈值-50~-100
    char isot_low_color[STRING_LENGH];//
低温颜色,十六进制值,如红色:0xff0000
    int isot_type;//范围类型:1 关闭等温线效
果 2 开启高等温线 3 开启低等温线 4 开启区间内
等温线 5 开启区间外等温线
    float ambient;//环境温度
};
```

结构体定义描述

```
struct SGP_VERSION_INFO
{
    char model[STRING_LENGH];
//设备型号
    char version[STRING_LENGH];
//系统版本
    char serial[STRING_LENGH];
//序列号
    char fpga_version[STRING_LENGH];
//FPGA版本
    char measure_version[STRING_LENGH];
//测温版本
    char sdk_version[STRING_LENGH];
//sdk版本
};
```

结构体定义描述

```
struct SGP_VIDEO_PARAM
{
    int bit_size;//主码流固定码流值，
可变码流时也需设置（宽*高*1.5*fps*8/压缩率）其中压缩率
范围（18-500）
如分辨率1280x720    取值范围：540Kb/s - 15000Kb/s
    int encodec;//主码流编码 0:h264; 1:h265;
2:mjpeg
    int fps;//主码流帧率1-25
    int gop_size;//主码流帧间隔1-50
    int level;//编码质量等级，等级效果随实际变化，如使
用ffmpeg，需服务端自映射（
默认medium，  可以上下延续几个等级）1:最好 2:更好 3:好
4:差 5:更差 6:最差
    int rate_control;//主码流控制 0:可变码流;1:固定
码流
    char ratio[STRING_LENGH];/*主码流分辨率
                                1920x1080
                                1280x960
                                1280x720
                                可见光辅码流分辨率
                                704x576
                                640x480
                                红外主码流分辨率
                                512x384(640*512)
                                红外辅码流分辨率
                                (384*288)256x192*/
    int svc;//帧率可分层编码，H264有效，其他格式也需传
入 0:分层编码; 1:不分层编码
};
```

结构体定义描述

```
enum SGP_VIDEO_PARAM_ENUM
{
    SGP_VL = 1,         //可见光主码流
    SGP_VL_SUB = 2,     //可见光辅码流
    SGP_IR = 3,         //红外主码流
    SGP_IR_SUB = 4,     //红外辅码流
};
```

结构体定义描述

enum SGP_VIDEO_TYPE
{
SGP_VL_VIDEO = 1,//可见光录像
SGP_IR_VIDEO = 2,//红外录像
};

结构体定义描述

```
enum SGP_VL_IMAGE_EFFECT_ENUM
{
    SGP_VL_BLC = 1,//背光补偿:0关闭; 1上; 2下; 3左; 4右; 5中; 6自动
    SGP_VL_BRIGHTNESS = 2,//亮度，取值范围0-100
    SGP_VL_CONTRAST = 3,//对比度，取值范围0-100
    SGP_VL_EXP = 4,//曝光补偿：0-100
    SGP_VL_HLC = 5,//强光抑制:0关闭;1开启
    SGP_VL_REVERSE = 6,//是否反转，0:不反转 1 反转
    SGP_VL_SATURATION = 7,//饱和度，取值范围0-100
    SGP_VL_SHARPNESS = 8,//锐度，取值范围0-100
    SGP_VL_WDR = 9,//宽动态 0:关闭; 1:20%; 2:40%; 3:60%; 4:80%; 5:100%
};
```

结构体定义描述

```
struct SGP_MEASURE_TEMP_INFO
{
    float jwtemp;//焦温温度
    float realshuttertemp;//实时快门温度
    float lastshuttertemp;//上次快门温度
    float realmirrortemp;//实时镜筒温度
    int    jwgears;//焦温档位
    int devgain;//探测器参数Gain
    int devint;//探测器参数Int
    int devres;//探测器参数Res
    int devgsk;//探测器参数gsk
    float centertemp;//全图中心温度
    float centermaxtemp;//全图最高温度
    float centermintemp;//全图最低温度
    int    centerx16;//中心点X16
    int centery16;//中心点Y16
    int avgshutter;//快门本底均值
    int rasel;//探测器参数RASEL
    int hssd;//探测器参数HSSD
    int gsktestnum;//探测器参数
gsk_test_num
    int gskval;//探测器参数gsk_val
    bool tempStabilityState;//焦温波动判断
设备稳定性
    float sharpnessleftup;//图像内左上区域
的清晰度
    float sharpnessrightup;//图像内右上区
域的清晰度
    float sharpnesscenter;//图像内中间区域
的清晰度
```

```
    float sharpnessleftdown;//图像内左下区域的清晰度
    float sharpnessrightdown;//图像内右下区域的清晰度
};
```

结构体定义描述

enum SGP_IR_IMAGE_INFO
{
    int denoise_flag_2d;//2D降噪开关
    int denoise_value_2d;//2D降噪值
    int denoise_flag_3d;//3D降噪开关
    int denoise_value_3d;//3D降噪值
};

结构体定义描述

enum SGP_IR_IMAGE_INFO_ENUM
{
SGP_2D_FLAG = 1,//2D降噪开关
SGP_2D_VALUE = 2,//2D降噪值
SGP_3D_FLAG = 3,//3D降噪开关
SGP_3D_VALUE = 4,//3D降噪值
SGP_SAVE_IR_IMAGE_INFO = 5,//保存参数信息
};

类型定义描述
typedef unsigned long long SGP_HANDLE;

宏定义

定义	数值	描述
STRING_LENGH	50	一般长度
RANGE_MAX_NUM	3	测温范围值
ANALYTIC_MAX_NUM	21	分析对象个数
SHIELD_AREA_MAX_NUM	2	屏蔽区域个数

错误码

错误码	定义	描述
0	SGP_OK	正常
1	SGP_ERR	错误
10001	SGP_ERR_10001	消息内容为空
10002	SGP_ERR_10002	消息内容无效
10003	SGP_ERR_10003	消息字段为空
10004	SGP_ERR_10004	无效用户名
10005	SGP_ERR_10005	未鉴权
10006	SGP_ERR_10006	密码修改失败
10007	SGP_ERR_10007	用户无权限操作
10008	SGP_ERR_10008	用户操作失败
10009	SGP_ERR_10009	密码错误
10010	SGP_ERR_10010	用户锁定
10011	SGP_ERR_10011	用户或密码错误
10012	SGP_ERR_10012	同版本升级
10013	SGP_ERR_10013	低版本升级
10014	SGP_ERR_10014	非法IP
10015	SGP_ERR_10015	IP冲突

10016	SGP_ERR_10016	非法子网掩码
10017	SGP_ERR_10017	非法网关
10018	SGP_ERR_10018	超过最大在线人数
10019	SGP_ERR_10019	DHCP错误
10020	SGP_ERR_10020	密码重复
10021	SGP_ERR_10021	请求温度矩阵时，正在打快门
10022	SGP_ERR_10022	请求温度矩阵时，正在切测温范围