短波红外相机
GH-SDK
软件开发手册

版本 SDK_V1.0

山西国惠光电科技有限公司

Shanxi guohui optoelectronic technology co. LTD

目　录

修订记录 **1**

1　概述 **2**

2　SDK 变量类型 **2**

2.1　一些自定义变量 2
2.2　设备枚举类型 2
2.3　相机设备信息结构 2
2.4　获取相机参数结构体 3
2.5　直方图信息结构体 4

3　SDK 主要函数(Api.h) **4**

3.1　初始化 IR_Init() 4
3.2　枚举设备 IR_EnumerateDevices() 4
3.3　查询设备状态 IR_IsOpen() 4
3.4　打开设备 IR_OpenDevice() 5
3.5　关闭设备 IR_Close() 5
3.6　发送串口命令 IR_SendSeriaPortCmd() 5
3.7　获取相机分辨率 IR_GetDigitArrayPara() 5
3.8　获取图像 IR_GetNewArray() 5
3.9　获取图像内容 IR_GetDigitArray() 5
3.10　获取显示图像内容 IR_DigitArrayToBmp() 5
3.11　保存原始图像内容 IR_SaveCurrImage() 5
3.12　打开图像 IR_RawImageOpen() 6
3.13　保存显示图像内容 IR_SaveCurrImageAsBmp() 6
3.14　查询录像状态 IR_ IsRecordingMovie() 6
3.15　启动录像 IR_StartRecordMovie() 6
3.16　停止录像 IR_StopRecordMovie() 6
3.17　打开录像视频 IR_MovieFileOpen() 6
3.18　设置播放模式 IR_SetCyclePlay() 6
3.19　查询播放模式 IR_IsCyclePlay() 6
3.20　暂停播放 IR_MoviePause() 7
3.21　查询播放状态 IR_MovieIsPause() 7
3.22　向前播放 IR_MovieForward() 7
3.23　向后播放 IR_MovieBackward() 7
3.24　获取播放位置 IR_MovieGetPosi() 7
3.25　获取总帧数 IR_MovieGetTotalFrm() 7
3.26　设置亮度对比度类型 IR_SetBrihgtContrastType() 7
3.27　获取亮度对比度类型 IR_GetBrihgtContrastType() 8
3.28　设置对比度 IR_SetCurrContrast() 8
3.29　获取对比度 IR_GetCurrContrast() 8

3.30 设置亮度 IR_SetCurrBright() 8
3.31 获取亮度 IR_GetCurrBright() 8
3.32 获取传感器温度 IR_GetVTemp() 8
3.33 转换传感器温度 IR_GH640V1_ADtoT() 8
3.34 获取相机参数 IR_GetCameraConfig() 8
3.35 获取图像序号 IR_GetFrameID() 9
3.36 获取相机帧率 IR_GETDeviceFPS() 9
3.37 获取积分时间 IR_GetCurrIntTime() 9
3.38 获取增益 IR_GetCurrGain() 9
3.39 获取分频因子 IR_GetCurrFrameDiv() 9

4 命令协议说明 9

4.1 数据格式 9
4.2 常用串口命令集 9

5 例程（C++） 10

5.1 连接设备 10
5.2 打快门 11
5.3 配置积分时间 11
5.4 配置增益 12
5.5 写入参数 12
5.6 场景校正 12
5.7 触发 13
5.8 帧频修改 13
5.9 温控设置目标温度 14
5.10 启动温控 14
5.11 关闭温控 14
5.12 切换图像 15

6 二次开发流程 15

7 常见问题 15

附录 16

修订记录

日期	修订版本	修改章节	修改描述	作者
20201116	首次编写V1.0	全文	编写手册所有内容	Ghopto工程师
20201120	V1.0	4.2, 5.9-5.11	添加相机温控指令和配置方法	Ghopto工程师
20210119	V1.0	1, 6	统一32位和64位开发说明	Ghopto工程师
2021119	V1.0	4.1,4.2	增加常用串口命令集	Ghopto工程师

1 概述

本文档描述国惠短波相机在 Windows 32 位和 64 位系统下的 SDK，用于指导用户进行二次开发。二者区别为：32 位设备句柄类型为 int，64 位设备句柄类型为 GHDEV_HND。以下内容以 32 位为例。

2 SDK 变量类型

本节介绍相机 SDK 中的相关变量。

2.1 一些自定义变量

```
typedef  unsigned int  UINT32;
typedef  int  INT32;
typedef  float  FLOAT32;
typedef  double  FLOAT64;
typedef  unsigned long  DWORD;
typedef  short  INT16;
typedef  signed char  INT8;
typedef  short  RAW_TYPE;
```

2.2 设备枚举类型

```
typedef enum
{
    EF_NULL             = 0x00000000,
    EF_Network          = 0x00000001,
    EF_Serial           = 0x00000002,
    EF_CameraLink       = 0x00000004,
    EF_GigEVision       = 0x00000008,
    EF_CoaXPress        = 0x00000010,
    EF_USB              = 0x00000020,
    EF_EnableAll        = 0x0000FFFF,
    EF_UseCached        = 0x01000000,
    EF_ReleaseCache     = 0x02000000,
} EnumerationFlag;
```

EF_USB：USB 接口相机变量；
EF_GigEVision：网口相机变量；
EF_CameraLink：CL 接口相机变量；
注：其他变量为预留参数，无物理意义。

2.3 相机设备信息结构

```
typedef struct {
    char name[16];
    char transport[16];
    char url[32];
    char address[16];
    char interfaceName[16];
    char serial_num[16];
    unsigned int pid;
    unsigned short width;
    unsigned short height;
} DeviceInformation;
```

2.4 获取相机参数结构体

```
typedef struct tagCameraConfig
{
    char detector_type[16];
    char detector_serial_num[16];
    char device_model[16];
    char device_comm_if_type0[16];
    char device_comm_if_type1[16];

    unsigned int integrationTimeMin;
    unsigned int integrationTimeMax;
    unsigned int integrationTime;

    unsigned short VSK;
    unsigned short GFID;
    unsigned short VBUS;
    unsigned short GSK;

    short autoExposureMode;

    short tecMode;
    short tecObject;
    unsigned short pidP;
    unsigned short pidI;
    unsigned short pidD;

    unsigned short frameFrequencyDivider;

    unsigned short sensorGain;
    unsigned short sensorGainCount;
    unsigned short sensorGainArray[4];
    unsigned short sensorOutputMode;
    unsigned short sensorOutputModeCount;
    unsigned short sensorOutputModeArray[4];
    unsigned short sensorReadOrder;
    unsigned short sensorChannelRevert;
    unsigned short sensorTestPattern;
    unsigned short sensorBias;
    unsigned short sensorCDSMode;
    unsigned short sensorReference;

    int  sensorPhaseObject;
    unsigned char sensorTrigMode;
    short   col_arith_type;
    short   col_add[4];

} CameraConfig;
```

integrationTimeMin：相机最小积分时间；
integrationTimeMax：相机最大积分时间；
integrationTime：相机当前积分时间；
frameFrequencyDivider：相机帧频分频因子；
sensorGain：相机传感器增益；
sensorTrigMode：相机触发模式；
注：以上为相机主要参数，其他未介绍参数为相机制造商内部调试使用。

2.5 直方图信息结构体

```
#define  HISTOGRAM_BIN_CNT    512
typedef struct  _HISTOGRAM_INFO_
{
    long   LEN;

    RAW_TYPE  MAX;
    RAW_TYPE  MIN;
    RAW_TYPE  CUT_MIN;
    RAW_TYPE  CUT_MAX;
    int TOP_CUT ;
    int BOT_CUT ;

    long BIN[HISTOGRAM_BIN_CNT];
    char bUseAllData;
}IR_16BIT_HISTOGRAM_INFO, *PIR_16BIT_HISTOGRAM_INFO;
```

MAX：图像数据最大值；

MIN：图像数据最小值；

CUT_MIN：直方图映射时使用的最小图像值；

CUT_MAX: 直方图映射时使用的最大图像值；

TOP_CUT：直方图统计时被扣除的像素（大于 CUT_MAX 值）个数所占总像素的千分数；

BOT_CUT：直方图统计时被扣除的像素（小于 CUT_MIN 值）个数所占总像素的千分数；

BIN：直方图采用的分组（箱子）个数；

bUseAllData：使用映射数据的类型，1-使用被扣除后的数据；0-不使用被扣除的数据。

3 SDK 主要函数(Api.h)

3.1 初始化 IR_Init()

void IR_Init()

功能：系统初始化，在使用 SDK 函数之前必须调用一次

参数：无

返回：无

3.2 枚举设备 IR_EnumerateDevices()

int IR_EnumerateDevices(DeviceInformation* deviceInfomation, unsigned int* deviceCount, EnumerationFlag flags)

功能：枚举设备

参数：deviceInfomation：设备信息指针；deviceCount：获得的设备个数指针；flags：对应设备的标志。

返回：0：无意义。

3.3 查询设备状态 IR_IsOpen()

BOOL IR_IsOpen(int handle)

功能：查询设备是否已经打开

参数：handle：被查询的设备句柄。

返回：1：打开；FALSE：未打开。

3.4 打开设备 IR_OpenDevice()

int IR_OpenDevice(int handle, char *name)

功能：打开设备；当没有设备句柄时可以获取设备句柄

参数：handle：设备句柄（当没有设备句柄时填 Null）；name：设备名。

返回：＞0：成功；＜0：失败。

3.5 关闭设备 IR_Close()

BOOL IR_Close(int handle)

功能：关闭相机

参数：handle：设备句柄。

返回：1：成功；FALSE：0。

3.6 发送串口命令 IR_SendSeriaPortCmd()

BOOL IR_SendSeriaPortCmd(int handle, unsigned char *buff, int len)

功能：向机芯发送通用命令，命令格式参见命令协议说明部分

参数：handle：设备句柄；buff：需要发送的命令数据的存储区，存储区由调用者分配；len: 需要发送命令数据的字节数。

返回：1：成功；0：失败。

3.7 获取相机分辨率 IR_GetDigitArrayPara()

BOOL IR_GetDigitArrayPara(int handle, int *iWidth, int *iHeight)

功能：获取当前设备的图像的分辨率

参数：handle：设备句柄；iWidth：返回的设备图像宽度；iHeight：返回的设备图像高度。

返回：1：成功；0：失败。

3.8 获取图像 IR_GetNewArray()

BOOL IR_GetNewArray(int handle)

功能：更新实时或文件视频的图像，即获取新的一帧图像

参数：handle：设备句柄。

返回：1：成功；0：失败。

3.9 获取图像内容 IR_GetDigitArray()

BOOL IR_GetDigitArray(int handle, void * const pRawShortData)

功能：获取当前设备的图像数据，使用该函数前需先调用 IR_GetNewArray (int handle)

参数：handle：设备句柄；pRawShortData：16-bit 原始图像数据，需分配存储空间。

返回：1：成功；FALSE：0。

3.10 获取显示图像内容 IR_DigitArrayToBmp()

BOOL IR_DigitArrayToBmp(int handle, const short* const pRawShortData, BYTE* const pBmpData, int isRGB, IR_16BIT_HISTOGRAM_INFO* pHistGram)

功能：采用设备自定义信息，转换原始 16-bit 数据为可显示的 8-bit 灰度数据

参数： handle：设备句柄；pRawShortData：16-bit 原始图像数据；pBmpData：8-bit 灰度图像数据；isRGB：RGB 颜色使能，0 不使用（短波红外相机），1 使用；*pHistGram：设备自定义直方图信息。

返回：1：成功；0：失败。

3.11 保存原始图像内容 IR_SaveCurrImage()

BOOL IR_SaveCurrImage(int handle, char * lpszPathName)

功能：抓拍当前帧图像，保存为*.bmp 图像（8bit 灰度数据和 16bit 原始数据）和*.raw 数据(16bit 原始数据）

参数：handle：设备句柄；lpszPathName：文件名，包括路径和文件名称。

返回：1：成功；0：失败。

3.12 打开图像 IR_RawImageOpen()

int IR_RawImageOpen(char * pszFileName)

功能：打开相机自定义保存的*.bmp 图像和*.raw 数据文件（IR_SaveCurrImage()函数保存的）

参数：pszFileName：文件名，包括路径和文件名称。

返回：＞0：成功；＜0：失败。

3.13 保存显示图像内容 IR_SaveCurrImageAsBmp()

BOOL IR_SaveCurrImageAsBmp(int handle, char * lpszPathName)

功能：抓拍当前帧图像，保存为*.bmp 图像（8bit 灰度数据）

参数：handle：设备句柄；lpszPathName：文件名，包括路径和文件名称。

返回：1：成功；0：失败。

3.14 查询录像状态 IR_ IsRecordingMovie()

BOOL IR_ IsRecordingMovie(int handle)

功能：查询是否正在录像

参数：handle：设备句柄。

返回：1：正在录像；0：未录像。

3.15 启动录像 IR_StartRecordMovie()

DWORD IR_StartRecordMovie(int handle, char * szScopeFile)

功能：启动录像，文件类型（后缀）为 irm

参数：handle：设备句柄；szScopeFile：文件名（*.irm），包括路径和文件名称。

返回：0。

3.16 停止录像 IR_StopRecordMovie()

DWORD IR_StopRecordMovie(int handle)

功能：停止录像

参数：handle：设备句柄。

返回：0。

3.17 打开录像视频 IR_MovieFileOpen()

int IR_MovieFileOpen(char * FileName)

功能：打开相机自定义保存的*.irm 录像视频文件

参数：pszFileName：文件名，包括路径和文件名称。

返回：＞0：成功；＜0：失败。

3.18 设置播放模式 IR_SetCyclePlay()

BOOL IR_SetCyclePlay(int handle, BOOL set)

功能：设置录像视频播放模式，循环播放还是单次播放

参数：handle：设备句柄；set：设置循环播放。

返回：1：成功；0：失败。

3.19 查询播放模式 IR_IsCyclePlay()

BOOL IR_IsCyclePlay(int handle)

功能：查询录像视频是否处于循环播放模式

参数：handle：设备句柄。

返回：1：成功；0：失败。

3.20 暂停播放 IR_MoviePause()

BOOL IR_ MoviePause (int handle, BOOL bPause)

功能：暂停录像视频播放

参数：handle：设备句柄；bPause：暂停播放-true；恢复视频播放-false。

返回：1：成功；0：失败。

3.21 查询播放状态 IR_MovieIsPause()

BOOL IR_MovieIsPause(int handle)

功能：查询正在播放的录像视频是否是暂停状态

参数：handle：设备句柄。

返回：1：录像视频播放暂停；0：录像视频播放未暂停。

3.22 向前播放 IR_MovieForward()

BOOL IR_MovieForward (int handle, long nFrame)

功能：控制录像视频向前播放

参数：handle：设备句柄，nFrame：设置向前播放的帧数。

返回：1：成功；0：失败。

3.23 向后播放 IR_MovieBackward()

BOOL IR_MovieBackward (int handle, long nFrame)

功能：控制录像视频向后播放

参数：handle：设备句柄，nFrame：设置向后播放的帧数。

返回：1：成功；0：失败。

3.24 获取播放位置 IR_MovieGetPosi()

int IR_MovieGetPosi (int handle)

功能：获取录像视频当前播放位置

参数：handle：设备句柄。

返回：录像视频播放位置帧序号。

3.25 获取总帧数 IR_MovieGetTotalFrm()

int IR_ MovieGetTotalFrm(int handle)

功能：获取录像视频的总帧数

参数：handle：设备句柄。

返回：录像视频总帧数。

3.26 设置亮度对比度类型 IR_SetBrihgtContrastType()

BOOL IR_SetBrihgtContrastType(int handle, int type)

功能：设置图像亮度对比度类型

参数：handle：设备句柄；type：类型 0 自动，1 手动。

返回：1：成功；0：失败。

3.27 获取亮度对比度类型 IR_GetBrihgtContrastType()

int IR_GetBrihgtContrastType(int handle)

功能：获取图像亮度对比度类型

参数：handle：设备句柄。

返回：0：自动；1：手动。

3.28 设置对比度 IR_SetCurrContrast()

BOOL IR_SetCurrContrast(int handle, int contrast)

功能：设置对比度

参数：handle：设备句柄；contrast：对比度值 0~99。

返回：1：成功；0：失败。

3.29 获取对比度 IR_GetCurrContrast()

BOOL IR_GetCurrContrast(int handle)

功能：获取图像对比度

参数：handle：设备句柄。

返回：对比度值 0~99。

3.30 设置亮度 IR_SetCurrBright()

BOOL IR_SetCurrBright(int handle, int bright)

功能：设置图像亮度

参数：handle：设备句柄；bright：亮度值 0~99。

返回：1：成功；0：失败。

3.31 获取亮度 IR_GetCurrBright()

BOOL IR_GetCurrBright(int handle)

功能：获取图像亮度

参数：handle：设备句柄。

返回：亮度值 0~99。

3.32 获取传感器温度 IR_GetVTemp()

WORD IR_GetVTemp(int handle)

功能：获取相机中传感器温度的采样值 AD（userdef.h 中函数，该头文件被 api.h 调用）

参数：handle：设备句柄。

返回：温度采样值 AD。

3.33 转换传感器温度 IR_GH640V1_ADtoT()

float IR_GH640V1_ADtoT(float ad)

功能：转换传感器温度的采样值 AD 为摄氏温度值，ad 通过 IR_GetVTemp()获得

参数：ad：传感器温度的采样值 AD。

返回：摄氏温度值（℃）。

3.34 获取相机参数 IR_GetCameraConfig()

BOOL IR_GetCameraConfig(int handle, CameraConfig *config)

功能：获取相机当前配置参数

参数：handle：设备句柄；config：参见 2.4。

返回：1：成功；0：失败。

3.35 获取图像序号 IR_GetFrameID()

unsigned int IR_GetFrameID(int handle)

功能：获取设备图像帧序号

参数：handle：设备句柄。

返回：设备的帧频序号。

3.36 获取相机帧率 IR_GETDeviceFPS()

float IR_GETDeviceFPS(int handle)

功能：获取设备帧频

参数：handle：设备句柄。

返回：设备帧频值。

3.37 获取积分时间 IR_GetCurrIntTime()

usingned int IR_GetCurrIntTime(int handle)

功能：获取相机当前积分时间，即曝光时间

参数：handle：设备句柄。

返回：相机当前积分时间。

3.38 获取增益 IR_GetCurrGain()

usingned char IR_GetCurrGain(int handle)

功能：获取相机当前增益类型, 0-高增益，1-中增益，2-低增益

参数：handle：设备句柄。

返回：相机当前增益类型。

3.39 获取分频因子 IR_GetCurrFrameDiv()

unsigned short IR_GetCurrFrameDiv(int handle)

功能：获取相机当前帧频分频因子

参数：handle：设备句柄。

返回：相机当前帧频分频因子。

4 命令协议说明

本节介绍相机自定义串行数据。

4.1 数据格式

首部	地址	数据长度	命令号	数据	校验和
3 字节	1 字节	1 字节	2 字节	8 字节	1 字节

1.首部：总是 0xFF 0xFF 0xAA

2.地址范围是 0x00~0xff （十进制是 0 ~ 255），其中 0xff 为广播地址

3.数据长度：1 字节（0x00）

4.命令号：范围为 0x0001~0xffff，命令号的低字节在前（小端字节），用于区分不同的命令。

5.数据：共 8 个字节，数据的低字节在前。

6.校验和：占一个字节，将前面除首部外的所有字节（地址、长度、命令号和数据）异或计算结果。

4.2 常用串口命令集

表中命令首部、地址位与数据长度都是不变的，不同命令拥有不同的命令号。其以 0x**标注的为可变字节，数据位除可变字节以外都以 0x00 填充。最后将前面除首部外的其余字节（地址、长度、命令号和数据）异或计

算得到校验位。

命令类别	命令号		数据	校验和	示例变量名称
打快门	0x09, 0x00		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0x5c	SHUTTER
配置积分时间	0x22, 0x00		0x**, 0x**, 0x**, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0x**	INT_TIME
配置增益	0x19, 0x00	高增益	0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0x4c	GAIN
		中增益	0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0x4d	
		低增益	0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0x4e	
保存所有参数	0x4F, 0x00		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0x1a	SAVE_ALL
场景矫正	0x0A, 0x00		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0x5f	BAK_NUC
触发	0x1a, 0x00		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0x4f	TRIGGER
分频参数	0x21, 0x00		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0x74	FRAME_RATE
设置温度	0x0E, 0x00		0x**, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0x**	VTEMP
图像类型	0x11, 0x00	矫正图像	0x12, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0xed	IMAGE_MODE
		原始图像	0x11, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0xee	
温控开关	0x37, 0x00	打开温控	0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0x62	TEC_MODE
		关闭温控	0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0x63	

5 例程（C++）

```
int m_DeviceHandle =0; //32位设备句柄，64位定义为GHDEV_HND m_DeviceHandle;
```

5.1 连接设备

```
int main()
{
    IR_Init();
    unsigned int m_deviceCount=0;
    unsigned int m_accessMode   = EF_USB;   // EF_USB->usb设备; EF_GigEVision->网口设备
    IR_EnumerateDevices(NULL, &m_deviceCount, (EnumerationFlag)m_accessMode);
    DeviceInformation *m_deviceList=NULL;
```

```
    if(m_deviceCount>0)
    {
        m_deviceList = new DeviceInformation[m_deviceCount];
        IR_EnumerateDevices(m_deviceList, &m_deviceCount, EF_USB);
        // 默认打开第一个设备
        m_DeviceHandle = IR_OpenDevice(NULL,m_deviceList[0].url);
    }
    //printf("Hello World!\n");
    return 0;
}
```

5.2 打快门

```
void OnButtonShutter()
{
    if(m_DeviceHandle)
    {
        SHUTTER[7] = 0x00;
        SHUTTER[15] = 0x00;
        for (int i=3; i<=14; i++)
        {
            SHUTTER[15] = SHUTTER[15] ^ SHUTTER[i];
        }
        IR_SendSeriaPortCmd(m_DeviceHandle, SHUTTER, sizeof(SHUTTER));
    }
}
```

5.3 配置积分时间

```
void OnSetIntTime(int m_SensorIntTime)
{
    if(m_DeviceHandle)
    {
        INT_TIME[7] = (m_SensorIntTime & 0x00ff);
        INT_TIME[8] = (m_SensorIntTime & 0xff00) >> 8;
        INT_TIME[9] = (m_SensorIntTime & 0xff0000) >> 16;
        INT_TIME[15] = 0x00;
        for (int i=3; i<=14; i++)
        {
            INT_TIME[15] =INT_TIME[15] ^ INT_TIME[i];
        }
        IR_SendSeriaPortCmd(m_DeviceHandle, INT_TIME, sizeof(INT_TIME));
```

```
        }
    }
```

5.4 配置增益

```
void OnSetGain(int m_SensorGain)
{
    if(m_DeviceHandle)
    {
        GAIN[7] = m_ SensorGain ; //0->高增益;1->中增益;2->低增益
        GAIN[15] = 0x00;
        for (int i=3; i<=14; i++)
        {
            GAIN[15] = GAIN[15] ^ GAIN[i];
        }
        IR_SendSeriaPortCmd(m_DeviceHandle, GAIN, sizeof(GAIN));
    }
}
```

5.5 写入参数

相机参数在线调整后，只在本次使用中生效，参数不会存入相机，掉电后失效，如果想把参数存入相机且掉电不丢失，需发送保存参数串行命令（保存过程耗时最少 40s，建议等待 50s 以上）

```
void OnSaveAll()
{
    if(m_DeviceHandle)
    {
        SAVE_ALL[15] = 0x00;
        for (int i = 3; i <= 14; i++)
        {
            SAVE_ALL[15] = SAVE_ALL[15] ^ SAVE_ALL[i];
        }
        IR_SendSeriaPortCmd(m_DeviceHandle, SAVE_ALL, sizeof(SAVE_ALL));
    }
}
```

5.6 场景校正

```
void OnBackgroundNuc()
{
    if(m_DeviceHandle)
    {
        BAK_NUC[15] = 0x00;
        for (int i = 3; i <= 14; i++)
```

```
        {
            BAK_NUC[15] = BAK_NUC[15] ^ BAK_NUC[i];
        }
        IR_SendSeriaPortCmd(m_DeviceHandle, BAK_NUC, sizeof(BAK_NUC));
    }
}
```

5.7 触发

```
void OnTriggerMode(int m_triggerMode)
{   // m_triggerMode = 0或2或3
    //0：内部触发；2：外部上升下降沿触发；3：外部上升沿触发。
    if(m_DeviceHandle)
    {
        TRIGGER[9] = m_triggerMode;
        TRIGGER[15] = 0x00;
        for (int i = 3; i <= 14; i++)
        {
            TRIGGER_GEN[15] =TRIGGER_GEN[15] ^ TRIGGER[i];
        }
        IR_SendSeriaPortCmd(m_DeviceHandle, TRIGGER, sizeof(TRIGGER));
    }
}
```

5.8 帧频修改

```
void OnSetFrameFrequency(int m_frameDiv)
{   // m_frameDiv =1,2，……，32  最大可到32分频
    if(m_DeviceHandle)
    {
        int frameRate = powf(2, m_frameDiv);
        FRAME_RATE[7] = (frameRate & 0x00ff);
        FRAME_RATE[8] = (frameRate & 0xff00)>>8;
        FRAME_RATE[15] = 0x00;
        for (int i = 3; i <= 14; i++)
        {
            FRAME_RATE[15] = FRAME_RATE[15] ^ FRAME_RATE[i];
        }
        IR_SendSeriaPortCmd(m_DeviceHandle, FRAME_RATE, sizeof(FRAME_RATE));
    }
}
```

5.9 温控设置目标温度

```
void OnSetVtemp(int m_VTEMP)
{    // m_VTEMP：温度值，单位为摄氏度
    if(m_DeviceHandle)
    {
        VTEMP[7] = (m_VTEMP & 0x00FF);
        VTEMP[8] = (m_VTEMP & 0xFF00 )>>8;
        VTEMP[15] = 0x00;
        for (int i = 3; i <= 14; i++)
        {
            VTEMP[15] = VTEMP[15] ^ VTEMP[i];
        }
        IR_SendSeriaPortCmd(m_DeviceHandle, VTEMP, sizeof(VTEMP));
    }
}
```

5.10 启动温控

TEC 控制功能属高级功能，使用需考虑散热能力（如果无外部散热装置，国惠相机（整机）温控能力最多可使相机相对环境温度下降 15℃，且不可长时间启动）。启动前请先设置目标温度。

```
void OnTecModeEnable()
{
    if(m_DeviceHandle)
    {
        TEC_MODE[7] = 1;
        TEC_MODE[15] = 0x00;
        for (int i = 3; i <= 14; i++)
        {
            TEC_MODE[15] = TEC_MODE[15] ^ TEC_MODE[i];
        }
        IR_SendSeriaPortCmd(m_DeviceHandle, TEC_MODE, sizeof(TEC_MODE));
    }
}
```

5.11 关闭温控

```
void OnTecModeDisable()
{
    if(m_DeviceHandle)
    {
        TEC_MODE[7] = 0;
        TEC_MODE[15] = 0x00;
```

```
            for (int i = 3; i <= 14; i++)
            {
                TEC_MODE[15] = TEC_MODE[15] ^ TEC_MODE[i];
            }
            IR_SendSeriaPortCmd(m_DeviceHandle, TEC_MODE, sizeof(TEC_MODE));
        }
    }
```

5.12 切换图像

```
void OnChangeImageType(int m_imgType)
{    // m_imgType=0x11切换图像为原始数据
     // m_imgType=0x12切换图像为校正数据
     if(m_DeviceHandle)
     {
          IMAGE_MODE[7] = m_imgType;
          IMAGE_MODE [15] = 0x00;
          for (int i = 3; i <= 14; i++)
          {
               IMAGE_MODE [15] = IMAGE_MODE [15] ^ IMAGE_MODE [i];
          }
          IR_SendSeriaPortCmd(m_DeviceHandle, IMAGE_MODE, sizeof(IMAGE_MODE));
     }
```

6 二次开发流程

1.安装WinPcap_4_1_3.exe微软库（32位），或Win10Pcap-v10.2-5002.msi微软库（64位）；

2.复制GHOPTO库文件夹32位Inc、Lib和Bin或64位IncX64、LibX64和BinX64到目标工程中，头文件（api.h）和库文件（IRSDK.lib和IRSDKD.lib）32位分别在Inc和Lib文件夹，64位分别在IncX64和LibX64文件夹，dll文件（IRSDK.dll和IRSDKD.dll）分别在Bin和BinX64文件夹；

3. IRSDK中包含两个版本Release和Debug，开发Release版本时使用IRSDK.dll、IRSDK.lib和api.h文件；开发Debug版本时使用IRSDKD.dll、IRSDKD.lib和api.h文件；

4.根据本文档描述进行相关开发。

备注：以上所需文件由GHOPTO提供，如有问题请联系GHOPTO。

7 常见问题

问题：采用vs开发win32控制台应用程序，调用SDK编译时，报错api.h中“缺少类型说明符”、“ 位置重写说明符”等问题。

解决方法：加入头文件#include<windows.h>

问题：使用QT开发时，添加动态库后sdk函数依然报错。

解决方法：添加头文件#include<windows.h>

附录

相关开发问题欢迎在网上 https://blog.csdn.net/wwending 进行互动交流。