CK 工业相机开发手册

V2.0 版本

简介

本文档提供给使用我公司工业相机进行二次开发的用户使用，针对系统开发中使用到的动态链接库函数进行详细说明，并提供了快速程序设计指导，以便在短时间内能够将相机集成到用户的系统中。

声明

目录

1 概述

1.1 文件结构

在二次开发中，需要直接使用到的库文件位于安装目录的 SDK 文件夹中，分为 32 位和 64 位两个开发包。其中 64 位的 SDK 开发文件，位于 SDK/X64 文件夹中，相对于 32 位的 SDK 文件，64 位的 SDK 文件名均以_X64 结尾。

SDK 目录下的 CKSDK.DLL 是相机的 SDK 动态链接库，对外提供相机所有的接口函数，VC/C++的例程都引用了该库文件。

1.2 开发环境

SDK 是标准的 C 语言接口动态链接库 ，我们提供了基于 DOME VS2015 的例程。

2 快速开发指南

2.1 相机操作流程概述

我们建议您按照如下的流程操作相机(其中有一些步骤是可选的，已经标明)：

一、载入 SDK 的动态链接库档 CKSDK.DLL。您可以使用动态或者静态加载两种方式。如果您使用 C/C++进行开发，在工程引用

CKCameraInterface.h 头文件（位于安装目录的SDK/DEMO/VC++/include 中),然后就可以直接在工程中引用 SDK 中的接口函数了,但是 CKSDK.DLL必须和您的应用程序放在同一目录下或者是系统的 system32 目录下，放置于其他目录时，必须设定系统的环境变量(PATH)。

二、枚举设备。调用 CameraEnumerateDevice 函数枚举设备，获得当前连接到 PC 上的相机的总个数，调用CameraGetEnumIndexInfo函数获得当前标号包括设备名昵称(可自己修改)、版本号、唯一序列号、相机型号等信息。

三、初始化设备。根据第二步中获得的相机设备枚举信息，调用CameraInit 函数初始化设定的相机，得到相机的句柄。如果需要同时打开多个相机,则利用多个相机的设备名多次调用 CameraInit 来获得多个相机的句柄，后续对相机的操作，都需要此时获得的相机句柄来指定操作的相机对象。

四、让 SDK 进入图像采集模式。调用 CameraPlay 函数，让相机进入工作模式，SDK 开始接收来自相机的图像。

五、 抓取图像。 SDK 提供了两种获得图像数据的方式，这两种方式的效率都是一样的，底层都使用了零拷贝机制来提高效率，您可以根据您的开发习惯来选择其中一种。

1.主动调用 CameraGetRawImageBuffer来获得一帧图像句柄。同时，该函数可以设定超时时间，在指定的时间内没有获得到图像(线程会被挂起)，则返回回超时。通过该句柄调用CameraGetImageInfo

该函数会获得一个 SDK 内被部用来接收图像数据的缓冲区地址，以及帧头信息。

2. 在第三步中，初始化相机以后，调用 CameraSetCallbackFunction 来指定一个回调函数。这种方式是被动的，在 SDK 内部接收到有效的图像数据帧后，才会调用您指定的回调函数来传递收到的图像数据帧。

六、处理图像。上一步获得的图像帧，是相机输出的原始格式，我公司大多数型号相机，原始输出都是 Bayer 格式或者 YUV 格式，这些格式信息会被自动添加到帧头信息中，调用 CameraGetOutImageBuffer 来获得图像处理的效果,如颜色增益调整、白平衡校正、饱和度等，并将 YUV 戒者 Bayer 格式的原始数据转换为 24BIT 或者 32BIT 的位图格式。

七、将图像保存或者显示图像(如果您的开发中，不需要将图像保存成文件或者进行显示，这一步可以略过)。

如果您需要保存图像到文件中，在第六步后，调用 CameraSaveImage 函数来保存图片，SDK 支持 BMP 和原始数据两种方式。如果要保存原始数据，您应该在第五步以后就调用 CameraSaveImage 函数；如果保存成 BMP 格式，您应该在第六步后调用 CameraSaveImage 函数；

如果您需要显示图像，有以下两个方式：

1.自己根据开发环境来实现图像显示，例如利用 OpenGL、DirectDraw、 Windows GDI 等方式来实现图像的显示。

2.利用我们的 SDK 里封装好的显示接口来显示图像。在第四步中初始化相机后，调用 CameraDisplayInit 函数来初始化显示接口，该函数需要传入显示控件的句柄(HWND 类型)。

七、在退出程序前关闭相机。在关闭相机时，调用 CameraUnInit 函数。

2.2 开发例程

2.2.1 主动获取图像数据的例程

```
if (CameraEnumerateDevice(&mCameraNums) != CAMERA_STATUS_SUCCESS ||
    mCameraNums <= 0)ret = CAMERA_STATUS_FAILED;
    if (ret != CAMERA_STATUS_SUCCESS) return FALSE;
    tDevInfo DevInfo;
    ////第一步：枚举
    if (ret == CAMERA_STATUS_SUCCESS)CameraGetEnumIndexInfo(mCameraNums - 1,
    &DevInfo);
    ////第二步：初始化相机
    if (ret == CAMERA_STATUS_SUCCESS && CameraInitEx(&mCKCamHandle, mCameraNums -
    1,-1,-1) != CAMERA_STATUS_SUCCESS)ret = CAMERA_STATUS_FAILED;

    if (ret == CAMERA_STATUS_SUCCESS)
    {
        /////第三步:SDK 内部显示初始化（如果用户不需要 SDK 内部提供显示功能的情况下，
        此步奏可以略去）
        ret = CameraDisplayInit(mCKCamHandle, hwnd);
        ////第四步：采用主动方式获取图像数据
        m_hDispThread = (HANDLE)_beginthreadex(NULL, 0, &uiDisplayThread, this, 0,
        &m_threadID);
        ASSERT(m_hDispThread);
        m_bExit = FALSE;
        ////第五步：相机进入运行状态
        CameraPlay(mCKCamHandle);
        m_bPause = FALSE;
        GetDlgItem(IDC_BTN_CAM_PLAY)->SetWindowText(_T("暂停"));
        isOpen = TRUE;
        GetDlgItem(IDC_BTN_CAMERA_OPEN)->SetWindowText(_T("关闭"));
    }
```

```
UINT uiDisplayThread()
{
    BYTE*            pbyBuffer;
    stImageInfo ImageInfo;
    int status = CAMERA_STATUS_SUCCESS;
    FrameStatistic fstatistic;
    while (!m_bExit)
    {
        FrameTimeCur = GetTickCount();
        HANDLE hBuf;
        //获取 RAW 数据
        status = CameraGetRawImageBuffer(mCKCamHandle, &hBuf, 1000);
        if (status == CAMERA_STATUS_SUCCESS)
        {
            //获得图像缓冲区地址
            pbyBuffer = CameraGetImageInfo(mCKCamHandle, hBuf, &ImageInfo);
            //获得经 ISP 处理的 RGB 数据
            ////////////////////////////////
            if (pRGBFrame == NULL || dRGBBufLen < (ImageInfo.iWidth*
            ImageInfo.iHeight * 4))
            {
                if (pRGBFrame) delete pRGBFrame;

                dRGBBufLen = (ImageInfo.iWidth*ImageInfo.iHeight * 4);
                pRGBFrame = new BYTE[dRGBBufLen];

            }
            CameraGetOutImageBuffer(mCKCamHandle, &ImageInfo, pbyBuffer, pRGBFrame);

            m_iDispFrameNum++;
            memcpy(&m_sFrInfo, &ImageInfo, sizeof(stImageInfo));
            ////////////////////////////////显示
            CameraDisplay(mCKCamHandle, pRGBFrame, &ImageInfo);
            //释放由 CameraGetRawImageBuffer 获得的缓冲区
            CameraReleaseFrameHandle(mCKCamHandle, hBuf);
        }
    }
    _endthreadex(0);
    return 0;
}
```

2.2.2 被动获取图像数据的例程

```
if (CameraEnumerateDevice(&mCameraNums) != CAMERA_STATUS_SUCCESS ||
    mCameraNums <= 0)ret = CAMERA_STATUS_FAILED;
        if (ret != CAMERA_STATUS_SUCCESS) return FALSE;
        tDevInfo DevInfo;
        ////第一步：枚举
        if (ret == CAMERA_STATUS_SUCCESS)CameraGetEnumIndexInfo(mCameraNums - 1,
&DevInfo);
        ////第二步：初始化相机
        if (ret == CAMERA_STATUS_SUCCESS && CameraInit(&mCKCamHandle, mCameraNums - 1) !=
CAMERA_STATUS_SUCCESS)ret = CAMERA_STATUS_FAILED;

        if (ret == CAMERA_STATUS_SUCCESS)
        {
            /////第三步：SDK 内部显示初始化（如果用户不需要 SDK 内部提供显示功能的情况下，
此步奏可以略去）
            ret = CameraDisplayInit(mCKCamHandle, hwnd);
            ////第四步：采用被动方式获取图像数据,设置回调函数
            CameraSetCallbackFunction(mCKCamHandle, GrabImageCallback, (PVOID)this,
NULL);
            ////第五步：相机进入运行状态
            CameraPlay(mCKCamHandle);
        }
void GrabImageCallbackHANDLE handle, BYTE *pFrameBuffer, stImageInfo* pFrameInfo)
{
    stImageInfo ImageInfo;
    memcpy(&ImageInfo, pFrameInfo, sizeof(stImageInfo));
    ////////////////////////////////
    if (pRGBFrame == NULL || dRGBBufLen < (ImageInfo.iWidth*ImageInfo.iHeight * 4))
    {
        if (pRGBFrame) delete pRGBFrame;

        dRGBBufLen = (ImageInfo.iWidth*ImageInfo.iHeight * 4);
        pRGBFrame = new BYTE[dRGBBufLen];

    }
    ////////////////////////////////
    CameraGetOutImageBuffer(mCKCamHandle, &ImageInfo, pFrameBuffer, pRGBFrame);

    m_iDispFrameNum++;
    memcpy(&m_sFrInfo, &ImageInfo, sizeof(stImageInfo));
    ////////////////////////////////显示
    CameraDisplay(mCKCamHandle, pRGBFrame, &ImageInfo);
```

```
}
```

3 SDK 数据类型定义

3.1 结构体定义

◆ tDevInfo

说明：相机的设备信息

```
typedef struct
{
    char acProductSeries[32];          // 产品系列
    char acProductName[32];                // 产品名称
    char acFriendlyName[32];           // 产品昵称,用户可自定义改昵称,保存在相机内， 用
                                       //于区分多个相机同时使用
    char acLinkName[32];               // 内核符号连接名，内部使用
    char acDriverName[32];             // 驱动名称
    char acDriverVersion[32];          // 驱动版本
    char acSensorType[32];             // sensor 类型
    char acPortType[32];               // 接口类型
    UINT uInstance;                        // 该型号相机在该电脑上的实例索引号，用于区
                                       //分同型号多相机
    char acSn[33];                         // 产品唯一序列号
    WORD VendorID;                         // 厂商 ID
    WORD DeviceID;                         // 设备 ID
    WORD DeviceVersionID;              // 设备版本 ID
    char SymbolicName[128];                // 设备标号，内部使用
    char Name[64];                         // 设备标号，内部使用
} tDevInfo;
```

◆ stImageInfo

说明：帧头信息

```
typedef struct Tag_stImageInfo
{
    UINT     iWidth;                   // 当前图像宽
    UINT     iHeight;                  // 当前图像高
    UINT     TotalBytes;               // 图像数据字节数,Total bytes
    UINT     uiMediaType;              // 图像格式
    double       ExpTime;              // 当前图像的曝光时间，单位为微秒 us
    double      ExpLineTime;           // 当前图像的行曝光时间，单位为微秒 us
    UINT     Gain;                     // 当前图像的增益倍数
}stImageInfo, *PstImageInfo;
```

◆ FrameStatistic

说明：帧率统计信息

```
typedef struct
{
    INT iTotal;                          //当前采集的总帧数（包括错误帧）
    INT iCapture;                       //当前采集的有效帧的数量
    INT iLost;                           //当前丢帧的数量
} FrameStatistic;
```

◆ tExpose

说明：相机曝光功能范围定义

```
typedef struct
{
    UINT  uiTargetMin;      //自动曝光亮度目标最小值
    UINT  uiTargetMax;      //自动曝光亮度目标最大值
    UINT  uiAnalogGainMin;  //模拟增益的最小值，单位为倍数，再扩大 1000 倍
    UINT  uiAnalogGainMax;  //模拟增益的最大值，单位为倍数，再扩大 1000 倍
    UINT  uiExposeTimeMin;  //手动模式下，曝光时间的最小值，单位:行。根据
                            //CameraGetExposureLineTime 可以获得
                        //一行对应的时间(微秒),从而得到整帧的曝光时间
    UINT  uiExposeTimeMax;  //手动模式下，曝光时间的最大值，单位:行
} tExpose;
```

◆ tResolutionRange

说明：相机的分辨率设定范围，用于构件 UI

```
typedef struct
{
    INT iHeightMax;             //图像最大高度
    INT iHeightMin;             //图像最小高度
    INT iWidthMax;              //图像最大宽度
    INT iWidthMin;              //图像最小宽度
    UINT uSkipModeMask;         //SKIP 模式掩码，为 0，表示不支持 SKIP 。bit0 为 1,表示支
                               //持 SKIP 2x2 ;bit1 为 1，表示支持 SKIP 3x3....
    UINTuBinSumModeMask;        //BIN(求和)模式掩码，为 0，表示不支持 BIN 。bit0 为 1,表示
                               //支持 BIN 2x2 ;bit1 为 1，表示支持 BIN 3x3....
    UINT uBinAverageModeMask;   //BIN(求均值)模式掩码，为 0，表示不支持 BIN 。bit0 为 1,
                               //表示支持 BIN 2x2 ;bit1 为 1，表示支持 BIN 3x3....
    UINT uResampleMask;         //硬件重采样的掩码
} tResolutionRange;
```

◆ tSensorCfg

说明：sensor 能力级

```
typedef struct tSensorCapbility_Tag
```

```
{
    tExpose            sExposeDesc;      // 曝光的范围值
    tResolutionRange   sResolutionRange; // 分辨率范围描述
} tSensorCfg;
```

◆ tRgbGainRange

说明：RGB 三通道数字增益的设定范围

```
typedef struct
{
    INT iRGainMin;    //红色增益的最小值
    INT iRGainMax;    //红色增益的最大值
    INT iGGainMin;    //绿色增益的最小值
    INT iGGainMax;    //绿色增益的最大值
    INT iBGainMin;    //蓝色增益的最小值
    INT iBGainMax;    //蓝色增益的最大值
} tRgbGainRange;
```

◆ tSaturationRange

说明：饱和度设定的范围

```
typedef struct
{
    INT iMin;   //最小值
    INT iMax;   //最大值
} tSaturationRange;
```

◆ tSharpnessRange

说明：锐化的设定范围

```
typedef struct
{
    INT iMin;   //最小值
    INT iMax;   //最大值
} tSharpnessRange;
```

◆ tGammaRange

说明：伽马的设定范围

```
typedef struct
{
    INT iMin;       //最小值
    INT iMax;       //最大值
} tGammaRange;
```

◆ tContrastRange

说明：对比度的设定范围

```
typedef struct
```

```
{
    INT iMin;    //最小值
    INT iMax;    //最大值
} tContrastRange;
```

◆ tSdkIspCapacity

说明：ISP 模块的使能信息

```
typedef struct
{
    BOOL bMonoSensor;        //表示该型号相机是否为黑白相机,如果是黑白相机，则颜色相关的
                             //功能都无法调节
    BOOL bWbOnce;            //表示该型号相机是否支持一次白平衡功能
    BOOL bAutoWb;            //表示该型号相机是否支持自动白平衡功能
    BOOL bAutoExposure;      //表示该型号相机是否支持自动曝光功能
    BOOL bManualExposure;    //表示该型号相机是否支持手动曝光功能
    BOOL bAntiFlick;         //表示该型号相机是否支持抗频闪功能
    BOOL bIspGamma;          //表示该型号相机是否支持 Gamma 功能
    BOOL bDeviceIsp;         //表示该型号相机是否支持硬件 ISP 功能
    BOOL bForceUseDeviceIsp;//bDeviceIsp 和 bForceUseDeviceIsp 同时为 TRUE 时，表示强制只
                             //用硬件 ISP，不可取消。
    BOOL bZoomHD;            //相机硬件是否支持图像缩放输出(只能是缩小)。
    BOOL rev[4];
} tSdkIspCapacity;
```

◆ tSdkTrigger

说明：触发模式描述

```
typedef struct
{
    INT   iIndex;            //模式索引号
    char  acDescription[32]; //该模式的描述信息
} tSdkTrigger;
```

◆ tSdkImageResolution

说明：相机的分辨率描述

```
typedef struct
{
    INT     iIndex;             // 索引号，[0,N]表示预设的分辨率(N 为预设分辨率的最大个
                                //数，一般不超过 20),0XFF 表示自定义分辨率(ROI)
    char    acDescription[32];  // 该分辨率的描述信息。仅预设分辨率时该信息有效。自定义
                                //分辨率可忽略该信息
    UINT    uBinSumMode;        // BIN(求和)的模式,范围不能超过 tSdkResolutionRange 中
                                //uBinSumModeMask
    UINT    uBinAverageMode;    // BIN(求均值)的模式,范围不能超过 tSdkResolutionRange 中
                                //uBinAverageModeMask
```

```
    UINT    uSkipMode;          // 是否 SKIP 的尺寸，为 0 表示禁止 SKIP 模式，范围不能超过
                                //tSdkResolutionRange 中 uSkipModeMask
    UINT    uResampleMask;      // 硬件重采样的掩码
    INT     iHOffsetFOV;        // 采集视场相对于 Sensor 最大视场左上角的水平偏移
    INT     iVOffsetFOV;        // 采集视场相对于 Sensor 最大视场左上角的垂直偏移
    INT     iWidthFOV;          // 采集视场的宽度
    INT     iHeightFOV;         // 采集视场的高度
    INT     iWidth;             // 相机最终输出的图像的宽度,偶数,且 4 字节对齐
    INT     iHeight;            // 相机最终输出的图像的高度
    INT     iWidthZoomHd;       // 硬件缩放的宽度,不需要进行此操作的分辨率,
                                //此变量设置为 0.
    INT     iHeightZoomHd;      // 硬件缩放的高度,不需要进行此操作的分辨率,
                                //此变量设置为 0.
    INT     iWidthZoomSw;       // 软件缩放的宽度,不需要进行此操作的分辨率,
                                //此变量设置为 0.
    INT     iHeightZoomSw;      // 软件缩放的高度,不需要进行此操作的分辨率,
                                //此变量设置为 0.
} tSdkImageResolution;
```

◆ tSdkColorTemperatureDes

说明：相机白平衡色温模式描述信息

```
typedef struct
{
    INT  iIndex;            // 模式索引号
    char acDescription[32]; // 描述信息
} tSdkColorTemperatureDes;
```

◆ tSdkPackLength

说明：传输分包大小描述(主要是针对网络相机有效)

```
typedef struct
{
    INT  iIndex;             //分包大小索引号
    char acDescription[32];  //对应的描述信息
    UINT iPackSize;
} tSdkPackLength;
```

◆ tSdkPresetLut

说明：预设的 LUT 表描述

```
typedef struct
{
    INT  iIndex;               //编号
    char acDescription[32];    //描述信息
} tSdkPresetLut;
```

◆ tSdkAeAlgorithm

说明：AE 算法描述

```
typedef struct
{
    INT  iIndex;                //编号
    char acDescription[32];     //描述信息
} tSdkAeAlgorithm;
```

◆ tSdkBayerDecodeAlgorithm

说明：RAW 转 RGB 算法描述

```
typedef struct
{
    INT  iIndex;                //编号
    char acDescription[32];     //描述信息
} tSdkBayerDecodeAlgorithm;
```

◆ tSdkMediaType

说明：相机输出的图像数据格式

```
typedef struct
{
    INT     iIndex;             //格式种类编号
    char    acDescription[32];  //描述信息
    UINT    iMediaType;         //对应的图像格式编码，如 CAMERA_MEDIA_TYPE_BAYGR8，在本
文件中有定义。
} tSdkMediaType;
```

◆ tSdkBayerType

说明：相机输出的图像数据格式

```
typedef struct
{
    INT     iIndex;             //格式种类编号
    char    acDescription[32];  //描述信息
    UINT    iMediaType;         //对应的 sensor 支持的格式编码，如
                               //CAMERA_MEDIA_TYPE_BAYGR8
} tSdkBayerType;
```

◆ tSdkFrameSpeed

说明：相机帧率描述信息

```
typedef struct
{
    INT  iIndex;            // 帧率索引号，一般 0 对应于低速模式，1 对应于普通模式，2 对
                            //应于高速模式
    char acDescription[32]; // 描述信息
} tSdkFrameSpeed;
```

◆ tDeviceCfg

说明：设备能力级

```
typedef struct tDeviceCfg_Tag
{

    tRgbGainRange           sRgbGainRange;          // 图像数字增益范围描述
    tSaturationRange        sSaturationRange;    // 饱和度范围描述
    tGammaRange             sGammaRange;         // 伽马范围描述
    tContrastRange          sContrastRange;          // 对比度范围描述
    tSharpnessRange         sSharpnessRange;     // 锐化范围描述
    tSdkIspCapacity         sIspCapacity;        // ISP 能力描述

    tSdkTrigger             *pTriggerDesc;         // 触发模式
    INT                     iTriggerDesc;          // 触发模式的个数，即
                                                   //pTriggerDesc 数组的大小

    tSdkImageResolution     *pImageSizeDesc;     // 预设分辨率选择
    INT                     iImageSizeDesc;          // 预设分辨率的个数，即
                                                   //pImageSizeDesc 数组的大小

    tSdkColorTemperatureDes *pClrTempDesc;           // 预设色温模式，用于白平
                                                     //衡
    INT                     iClrTempDesc;

    tSdkBayerType           *pBayerTypeDesc;     // 相机 Bayer 格式
    INT                     iBayerTypdeDesc;     // 相机 Bayer 格式的种类个数，
                                                 //即 pBayerTypeDesc 数组的大小。

    tSdkMediaType           *pMediaTypeDesc;     // 相机输出图像格式
    INT                     iMediaTypdeDesc;     // 相机输出图像格式的种类个
                                                 //数，即 pMediaTypeDesc 数组的大小。

    tSdkFrameSpeed          *pFrameSpeedDesc;    // 可调节帧速类型，对应界面上
                                                 //普通 高速 和超级三种速度设置
    INT                     iFrameSpeedDesc;     // 可调节帧速类型的个数，即
                                                 //pFrameSpeedDesc 数组的大小。

    tSdkPackLength          *pPackLenDesc;           // 传输包长度，一般用于网
                                                     //络设备
    INT                     iPackLenDesc;        // 可供选择的传输分包长度的个
                                                 //数，即 pPackLenDesc 数组的大小。

    INT                     iOutputIoCounts;     // 可编程输出 IO 的个数
```

```
    INT                         iInputIoCounts;         // 可编程输入 IO 的个数

    tSdkPresetLut               *pPresetLutDesc;      // 相机预设的 LUT 表
    INT                         iPresetLut;                 // 相机预设的LUT表的个数，
                                                    //即 pPresetLutDesc 数组的大小

    INT                         iUserDataMaxLen;        // 指示该相机中用于保存用户
                                                    //数据区的最大长度。为 0 表示无。
    BOOL                        bParamInDevice;         // 指示该设备是否支持从设备中读
                                                    //写参数组。1 为支持，0 不支持。

    tSdkAeAlgorithm             *pAeAlmSwDesc;              // 软件自动曝光算法描述
    int                         iAeAlmSwDesc;           // 软件自动曝光算法个数

    tSdkAeAlgorithm             *pAeAlmHdDesc;              // 硬件自动曝光算法描述，
                                                    //为 NULL 表示不支持硬件自动曝光
    int                         iAeAlmHdDesc;           // 硬件自动曝光算法个数，为 0
                                                    //表示不支持硬件自动曝光

    tSdkBayerDecodeAlgorithm    *pBayerDecAlmSwDesc;  // 软件 Bayer 转换为 RGB 数据的算法描
                                                    //述
    int                         iBayerDecAlmSwDesc;         // 软件 Bayer 转换为 RGB 数
                                                    //据的算法个数

    tSdkBayerDecodeAlgorithm    *pBayerDecAlmHdDesc;  // 硬件 Bayer 转换为 RGB 数据的算法描
                                                    //述，为 NULL 表示不支持
    int                         iBayerDecAlmHdDesc;         // 硬件 Bayer 转换为 RGB 数
                                                    //据的算法个数，为 0 表示不支持

} tDeviceCfg;
```

◆ tSdkCameraCapbility

说明：相机能力级

```
typedef struct tSdkCameraCapbility_Tag
{
    tDeviceCfg tDeviceCapbility;        //设备能力级
    tSensorCfg tSensorCapbility;        //sensor 能力级

} tSdkCameraCapbility;
```

3.2 宏定义

```
typedef enum
```

```
{
    IMAGEOUT_MODE_512X384 = 0,
    IMAGEOUT_MODE_640X480 = 1,
    IMAGEOUT_MODE_800X600 = 2,
    IMAGEOUT_MODE_1024X768 = 3,
    IMAGEOUT_MODE_1280X720 = 4,
    IMAGEOUT_MODE_1280X960 = 5,
    IMAGEOUT_MODE_1920X1280 = 6,
    IMAGEOUT_MODE_2048X1536 = 7,
    IMAGEOUT_MODE_CUSTOM = 0XFF,
}emResolutionMode;

typedef enum
{
    FRAME_SPEED_LOW = 0,//帧速度 低速
    FRAME_SPEED_MIDDLE,    //帧速度 中速
    FRAME_SPEED_HIGH, //帧速度 高速
}emFrameMode;

enum
{
    FREQ_FLICKER_MODE_NULL = 0,
    FREQ_FLICKER_MODE_50HZ,
    FREQ_FLICKER_MODE_60HZ,
};

enum
{
    FILE_JPG = 1,//JPG
    FILE_BMP = 2,//BMP 24bit
    FILE_RAW = 4,//相机输出的 bayer 格式文件,对于不支持 bayer 格式输出相机，无法保存为该格
式
    FILE_PNG = 8, //PNG 24bit
    FILE_BMP_8BIT = 16,//BMP 8bit
    FILE_PNG_8BIT = 32, //PNG 8bit
    FILE_RAW_16BIT = 64
};

enum
{
    STATISTICS_AVG = 0,//均值统计
    STATISTICS_HIST,//直方图的统计方式
    STATISTICS_WINDOW,//带权重 3x3 的统计方式
    STATISTICS_REFER_WINDOW,//参考窗口的统计方式
```

```
};

typedef enum
{
    PARAM_MODE_BY_MODEL = 0,   //根据相机型号名从文件中加载参数，例如 CK-U300
    PARAM_MODE_BY_NAME,        //根据设备昵称(tDevInfo.acFriendlyName)从文件中加载参数，
                                                    例如 CKCamera_3M,该昵称可自定义
    PARAM_MODE_BY_SN,          //根据设备的唯一序列号从文件中加载参数,序列号在出厂时已经
                               写入设备，每台相机拥有不同的序列号。
    PARAM_MODE_IN_DEVICE       //从设备的固态存储器中加载参数。不是所有的型号都支持从相机
                               中读写参数组，由 tSdkCameraCapbility.bParamInDevice 决定
}emSdkParameterMode;

说明：相机的配置参数，分为 A,B,C,D 4 组进行保存。
typedef enum
{
    PARAMETER_TEAM_DEFAULT = 0xff,
    PARAMETER_TEAM_A = 0,
    PARAMETER_TEAM_B = 1,
    PARAMETER_TEAM_C = 2,
    PARAMETER_TEAM_D = 3
}emSdkParameterTeam;

typedef enum
{
    TRIGGER_MODE_CONTINUOUS = 0,
    TRIGGER_MODE_SOFT,
    TRIGGER_MODE_HARD,
}emTriggerMode;

#define CAMERA_MEDIA_TYPE_MONO                  0x01000000
#define CAMERA_MEDIA_TYPE_RGB                   0x02000000
#define CAMERA_MEDIA_TYPE_COLOR                 0x02000000
#define CAMERA_MEDIA_TYPE_OCCUPY1BIT            0x00010000
#define CAMERA_MEDIA_TYPE_OCCUPY2BIT            0x00020000
#define CAMERA_MEDIA_TYPE_OCCUPY4BIT            0x00040000
#define CAMERA_MEDIA_TYPE_OCCUPY8BIT            0x00080000
#define CAMERA_MEDIA_TYPE_OCCUPY10BIT           0x000A0000
#define CAMERA_MEDIA_TYPE_OCCUPY12BIT           0x000C0000
#define CAMERA_MEDIA_TYPE_OCCUPY16BIT           0x00100000
#define CAMERA_MEDIA_TYPE_OCCUPY24BIT           0x00180000
#define CAMERA_MEDIA_TYPE_OCCUPY32BIT           0x00200000

#define CAMERA_FORMAT_RGB                       0x00000014
```

```
#define CAMERA_FORMAT_BGR                                   0x00000015

#define CAMERA_MEDIA_TYPE_COLOR_MASK                        0xFF000000
#define CAMERA_MEDIA_TYPE_EFFECTIVE_PIXEL_SIZE_MASK         0x00FF0000
#define CAMERA_MEDIA_TYPE_ID_MASK                           0x0000FFFF
#define CAMERA_MEDIA_TYPE_MEDIA_FORMAT                      0x000000FF
/*Bayer */
#define CAMERA_MEDIA_TYPE_BAYGR8              (CAMERA_MEDIA_TYPE_MONO |
CAMERA_MEDIA_TYPE_OCCUPY8BIT | 0x0008)
#define CAMERA_MEDIA_TYPE_BAYRG8              (CAMERA_MEDIA_TYPE_MONO |
CAMERA_MEDIA_TYPE_OCCUPY8BIT | 0x0009)
#define CAMERA_MEDIA_TYPE_BAYGB8              (CAMERA_MEDIA_TYPE_MONO |
CAMERA_MEDIA_TYPE_OCCUPY8BIT | 0x000A)
#define CAMERA_MEDIA_TYPE_BAYBG8              (CAMERA_MEDIA_TYPE_MONO |
CAMERA_MEDIA_TYPE_OCCUPY8BIT | 0x000B)

/*RGB */
#define CAMERA_MEDIA_TYPE_RGB8                (CAMERA_MEDIA_TYPE_COLOR |
CAMERA_MEDIA_TYPE_OCCUPY24BIT | CAMERA_FORMAT_RGB)
#define CAMERA_MEDIA_TYPE_BGR8                (CAMERA_MEDIA_TYPE_COLOR |
CAMERA_MEDIA_TYPE_OCCUPY24BIT | CAMERA_FORMAT_BGR)
#define CAMERA_MEDIA_TYPE_RGBA8               (CAMERA_MEDIA_TYPE_COLOR |
CAMERA_MEDIA_TYPE_OCCUPY32BIT | CAMERA_FORMAT_RGB)
#define CAMERA_MEDIA_TYPE_BGRA8               (CAMERA_MEDIA_TYPE_COLOR |
CAMERA_MEDIA_TYPE_OCCUPY32BIT | CAMERA_FORMAT_BGR)
```

3.3 接口返回值定义（错误码解析）

```
#define VTDEVICE_IOCTLERR                   2
#define VTDEVICE_NOOPENDEVICEERR            3
#define VTDEVICE_RETURNERR                  4       //返回出错
#define VTDEVICE_UNKNOWSENSOR               5
#define VTDEVICE_NOOPENSTREAM               6
#define VTDEVICE_READERROR                  7
#define VTDEVICE_BUFERR                     8
#define VTDEVICE_READFRAMEERR               9
#define VTDEVICE_FINDLOSTDEV                10      //找到失联设备
#define VTDEVICE_DISFINDLOSTDEV             11      //未找到失联设备
#define VTDEVICE_MATCH_OK                   12      //配对成功
#define VTDEVICE_MATCH_FAIL                 13      //配对失败
#define VTDEVICE_DECODE_OK                  14      //解密成功
#define VTDEVICE_DECODE_FAIL                15      //解密失败
```

```
#define VTDEVICE_OPENDEVICEERROR                16
#define VTDEVICE_OPENSTREAMERROR                17
#define VTDEVICE_DECODE_ESN_FAIL                18      //解密 SN 失败
#define VTDEVICE_MODULE_DECODE_FAIL             19      //模块解密失败
#define VTDEVICE_WRITE_FAIL                     20      //写 EEPROM 失败
#define VTDEVICE_NO_WRITE_EEPROM                21      //没有写 EE 数据
#define VTDEVICE_IOCPERR                        22      //完成端口出错
#define VTDEVICE_IOQUEUEERR                     23      //队列出错
#define VTDEVICE_DEVICEISOPENERR                24    //设备已被打开
#define VTDEVICE_NODEVICE                       25
#define VTDEVICE_DEVICELOST                     26
#define VTDEVICE_NO_FILE_ACCESS                 27      //没有找到文件
#define VTDEVICE_EEPROM_UPDATA                  28      //EE 数据版本需要更新


#define VTIMG_NOERR                             0
#define VTIMG_UNKNOWERR                         0x100
#define VTIMG_OPENERR                           0x102
#define VTIMG_NODEVICE                          0x103
#define VTIMG_UNKNOWMODE                        0x104

#define CAMERA_STATUS_SUCCESS                    0   // 操作成功
#define CAMERA_STATUS_FAILED                    -1   // 操作失败
#define CAMERA_STATUS_INTERNAL_ERROR            -2   // 内部错误
#define CAMERA_STATUS_UNKNOW                    -3   // 未知错误
#define CAMERA_STATUS_NOT_SUPPORTED             -4   // 不支持该功能
#define CAMERA_STATUS_NOT_INITIALIZED           -5   // 初始化未完成
#define CAMERA_STATUS_PARAMETER_INVALID         -6   // 参数无效
#define CAMERA_STATUS_PARAMETER_OUT_OF_BOUND    -7   // 参数越界
#define CAMERA_STATUS_UNENABLED                 -8   // 未使能
#define CAMERA_STATUS_USER_CANCEL               -9   // 用户手动取消了，比如 roi 面板点击
取消，返回
#define CAMERA_STATUS_PATH_NOT_FOUND            -10  // 注册表中没有找到对应的路径
#define CAMERA_STATUS_SIZE_DISMATCH             -11  // 获得图像数据长度和定义的尺寸不
匹配
#define CAMERA_STATUS_TIME_OUT                  -12  // 超时错误
#define CAMERA_STATUS_IO_ERROR                  -13  // 硬件 IO 错误
#define CAMERA_STATUS_COMM_ERROR                -14  // 通讯错误
#define CAMERA_STATUS_BUS_ERROR                 -15  // 总线错误
#define CAMERA_STATUS_NO_DEVICE_FOUND           -16  // 没有发现设备
#define CAMERA_STATUS_NO_LOGIC_DEVICE_FOUND     -17  // 未找到逻辑设备
#define CAMERA_STATUS_DEVICE_IS_OPENED          -18  // 设备已经打开
#define CAMERA_STATUS_DEVICE_IS_CLOSED          -19  // 设备已经关闭
#define CAMERA_STATUS_DEVICE_VEDIO_CLOSED       -20  // 没有打开设备视频，调用录像相关的
```

```
                                                函数时，如果相机视频没有打开，则回返回该错误。
#define CAMERA_STATUS_NO_MEMORY                 -21  // 没有足够系统内存
#define CAMERA_STATUS_FILE_CREATE_FAILED        -22  // 创建文件失败
#define CAMERA_STATUS_FILE_INVALID              -23  // 文件格式无效
#define CAMERA_STATUS_WRITE_PROTECTED           -24  // 写保护，不可写
#define CAMERA_STATUS_GRAB_FAILED               -25  // 数据采集失败
#define CAMERA_STATUS_LOST_DATA                 -26  // 数据丢失，不完整
#define CAMERA_STATUS_EOF_ERROR                 -27  // 未接收到帧结束符
#define CAMERA_STATUS_BUSY                      -28  // 正忙(上一次操作还在进行中)，此次
                                                        操作不能进行
#define CAMERA_STATUS_WAIT                      -29  // 需要等待(进行操作的条件不成立)，
                                                        可以再次尝试
#define CAMERA_STATUS_IN_PROCESS                -30  // 正在进行，已经被操作过
#define CAMERA_STATUS_IIC_ERROR                 -31  // IIC 传输错误
#define CAMERA_STATUS_SPI_ERROR                 -32  // SPI 传输错误
#define CAMERA_STATUS_USB_CONTROL_ERROR         -33  // USB 控制传输错误
#define CAMERA_STATUS_USB_BULK_ERROR            -34  // USB BULK 传输错误
#define CAMERA_STATUS_SOCKET_INIT_ERROR         -35  // 网络传输套件初始化失败
#define CAMERA_STATUS_GIGE_FILTER_INIT_ERROR    -36  // 网络相机内核过滤驱动初始化失败，
                                                        请检查是否正确安装了驱动，或者重新安装。
#define CAMERA_STATUS_NET_SEND_ERROR            -37  // 网络数据发送错误
#define CAMERA_STATUS_DEVICE_LOST               -38  // 与网络相机失去连接，心跳检测超时
#define CAMERA_STATUS_DATA_RECV_LESS            -39  // 接收到的字节数比请求的少
#define CAMERA_STATUS_FUNCTION_LOAD_FAILED      -40  // 从文件中加载程序失败
#define CAMERA_STATUS_CRITICAL_FILE_LOST        -41  // 程序运行所必须的文件丢失。
#define CAMERA_STATUS_SENSOR_ID_DISMATCH        -42  // 固件和程序不匹配，原因是下载了错
                                                        误的固件。
#define CAMERA_STATUS_OUT_OF_RANGE              -43  // 参数超出有效范围。
#define CAMERA_STATUS_REGISTRY_ERROR            -44  // 安装程序注册错误。请重新安装程
                                                        序，或者运行安装目录 Setup/Installer.exe
#define CAMERA_STATUS_ACCESS_DENY               -45  // 禁止访问。指定相机已经被其他程序
                                                        占用时，再申请访问该相机，会返回该状态。
                                                        (一个相机不能被多个程序同时访问)
#define CAMERA_STATUS_CAMERA_NEED_RESET         -46  // 表示相机需要复位后才能正常使用，
此时请让相机断电重启，或者重启操作系统后，
                                                        便可正常使用。
#define CAMERA_STATUS_ISP_MOUDLE_NOT_INITIALIZED -47 // ISP 模块未初始化
#define CAMERA_STATUS_ISP_DATA_CRC_ERROR        -48  // 数据校验错误
#define CAMERA_STATUS_MV_TEST_FAILED            -49  // 数据测试失败
#define CAMERA_STATUS_INTERNAL_ERR1             -50
```

4.SDK 接口函数说明

本章节描述 SDK 中各接口函数的原型及说明，针对 C/C++ 可直接按本手册的函数名进行调用。

4.1 获得相机设备列表

对于相机的工作流程，我们采用先枚举、初始化，然后退出程序前反初始化相机。

4.1.1 枚举函数

```
函数名    : CameraEnumerateDevice
功能描述 : 枚举设备，并建立设备列表。在调用 CameraInit
             之前，必须调用该函数来获得设备的信息。
参数      :   pDeviceNum         设备的个数指针，
             函数返回时，保存实际找到的设备个数。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
           否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraEnumerateDevice(INT *pDeviceNum)
```

通过该函数获得当前查询到的所有设备，返回实际找到的设备个数。

4.1.2 获得设备列表的信息

```
函数名    : CameraGetEnumIndexInfo
功能描述 : 获得指定设备的信息
参数      : CameraIndex 相机标号，通过 CameraEnumerateDevice 获得
           pDevInfo 指针，返回设备的信息。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
           否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraGetEnumIndexInfo(INT CameraIndex,tDevInfo* pDevInfo)
```

该函数通过 CameraEnumerateDevice 获得设备列表的个数，确定相机标号，从而查询相应设备信息，描述信息以结极体的方式返回，比如

相机的名称、型号、序列号、接口类型等都会详细列出；

4.2 相机初始化、反初始化及相机能力级的获取

4.2.1 相机初始化函数

相机的初始化我们提供了 CameraInit、 CameraInitEx、CameraInitEx2 三个函数。这三个函数都需要和 CameraEnumerateDevice 配套使用。

```
函数名    : CameraInit
功能描述 : 相机初始化。初始化成功后，才能调用任何其他
          相机相关的操作接口。
参数      :    pCameraHandle  相机的句柄指针，初始化成功后，该指针
                返回该相机的有效句柄，在调用其他相机
               相关的操作接口时，都需要传入该句柄，主要
                用于多相机之间的区分。
               CameraIndex 相机标号，通过 CameraEnumerateDevice 获得
               设备个数，从而设置要打开的相机标号，选取(0-DeviceNum-1)
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
           否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraInit(PHANDLE phCamera, INT CameraIndex)
```

CameraInit 只要传入相机的序号 ID 即可,例如初始化第 1 个相机，传入 0，第二个相机传入 1，以此类推；

```
函数名    : CameraInitEx
功能描述 : 相机初始化。初始化成功后，才能调用任何其他
           相机相关的操作接口。
参数      :    pCameraHandle  相机的句柄指针，初始化成功后，该指针
                                返回该相机的有效句柄，在调用其他相机
                                相关的操作接口时，都需要传入该句柄，主要
                                用于多相机之间的区分。
               CameraIndex 相机标号，通过 CameraEnumerateDevice 获得
                                设备个数，从而设置要打开的相机标号，选取
                                (0-DeviceNum-1)
               iParamLoadMode  相机初始化时使用的参数加载方式。-1 表示使用上次退出时
                               的参数加载方式。
```

为 PARAM_MODE_BY_MODEL 表示按型号加载
为 PARAM_MODE_BY_SN 表示按序列号加载
为 PARAM_MODE_BY_NAME 表示按昵称加载
详细请参开 CameraDefine.h 中 emSdkParameterMode 定义。

emTeam 初始化时使用的参数组。-1 表示加载上次退出时的参数组。

返回值 : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraInitEx
        (PHANDLE phCamera, INT CameraIndex, INT iParamLoadMode, INT emTeam)
```

CameraInitEx 不仅要传入相机的序号 ID ，而且还要传入参数加载模式和组别，初始化加载相应参数，当 iParamLoadMode 和 emTeam 都位-1 时，按照上一次退出时的参数进行加载；

函数名 : CameraInitEx2
功能描述 : 相机初始化。初始化成功后，才能调用任何其他
相机相关的操作接口。注意需要先调用 CameraEnumerateDevice 枚举相机
参数 : CameraName 相机名称
pCameraHandle 相机的句柄指针，初始化成功后，该指针
返回该相机的有效句柄，在调用其他相机
相关的操作接口时，都需要传入该句柄，主要
用于多相机之间的区分。
pFriendlyName 相机昵称，用于找到相同昵称的相机，字节长度最大为 32 个字节
返回值 : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraInitEx2(PHANDLE phCamera, char *pFriendlyName)
```

CameraInitEx2 不需要传入相机的序列 ID，只需要传入相机的名称，由于出厂时相机的昵称都是一样的，需要事先通过 CameraSetFriendlyName 函数将相机名称改为"Camera1"，CameraInitEx2 调用时，传入"Camera1"即可，对于多相机同时工作的案例，这个方法可以有效的建立一一对应的关系，但是要注意，每个相机的名称必须改成不同的，确保唯一性。

4.2.2 相机反初始化函数

函数名 : CameraUnInit

功能描述 ： 相机反初始化。释放资源。
参数　　： hCamera 相机的句柄，由 CameraInit 函数获得。
返回值　： 成功时，返回 CAMERA_STATUS_SUCCESS (0);
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraUnInit(HANDLE hCamera)
```

无论采用哪种初始化方式，反初始化调用 CameraUnInit 即可。

4.2.3 相机能力级的获取函数

函数名　： CameraGetCapability
功能描述 ： 获得相机的特性描述结构体。该结构体中包含了相机
　　可设置的各种参数的范围信息。决定了相关函数的参数
　　返回，也可用于动态创建相机的配置界面。
参数　　： hCamera　相机的句柄，由 CameraInit 函数获得。
　　pCameraInfo 指针，返回该相机特性描述的结构体。
　　tSdkCameraCapbility 在 CameraTypeDef.h 中定义。
返回值　： 成功时，返回 CAMERA_STATUS_SUCCESS (0);
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus __stdcall CameraGetCapability
(HANDLE hCamera, tSdkCameraCapbility* pCameraInfo)
```

4.2.4 获得指定设备的枚举信息函数

函数名　： CameraGetEnumInfo
功能描述 ： 获得指定设备的枚举信息
参数　　： hCamera 相机的句柄，由 CameraInit 函数获得。
　　pCameraInfo 指针，返回设备的枚举信息。
返回值　： 成功时，返回 CAMERA_STATUS_SUCCESS (0);
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetEnumInfo(HANDLE  hCamera, tDevInfo* DevInfo)
```

4.3 获取原始 RAW 数据及 RAW 数据的释放

4.3.1 获得原始 RAW 数据函数

函数名　： CameraGetRawImageBuffer
功能描述 ： 获得一帧图像原始数据。为了提高效率，SDK 在图像抓取时采用了零拷贝机制，
　　CameraGetRawImageBuffer 实际获得是内核中的一个缓冲区地址，
　　该函数成功调用后，必须调用 CameraReleaseFrameHandle,以便让内核继续使用

```
        该缓冲区。
参数      : hCamera    相机的句柄，由 CameraInit 函数获得。
            pHBuf    获得图像的数据的句柄。
            UINT mTimes 抓取图像的超时时间。单位毫秒。在
            mTimes 时间内还未获得图像，则该函数
            会返回超时信息。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考 CKDeviceStatus.h
            中错误码的定义。
CKSDK_API CameraSdkStatus CameraGetRawImageBuffer
          (HANDLE hCamera, PHANDLE pHBuf, UINT mTimes)
```

一般情况下，工业相机传输的都是原始的 RAW 数据，对于黑白相机，就是灰度图像数据，对于彩色相机，就是 Bayer 格式的图像数据。CameraGetRawImageBuffer 函数，得到的就是原始的 RAW 数据，默认是 8bit 。

该函数可以设置超时时间，例如设置 1000 毫秒的超时时间，则在 1000 毫秒内，如果没有获取到有效图像，则该函数会阻塞，线程会被挂起，直到超过 1000 毫秒或者读到了有效图像，所以在软件结构的设计上，用户可以创建一个专门采集图像的线程，然后一直进行图像采集，只需要设定一个合理的超时时间即可，或者在需要采集图像的时候调用一次函数，获取一张图像。

注意:CameraGetRawImageBuffer 需要和 CameraReleaseFrameHandle 配套使用。

4.3.2 RAW 数据的释放

```
函数名    : CameraReleaseImageBuffer
功能描述 : 释放由 CameraGetRawImageBuffer 获得的缓冲区。
参数      : hCamera    相机的句柄，由 CameraInit 函数获得。
            pBuf    通过 CameraGetRawImageBuffer 指向图像的数据的缓冲区句柄。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
```

```
CKSDK_API CameraSdkStatus CameraReleaseFrameHandle(HANDLE hCamera, HANDLE hBuf)
```

注意：为了提高效率，SDK 在图像抓取时采用了零拷贝机制，CameraGetRawImageBuffer 实际获得是内核中的一个缓冲区地址，该函数成功调用后，必须调用 CameraReleaseFrameHandle 释放由 CameraGetRawImageBuffer 得到的缓冲区,以便让内核继续使用该缓冲区。

4.4 获得图像数据

```
函数名    : CameraGetImageInfo
功能描述 : 获得当前帧图像信息
参数      : hCamera    相机的句柄，由 CameraInit 函数获得。
            pBuf    通过 CameraGetRawImageBuffer 获得图像的数据的缓冲区句柄。
           info     获得帧信息
返回值    : 成功时，返回帧数据的缓冲区地址
CKSDK_API BYTE * CameraGetImageInfo(HANDLE hCamera, HANDLE hBuf, stImageInfo *info)
```

通过调用 CameraGetRawImageBuffer 获得一帧图像句柄，通过该句柄获得图像数据缓冲区地址，以及帧头信息。

初始化完成之后，就可以开始相机取图工作。 按照取图方式，SDK 分为主动和被动（回调函数）两种。

1.主动取图方式

在完成相机初始化后工作，可以使用 CameraGetRawImageBuffer 函数和 CameraGetImageInfo 函数主动读取图像，二者的区别是，CameraGetRawImageBuffer 函数得到的是原始的 RAW 数据缓冲区，需要使用 CameraGetOutImageBuffer 函数将 RAW 数据转换为指定的

数据格式，在使用完得到的缓冲区指针后，要调用CameraReleaseFrameHandle 释放缓冲区使用权，需要解释的是CameraReleaseFrameHandle 只是释放由 CameraGetRawImageBuffer 得到的数据缓冲区的使用权，并不会反复申请和释放内存，无需担心效率问题；具体的格式由输入的参数决定，可以是 8 位灰度，也可以是 24 位、 32 位彩色格式数据。

2.被动读取方式（回调函数）

如果需要使用回调函数进行读图处理，则需要在相机初始化以后设置好读图的回调函数。

4.4.1 读取图像函数

```
函数名    : CameraGetOutImageBuffer
功能描述 : 将获得的相机原始输出图像数据进行处理，叠加饱和度、
           颜色增益和校正、降噪等处理效果，处理完后图像的输出格式
           8 位灰度，24 位、32 位彩色图像
参数      : hCamera   相机的句柄，由 CameraInit 函数获得。
           info   输入图像的帧头信息，处理完成后，帧头信息
           中的图像格式 uiMediaType 会随之改变。
           pbuffer     输入图像数据的缓冲区地址，不能为 NULL。
           pRGBbuffer    处理后图像输出的缓冲区地址，不能为 NULL。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
           否则返回非 0 值的错误码,请参考错误码的定义。
    CKSDK_API CameraSdkStatus CameraGetOutImageBuffer
(HANDLE hCamera, stImageInfo *info, BYTE *pbuffer, BYTE *pRGBbuffer)
```

4.4.2 设置图像捕捉的回调方式

```
函数名    : CameraSetCallbackFunction
功能描述 : 设置图像捕获的回调函数。当捕获到新的图像数据帧时，
           pCallBack 所指向的回调函数就会被调用。
参数      : hCamera 相机的句柄，由 CameraInit 函数获得。
           pCallBack 回调函数指针。
           lpParam   回调函数的附加参数，在回调函数被调用时
```

该附加参数会被传入，可以为 NULL。多用于
多个相机时携带附加信息。
pCallbackOld 用于保存当前的回调函数。可以为 NULL。
返回值 : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSetCallbackFunction
(HANDLE hCamera, CAMERA_SNAP_PROC pCallBack, LPVOID lpParam, CAMERA_SNAP_PROC
*pCallbackOld)
```

4.4.3 直接获得输出图像函数

函数名 : CameraGetImageBufferEx
功能描述 : 获得一帧图像数据。该接口获得的图像是经过处理后的格式。该格式可以通过 CameraSetIspOutFormat 设置该函数调用后，不需要调用 CameraReleaseImageBuffer 释放，也不要调用 free 之类的函数释放该函数返回的图像数据缓冲区,适用于主动方式调用。
参数 : hCamera 相机的句柄，由 CameraInit 函数获得。
info 结构体，返回帧头信息
UINT wTimes 抓取图像的超时时间。单位毫秒。在
wTimes 时间内还未获得图像，则该函数
会返回超时信息。
返回值 : 成功时，返回 RGB 数据缓冲区的首地址;
否则返回 NULL。

```
CKSDK_API BYTE* CameraGetImageBufferEx(HANDLE hCamera, stImageInfo *info,UINT wTimes)
```

注意：区别于 CameraGetRawImageBuffer 函数，该函数得到的就是按输出格式输出图像数据。 后续不需要再调用 CameraGetOutImageBuffer 函数和 CameraReleaseImageBuffer 函数。

4.5 利用显示控件预览图像

为了简化用户二次开发的流程，SDK 内部封装了一些接口函数可以方便可以进行图像预览，前提是必须使用 VS 相关的工具进行界面开发。在相机初始化后，调用如下代码完成显示部分的初始化工作。CameraDisplayInit 传入窗口的 hWnd 句柄。 不同的开发工具，有

不同的方式获得显示控件的 hWnd 句柄。

4.5.1 初始化 SDK 内部的显示模块

```
函数名    : CameraDisplayInit
功能描述 : 初始化 SDK 内部的显示模块。在调用 CameraDisplay
           前必须先调用该函数初始化。如果您在二次开发中，
           使用自己的方式进行图像显示(不调用 CameraDisplay)，
           则不需要调用本函数。
参数      : hCamera    相机的句柄，由 CameraInit 函数获得。
           hWndDisplay 显示窗口的句柄，一般为窗口的 m_hWnd 成员。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
           否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraDisplayInit(HANDLE hCamera, HWND hWndDisplay)
```

注意：在调用 CameraDisplay 前必须先调用该函数初始化。如果您在二次开发中，使用自己的方式进行图像显示(不调用 CameraDisplay)，则不需要调用 CameraDisplayInit 函数 。

4.5.2 显示图像函数

```
函数名    : CameraDisplay
功能描述 : 显示图像。必须调用过 CameraDisplayInit 进行
       初始化才能调用本函数。
参数      : hCamera  相机的句柄，由 CameraInit 函数获得。
           pFrameBuffer 图像的数据缓冲区，可以是 CAMERA_MEDIA_TYPE_MONO
           CAMERA_MEDIA_TYPE_BGR8 CAMERA_MEDIA_TYPE_BGRA8 的其中一种。
           pFrameInfo  图像的帧头信息。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
           否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraDisplay
(HANDLE hCamera, BYTE* pFrameBuffer, stImageInfo* pFrameInfo)
```

4.5.3 设置显示模式函数

```
函数名    : CameraSetDisplayMode
功能描述 : 设置显示的模式。必须调用过 CameraDisplayInit
       进行初始化才能调用本函数。
参数      : hCamera  相机的句柄，由 CameraInit 函数获得。
```

iMode　　显示模式，DISPLAYMODE_SCALE 或者
DISPLAYMODE_REAL,具体参见 CKDeviceDef.h
中 emSdkDisplayMode 的定义。
返回值　：成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSetDisplayMode(HANDLE hCamera, int mode)
```

4.5.4 设置指定十字线函数

函数名　：CameraSetCrossLine
功能描述：设置指定十字线的参数。
参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。
iLine　　表示要设置第几条十字线的状态。范围为[0,8]，共 9 条。
x　　十字线中心位置的横坐标值。
y　　十字线中心位置的纵坐标值。
uColor　　十字线的颜色，格式为(R|(G<<8)|(B<<16))
bVisible　十字线的显示状态。TRUE，表示显示。
只有设置为显示状态的十字线，在调用
CameraImageOverlay 后才会被叠加到图像上。
返回值　：成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSetCrossLine
(HANDLE hCamera, INT iLine,INT x,INT y,UINT uColor,BOOL bVisible)
```

4.5.5 获得指定十字线函数

函数名　：CameraGetCrossLine
功能描述：获得指定十字线的状态。
参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。
iLine　　表示要获取的第几条十字线的状态。范围为[0,8]，共 9 条。
px　　指针，返回该十字线中心位置的横坐标。
py　　指针，返回该十字线中心位置的横坐标。
pcolor　　指针，返回该十字线的颜色，格式为(R|(G<<8)|(B<<16))。
pbVisible　指针，返回 TRUE，则表示该十字线可见。
返回值　：成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetCrossLine
(HANDLE hCamera, INT iLine, INT *px, INT *py, UINT *puColor, BOOL *pbVisible)
```

4.5.6 图像数据的叠加函数

函数名　：CameraImageOverlay

功能描述 ： 将输入的图像数据上叠加十字线、白平衡参考窗口、
自动曝光参考窗口等图形。只有设置为可见状态的
十字线和参考窗口才能被叠加上。
注意，该函数的输入图像必须是 RGB 格式。
参数 ： hCamera 相机的句柄，由 CameraInit 函数获得。
pRgbBuffer 图像数据缓冲区。
pFrInfo 图像的帧头信息。
返回值 ： 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraImageOverlay
    (HANDLE hCamera,BYTE* pRgbBuffer, stImageInfo* pFrInfo)
```

4.5.7 设置显示回调函数

函数名 ： CameraSetDisplayCallbackFun
功能描述 ： 设置显示回调。必须调用过 CameraDisplayInit 进行
初始化才能调用本函数，同时当显示图像数据帧时，
pCallBack 所指向的回调函数就会被调用。
参数 ： hCamera 相机的句柄，由 CameraInit 函数获得。
pCallBack 回调函数指针。
lpParam 回调函数的附加参数，在回调函数被调用时
该附加参数会被传入，可以为 NULL。
pCallbackOld 用于保存当前的回调函数。可以为 NULL。
返回值 ： 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSetDisplayCallbackFun(HANDLE hCamera,
CAMERA_DISPLAY_PROC pCallBack, LPVOID lpParam, CAMERA_DISPLAY_PROC *pCallbackOld)
```

4.5.8 设置绘制文字函数

函数名 ： CameraDrawText
功能描述 ： 在输入的图像数据中绘制文字
参数 ： pRgbBuffer 图像数据缓冲区
pFrInfo 图像的帧头信息
pFontFileName 字体文件名
FontWidth 字体宽度
FontHeight 字体高度
pText 要输出的文字
(Left, Top, Width, Height) 文字的输出矩形
TextColor 文字颜色 RGB
uFlags 输出标志,详见 emCameraDrawTextFlags 中的定义
返回值 ： 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraDrawText
    (BYTE* pRgbBuffer, stImageInfo*  pFrInfo,
    char const*       pFontFileName,
    UINT          FontWidth,
    UINT          FontHeight,
    char const*       pText,
    INT               Left,
    INT               Top,
    UINT          Width,
    UINT          Height,
    UINT          TextColor,
    UINT          uFlags
)
```

4.6 相机的播放与暂停

4.6.1 播放函数

```
函数名    : CameraPause
功能描述 : 让 SDK 进入暂停模式，不接收来自相机的图像数据，
           同时也会发送命令让相机暂停输出，释放传输带宽。
           暂停模式下，可以对相机的参数进行配置，并立即生效。
参数      : hCamera    相机的句柄，由 CameraInit 函数获得。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
           否则返回非 0 值的错误码,请参考错误码的定义。
    CKSDK_API CameraSdkStatus CameraPause(HANDLE hCamera)
```

4.6.2 暂停函数

```
函数名    : CameraPlay
功能描述 : 让 SDK 进入工作模式，开始接收来自相机发送的图像
           数据。如果当前相机是触发模式，则需要接收到
           触发帧以后才会更新图像。
参数      : hCamera    相机的句柄，由 CameraInit 函数获得。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
           否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraPlay(HANDLE hCamera)
```

4.7 相机曝光功能的设置

相机出厂时，为了方便演示效果，默认是自动曝光，也就是说，相机会根据环境光的亮度，自动调节相机的曝光时间和模拟增益值来获取最佳的图像亮度。但是这种自动调节模式，并不适合工业应用，因此需要调用一些函数，来手动设置曝光和增益，来稳定图像亮度，用以适合软件算法的需求，例如一些情况，用户可能希望图像大部分区域过曝光以减少图像干扰。

4.7.1 设置相机曝光模式函数

```
函数名    : CameraSetAeState
功能描述 : 设置相机曝光的模式。自动或者手动。
参数      : hCamera    相机的句柄，由 CameraInit 函数获得。
            AeState     TRUE，使能自动曝光；FALSE，停止自动曝光。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraSetAeState(HANDLE hCamera, bool AeState)
```

4.7.2 获得曝光模式函数

```
函数名    : CameraGetAeState
功能描述 : 获得相机当前的曝光模式。
参数      : hCamera   相机的句柄，由 CameraInit 函数获得。
            pAeState    指针，用于返回自动曝光的使能状态。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考 CKDeviceStatus.h
            中错误码的定义。
CKSDK_API CameraSdkStatus CameraGetAeState(HANDLE hCamera, bool *pAeState)
```

4.7.3 设定自动曝光的亮度目标值函数

```
函数名    : CameraSetAeTarget
功能描述 : 设定自动曝光的亮度目标值。设定范围由 CameraGetCapability
       函数获得。
参数      : hCamera  相机的句柄，由 CameraInit 函数获得。
            AeTarget  亮度目标值。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考 CKDeviceStatus.h
            中错误码的定义。
CKSDK_API CameraSdkStatus CameraSetAeTarget(HANDLE hCamera, WORD AeTarget)
```

4.7.4 获得自动曝光的亮度目标值函数

```
函数名    : CameraGetAeTarget
功能描述 : 获得自动曝光的亮度目标值。
参数      : hCamera   相机的句柄，由 CameraInit 函数获得。
            *piAeTarget 指针，返回目标值。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考 CKDeviceStatus.h
            中错误码的定义。
CKSDK_API CameraSdkStatus __stdcall CameraGetAeTarget
(HANDLE hCamera, WORD *pAeTarget)
```

4.7.5 设置曝光时间函数

```
函数名    : CameraSetExposureTime
功能描述 : 设置曝光时间。单位为微秒。对于 CMOS 传感器，其曝光
       的单位是按照行来计算的，因此，曝光时间并不能在微秒
       级别连续可调。而是会按照整行来取舍。在调用
       本函数设定曝光时间后，建议再调用 CameraGetExposureTime
       来获得实际设定的值。
参数      : hCamera      相机的句柄，由 CameraInit 函数获得。
            fExposureTime 曝光时间，单位微秒。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考 CKDeviceStatus.h
            中错误码的定义。
CKSDK_API CameraSdkStatus CameraSetExposureTime
(HANDLE hCamera, double fExposureTime)
```

该函数设置相机的曝光时间，单位为微秒。曝光时间越大，相机的帧

率越低，例如曝光时间设置到 500 毫秒，则一秒相机最多成像 2 次，图像看上去就会比较卡顿。因此为了减少取图时间，曝先时间应该是要越低越好，当然，曝先时间低，就需要外部光源加大光照，否则图像就会很暗。

注意：该函数必须在手动模式下才可以生效。

4.7.6 获得曝光时间函数

```
函数名    : CameraGetExposureTime
功能描述 : 获得相机的曝光时间。请参见 CameraSetExposureTime
       的功能描述。
参数      : hCamera         相机的句柄，由 CameraInit 函数获得。
            pfExposureTime    指针，返回当前的曝光时间，单位微秒。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考 CKDeviceStatus.h
            中错误码的定义。
CKSDK_API CameraSdkStatus CameraGetExposureTime
(HANDLE hCamera, double *pfExposureTime)
```

4.7.7 设置图像模拟增益函数

```
函数名    : CameraSetAnalogGain
功能描述 : 设置相机的图像模拟增益值。该值乘以 CameraGetCapability 获得
            的相机属性结构体中 sExposeDesc.fAnalogGainStep，就
            得到实际的图像信号放大倍数。
参数      : hCamera    相机的句柄，由 CameraInit 函数获得。
            iAnalogGain 设定的模拟增益值。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraSetAnalogGain(HANDLE hCamera, UINT iAnalogGain)
```

设置相机的模拟增益值，这个值的大小只会影响图像亮度，不会影响图像帧率，因为只是一个电路放大系数，不过该值增大后会提升图像背景噪声，对于追求画质的应用场合，模拟增益应该设置到最小。

注意：该函数必须在手动模式下才可以生效。

4.7.8 获得图像模拟增益函数

函数名　：CameraGetAnalogGain
功能描述：获得图像信号的模拟增益值。参见 CameraSetAnalogGain
　　详细说明。
参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。
　　piAnalogGain 指针，返回当前的模拟增益值。
返回值　：成功时，返回 CAMERA_STATUS_SUCCESS (0);
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetAnalogGain(HANDLE hCamera, UINT *piAnalogGain)
```

4.7.9 设置自动曝光的参考窗口函数

函数名　：CameraSetAeWindow
功能描述：设置自动曝光的参考窗口。
参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。
　　left　　窗口左上角的横坐标
　　top　　窗口左上角的纵坐标
　　width　　窗口的宽度
　　height　　窗口的高度
返回值　：成功时，返回 CAMERA_STATUS_SUCCESS (0);
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSetAeWindow
(HANDLE hCamera, WORD left, WORD top, WORD width, WORD height)
```

4.7.10 获得获得自动曝光参考窗口的位置函数

函数名　：CameraGetWbWindow
功能描述：获得白平衡参考窗口的位置。
参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。
　　left　　指针，返回参考窗口的左上角横坐标 。
　　top　　指针，返回参考窗口的左上角纵坐标 。
　　width　　指针，返回参考窗口的宽度。
　　height　　指针，返回参考窗口的高度。
返回值　：成功时，返回 CAMERA_STATUS_SUCCESS (0);
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetAeWindow
(HANDLE hCamera, WORD *left, WORD *top, WORD *width, WORD *height)
```

4.7.11 设置自动曝光时抗频闪功能的使能状态函数

```
函数名    : CameraSetAntiFlick
功能描述 : 设置自动曝光时抗频闪功能的使能状态。对于手动
        曝光模式下无效。
参数      : hCamera   相机的句柄，由 CameraInit 函数获得。
            bEnable    TRUE，开启抗频闪功能;FALSE，关闭该功能。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraSetAntiFlick(HANDLE hCamera, bool bEnable)
```

4.7.12 获得自动曝光时抗频闪功能的使能状态函数

```
函数名    : CameraGetAntiFlick
功能描述 : 获得自动曝光时抗频闪功能的使能状态。
参数      : hCamera   相机的句柄，由 CameraInit 函数获得。
            pbEnable    指针，返回该功能的使能状态。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraGetAntiFlick(HANDLE hCamera, bool *pbEnable)
```

4.7.13 设置自动曝光时消频闪的频率函数

```
函数名    : CameraSetLightFrequency
功能描述 : 设置自动曝光时消频闪的频率。
参数      : hCamera      相机的句柄，由 CameraInit 函数获得。
            iFrequencySel 1:50HZ , 2:60HZ
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraSetLightFrequency(HANDLE hCamera, INT iFrequencySel)
```

4.7.14 获得获得自动曝光时消频闪的频率函数

```
函数名    : CameraGetLightFrequency
功能描述 : 获得自动曝光时，消频闪的频率选择。
参数      : hCamera      相机的句柄，由 CameraInit 函数获得。
            piFrequencySel 指针，返回选择的索引号。1:50HZ 2:60HZ
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraSetLightFrequency
```

(*HANDLE* hCamera, *INT* *piFrequencySel)

4.8 相机白平衡功能的设置

4.8.1 设置相机白平衡模式函数

函数名 : CameraSetWbMode
功能描述 : 设置相机白平衡模式。分为手动和自动两种方式。
参数 : hCamera 相机的句柄，由 CameraInit 函数获得。
bAuto TRUE，则表示使能自动模式。
FALSE，则表示使用手动模式，通过调用
CameraSetOnceWB 来进行一次白平衡。
返回值 : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSetWbMode(HANDLE hCamera, bool bAuto)
```

4.8.2 获得相机白平衡模式函数

函数名 : CameraGetWbMode
功能描述 : 获得当前的白平衡模式。
参数 : hCamera 相机的句柄，由 CameraInit 函数获得。
pbAuto 指针，返回 TRUE 表示自动模式，FALSE
为手动模式。
返回值 : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetWbMode(HANDLE hCamera, bool *pbAuto)
```

4.8.3 设置图像的数字增益函数

函数名 : CameraSetGain
功能描述 : 设置图像的数字增益。设定范围由 CameraGetCapability
获得的相机属性结构体中 sRgbGainRange 成员表述。
实际的放大倍数是设定值/100。
参数 : hCamera 相机的句柄，由 CameraInit 函数获得。
DGainR 红色通道的增益值。
DGainG 绿色通道的增益值。
DGainB 蓝色通道的增益值。
返回值 : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSetGain
    (HANDLE hCamera, WORD DGainR, WORD DGainG, WORD DGainB)
```

该函数设置 R、 G、 B 三个颜色通道的增益值。

注意：该函数必须在手动模式下才可以生效。

4.8.4 获得图像的数字增益函数

```
函数名    : CameraGetGain
功能描述 : 获得图像处理的数字增益。具体请参见 CameraSetGain
        的功能描述部分。
参数      : hCamera   相机的句柄，由 CameraInit 函数获得。
            pDGainR   指针，返回红色通道的数字增益值。
            pDGainG     指针，返回绿色通道的数字增益值。
            pDGainB     指针，返回蓝色通道的数字增益值。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraGetGain
(HANDLE hCamera, WORD *pDGainR, WORD *pDGainG, WORD *pDGainB)
```

4.8.5 设置一次白平衡函数

```
函数名    : CameraSetOnceWB
功能描述 : 在手动白平衡模式下，调用该函数会进行一次白平衡。
            生效的时间为接收到下一帧图像数据时。
参数      : hCamera   相机的句柄，由 CameraInit 函数获得。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraSetOnceWB(HANDLE hCamera)
```

注意：该函数必须在手动模式下才可以生效。

4.8.6 设置白平衡参考窗口的位置函数

```
函数名    : CameraSetWbWindow
功能描述 : 设置白平衡参考窗口的位置。
参数      : hCamera 相机的句柄，由 CameraInit 函数获得。
            left    参考窗口的左上角横坐标。
            top     参考窗口的左上角纵坐标。
            width     参考窗口的宽度。
            height    参考窗口的高度。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
```

```
CKSDK_API CameraSdkStatus CameraSetWbWindow
    (HANDLE hCamera, WORD left, WORD top, WORD width, WORD height)
```

4.8.7 获得白平衡参考窗口的位置函数

```
函数名    : CameraGetWbWindow
功能描述 : 获得白平衡参考窗口的位置。
参数      : hCamera   相机的句柄，由 CameraInit 函数获得。
            left    指针，返回参考窗口的左上角横坐标 。
            top     指针，返回参考窗口的左上角纵坐标 。
            width    指针，返回参考窗口的宽度。
            height   指针，返回参考窗口的高度。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraGetWbWindow
    (HANDLE hCamera, WORD *left, WORD *top, WORD *width, WORD *height)
```

4.9 相机 ISP 功能的设置

4.9.1 设置图像饱和度函数

```
函数名    : CameraSetSaturation
功能描述 : 设定图像处理的饱和度。对黑白相机无效。
            设定范围由 CameraGetCapability 获得。100 表示
            表示原始色度，不增强。
参数      : hCamera     相机的句柄，由 CameraInit 函数获得。
            iSaturation   设定的饱和度值。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
    CKSDK_API CameraSdkStatus CameraSetSaturation(HANDLE hCamera, INT iSaturation)
```

该函数设置图像饱和度。饱和度越大色彩越浓；反之越淡，饱和度如果设置为 0，图像就完全没有色彩了，等效于黑白相机。默认值是 100。调节范围是 0 到 200。

4.9.2 获得图像饱和度函数

```
函数名    : CameraGetSaturation
```

```
功能描述 : 获得图像处理的饱和度。
参数     : hCamera     相机的句柄，由 CameraInit 函数获得。
           piSaturation 指针，返回当前图像处理的饱和度值。
返回值   : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
           否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraGetSaturation(HANDLE hCamera, INT *piSaturation)
```

4.9.3 设置图像对比度函数

```
函数名   : CameraSetContrast
功能描述 : 设定图像处理的对比度
参数     : hCamera  相机的句柄，由 CameraInit 函数获得。
           iContrast  设定的对比度值。
返回值   : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
           否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraSetContrast(HANDLE hCamera, INT iContrast)
```

该函数设置相机的对比度值，对比度越大，会使图像黑的区域越黑，白的区域越白，使得图像看起来黑白分明，在一些视觉处理上可以有效的捕捉轮廓；反之，如果对比度越小，会是黑白不分明，看起来比较朦胧。对比度默认是 50，最小可以到 1，最大到 100。

4.9.4 获得图像对比度函数

```
函数名   : CameraGetContrast
功能描述 : 获得对比度值。请参考
       CameraSetContrast 函数的功能描述。
参数     : hCamera  相机的句柄，由 CameraInit 函数获得。
           piContrast 指针，返回当前的对比度值。
返回值   : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
           否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraGetContrast(HANDLE hCamera, INT *piContrast)
```

4.9.5 设置图像锐度函数

```
函数名   : CameraSetSharpness
功能描述 : 设置图像的处理的锐化参数。
```

参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。
　　iSharpness 锐化参数。范围由 CameraGetCapability
　　获得，一般是[0,100]，0 表示关闭锐化处理。
返回值　：成功时，返回 CAMERA_STATUS_SUCCESS (0);
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSetSharpness(HANDLE hCamera, INT iSharpness)
```

该函数设置图像的锐化级别。锐化度越高，图像清晰度越好，但是噪声也会越大；反之锐化度越低，图像朦胧感就强，但是噪声降低，显得平滑。默认值是 0，就是没有锐化增强的效果。

4.9.6 获得图像锐度函数

函数名　：CameraGetSharpness
功能描述：获取当前锐化设定值。
参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。
　　piSharpness 指针，返回当前设定的锐化的设定值。
返回值　：成功时，返回 CAMERA_STATUS_SUCCESS (0);
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetSharpness(HANDLE hCamera, INT *piSharpness)
```

4.9.7 设置图像分辨率函数

函数名　：CameraSetResolution
功能描述：设定相机输出图像的分辨率。
参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。
　　iResolutionIndex 选择的帧率模式索引号,相机可供选择的分辨率模式由
　　CameraGetCapability 获得的信息结构体中 iImageSizeDesc
　　表示可以选择的分辨率模式个数。
返回值　：成功时，返回 CAMERA_STATUS_SUCCESS (0);
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSetResolution(HANDLE hCamera, INT iResolutionIndex)
```

4.9.8 获得图像分辨率函数

函数名　：CameraGetResolution
功能描述：获得相机输出图像的分辨率。
参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。
　　piResolutionIndex 指针，返回选择的帧率模式索引号,请参见 emResolutionMode。
返回值　：成功时，返回 CAMERA_STATUS_SUCCESS (0);

否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetResolution(HANDLE hCamera, INT *piResolutionIndex)
```

4.9.9 设置自定义预览的分辨率函数

函数名 : CameraSetImageResolution
功能描述 : 设置自定义预览的分辨率。
参数 : hCamera 相机的句柄，由 CameraInit 函数获得。
pImageResolution 结构体指针，用于返回当前的分辨率。
返回值 : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSetResolutionEx
        (HANDLE hCamera,tSdkImageResolution* pImageResolution)
```

4.9.10 获得索引号的分辨率函数

函数名 : CameraGetResolutionEx
功能描述 : 获得索引号的分辨率。
参数 : hCamera 相机的句柄，由 CameraInit 函数获得。
iResolution 获得分辨率的索引号，参见 emResolutionMode，自定义为 0xff
pImageResolution 结构体指针，用于返回当前的分辨率。
返回值 : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetResolutionEx
    (HANDLE hCamera, int iResolution, tSdkImageResolution* pImageResolution)
```

4.9.11 获得当前预览的分辨率函数

函数名 : CameraGetCurResolution
功能描述 : 获得当前预览的分辨率。
参数 : hCamera 相机的句柄，由 CameraInit 函数获得。
pImageResolution 结构体指针，用于返回当前的分辨率。
返回值 : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus  CameraGetCurResolution
    (HANDLE hCamera, tSdkImageResolution* pImageResolution)
```

4.9.12 设置图像帧速度函数

函数名 : CameraSetFrameSpeed
功能描述 : 设定相机输出图像的帧率。相机可供选择的帧率模式由
CameraGetCapability 获得的信息结构体中 iFrameSpeedDesc

```
        表示最大帧率选择模式个数。
参数      : hCamera    相机的句柄，由 CameraInit 函数获得。
            iFrameSpeed 选择的帧率模式索引号，范围从 0 到
              CameraGetCapability 获得的信息结构体中 iFrameSpeedDesc - 1
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraSetFrameSpeed(HANDLE hCamera, INT framespeed)
```

通过 CameraSetFrameSpeed 函数，可以动态设置相机的输出帧率。CameraSetFrameSpeed 是设置速度的档位，我们将其分为高速、中速、低速等几种模式，对于不同型号的相机，高中低速度模式下对应具体的帧率是不一样的。

CameraSetFrameSpeed（hCamera,0）;//讴置为低速模式。

CameraSetFrameSpeed（hCamera,1）;//讴置为中速模式。

CameraSetFrameSpeed（hCamera,2）;//讴置为高速模式。

一般相机出厂时，默认就是最高速度的，如果需要降速运行，就可以调用 CameraSetFrameSpeed 函数迕行降速。 另外，有些型号的相机，另支持 2 个速度模式，有些支持 4 个，具体支持模式的数量，可以通过 CameraGetCapability 函数，得到相机的描述信息 tSdkCameraCapbility 结极体，tSdkCameraCapbility 结极体中的 iFrameSpeedDesc 成员，表明了当前型号的相机支持的速度模式的个数。

4.9.13 获得图像帧速度函数

```
函数名    : CameraGetFrameSpeed
功能描述 : 获得相机输出图像的帧率选择索引号。具体用法参考
           CameraSetFrameSpeed 函数的功能描述部分。
参数      : hCamera    相机的句柄，由 CameraInit 函数获得。
            piFrameSpeed 指针，返回选择的帧率模式索引号。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
```

否则返回非 0 值的错误码,请参考错误码的定义。

```
TSDK_API CameraSdkStatus CameraGetFrameSpeed(HANDLE hCamera, INT *pframespeed)
```

4.9.14 获得当前图像帧函数

```
函数名    : CameraGetFrame
功能描述 : 获得当前图像帧时间。
参数      : hCamera  相机的句柄，由 CameraInit 函数获得。
            frametime 指针，返回当前的帧时间，单位微秒。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus __stdcall CameraGetFrame(HANDLE hCamera, double *frametime)
```

4.9.15 获得相机接收帧的统计信息

```
函数名    : CameraGetFrameStatistic
功能描述 : 获得相机接收帧率的统计信息，包括错误帧和丢帧的情况。
参数      : hCamera        相机的句柄，由 CameraInit 函数获得。
            psFrameStatistic 指针，返回统计信息。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraGetFrameStatistic
    (HANDLE hCamera, FrameStatistic *psFrameStatistic)
```

4.9.16 设置图像输出格式函数

```
函数名    : CameraSetIspOutFormat
功能描述 : 设置 CameraImageProcess 函数的图像处理的输出格式，支持
            8 位灰度图像和 24RGB、32 位 RGB、24 位 BGR、32 位 BGR 彩色图像。
            默认输出是 24 位 BGR 格式。
参数      :  hCamera        相机的句柄，由 CameraInit 函数获得。
             uFormat    要设定格式。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraSetIspOutFormat(HANDLE hCamera, INT format)
```

该函数是最终通过 CameraGetOutImageBuffer 函数的得到的图像像素格式，而并非相机输出的原始 RAW 的像素格式。

在相机初始化以后，通过调用 CameraSetIspOutFormat 来设置了。

1.CameraSetIspOutFormat(hCamera, CAMERA_MEDIA_TYPE_MONO);

设置后，CameraGetOutImageBuffer 输出的图像就是 8bit 的灰度图像了，1 个像素占用 1 个字节，依次排列。

2.CameraSetIspOutFormat(hCamera, CAMERA_MEDIA_TYPE_RGB8);

设置后，CameraGetOutImageBuffer 输出的图像就是 24bit RBG 的彩色图像了，1 个像素占用 3 个字节，依次是红色、 绿色、 蓝色返样排列。

3.CameraSetIspOutFormat(hCamera, CAMERA_MEDIA_TYPE_BGR8);

设置后，CameraGetOutImageBuffer 输出的图像就是 24bit BGR 的彩色图像了，1 个像素占用 3 个字节，依次是蓝色、 绿色、 红色排列。

4.CameraSetIspOutFormat(hCamera, CAMERA_MEDIA_TYPE_RGBA8);

设置后，CameraGetOutImageBuffer 输出的图像就是 32bit RBGA 的彩色图像了，1 个像素占用 4 个字节，依次是红色 8bit、 绿色 8bit、 蓝色 8bit、 Alpha8bit 排列。

5.CameraSetIspOutFormat(hCamera, CAMERA_MEDIA_TYPE_BGRA8);

设置后，CameraGetOutImageBuffer 输出的图像就是 32bit BGRA 的彩色图像了，1 个像素占用 3 个字节，依次是蓝色、 绿色、 红色、 Alpha 排列。

4.9.17 获得图像输出格式函数

函数名 ： CameraGetIspOutFormat
功能描述 ： 获得 CameraGetImageBuffer 函数图像处理的输出格式，支持
8 位灰度图像和 24RGB、32 位 RGB、24 位 BGR、32 位 BGR 彩色图像。
参数 ： hCamera 相机的句柄，由 CameraInit 函数获得。
puFormat 返回当前设定的格式。

返回值　：成功时，返回 CAMERA_STATUS_SUCCESS (0)；
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetIspOutFormat(HANDLE hCamera, INT *pformat)
```

4.9.18 获得图像保存成图片的函数

函数名　：CameraSaveImage
功能描述：将图像缓冲区的数据保存成图片文件。
参数　　：hCamera　　相机的句柄，由 CameraInit 函数获得。
　　lpszFileName　图片保存文件完整路径。
　　pbyImageBuffer 图像的数据缓冲区。
　　pFrInfo　　图像的帧头信息。
　　byFileType　　图像保存的格式。取值范围参见 CameraDefine.h
　　中 emSdkFileType 的类型定义。目前支持
　　BMP、RAW 两种种格式。其中 RAW 表示
　　相机输出的原始数据，保存 RAW 格式文件要求
　　pbyImageBuffer 和 pFrInfo 是由 CameraGetImageBuffer
　　获得的数据，而且未经 CameraImageProcess 转换
　　成 BMP 格式；反之，如果要保存成 BMP 格式，则 pbyImageBuffer 和 pFrInfo 是
　　由 CameraImageProcess 处理后的 RGB 格式数据。
　　byQuality　　图像保存的质量因子，仅当保存为 JPG 格式时该参数有效，范
　　围 1 到 100。其余格式可以写成 0。
返回值　：成功时，返回 CAMERA_STATUS_SUCCESS (0)；
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSaveImage
    (HANDLE hCamera, char* lpszFileName, BYTE* pbyImageBuffer,
    stImageInfo*  pFrInfo, UINT byFileType, BYTE byQuality)
```

图像保存功能。使用 CameraSaveImage，可以将图像保存为 RAW、BMP24bit 的其中一种。一个典型的流程为：

1，使用 CameraGetRawImageBuffer 函数和 CameraGetImageInfo 函数取到 RAW 图像数据；

2，使用 CameraGetOutImageBuffer 函数将 RAW 转换成 BGR24 或者 8 位灰度格式。如果要保存 RAW 格式数据，返一步跳过。

3，CameraReleaseFrameHandle，释放 CameraGetRawImageBuffer 的到的 RAW 数据缓冲区的使用权。

4，CameraSaveImage 函数将 CameraGetOutImageBuffer 函数得到的图像数据进行保存成 BMP 。如果是 RAW 数据保存则使用 CameraGetRawImageBuffer 函数和 CameraGetImageInfo 函数得到的 RAW 数据 buffer。

4.9.19 设置图像水平、垂直镜像的函数

```
函数名    : CameraSetMirror
功能描述 : 设置图像镜像操作。镜像操作分为水平和垂直两个方向。
参数      : hCamera   相机的句柄，由 CameraInit 函数获得。
            iDir      表示镜像的方向。0，表示水平方向；1，表示垂直方向。
            bEnable   TRUE，使能镜像;FALSE，禁止镜像
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraSetMirror(HANDLE hCamera, int iDir, bool bEnable)
```

4.9.20 获得图像水平、垂直镜像的函数

```
函数名    : CameraGetMirror
功能描述 : 获得图像的镜像状态。
参数      : hCamera   相机的句柄，由 CameraInit 函数获得。
            iDir      表示要获得的镜像方向。
             0，表示水平方向；1，表示垂直方向。
            pbEnable    指针，返回 TRUE，则表示 iDir 所指的方向
             镜像被使能。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraGetMirror(HANDLE hCamera, int iDir, bool *pbEnable)
```

4.9.21 设置彩色转为黑白功能的函数

```
函数名    : CameraSetMonochrome
功能描述 : 设置彩色转为黑白功能的使能。
参数      : hCamera 相机的句柄，由 CameraInit 函数获得。
            bEnable    TRUE，表示将彩色图像转为黑白。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraSetMonochrome(HANDLE hCamera, bool bEnable)
```

4.9.22 获得彩色转为黑白功能的函数

函数名　　：CameraGetMonochrome
功能描述：获得彩色转换黑白功能的使能状况。
参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。
　　pbEnable　指针。返回 TRUE 表示开启了彩色图像
　　转换为黑白图像的功能。
返回值　　：成功时，返回 CAMERA_STATUS_SUCCESS (0);
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetMonochrome(HANDLE hCamera, bool *pbEnable)
```

4.9.23 设置图像的黑电平函数

函数名　　：CameraSetBlackLevel
功能描述：设置图像的黑电平基准，默认值为 0
参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。
　　iBlackLevel 要设定的电平值。范围为 0 到 255。
返回值　　：成功时，返回 CAMERA_STATUS_SUCCESS (0);
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSetBlackLevel(HANDLE hCamera,INT iBlackLevel)
```

4.9.24 获得图像的黑电平函数

函数名　　：CameraGetBlackLevel
功能描述：获得图像的黑电平基准，默认值为 0
参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。
　　piBlackLevel 返回当前的黑电平值。范围为 0 到 255。
返回值　　：成功时，返回 CAMERA_STATUS_SUCCESS (0);
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetBlackLevel(HANDLE hCamera,INT* piBlackLevel)
```

4.10 相机 Gamma 功能的设置

4.10.1 设置相机的查表变换模式函数

函数名　　：CameraSetLutMode
功能描述：设置相机的查表变换模式 LUT 模式。
参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。
　　emLutMode　LUTMODE_PARAM_GEN 表示由伽马和对比度参数动态生成 LUT 表。
　　LUTMODE_PRESET　表示使用预设的 LUT 表。
　　LUTMODE_USER_DEF　表示使用用户自定的 LUT 表。

LUTMODE_PARAM_GEN 的定义参考 CameraDefine.h 中 emSdkLutMode 类型。

返回值 ： 成功时，返回 CAMERA_STATUS_SUCCESS (0)；
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSetLutMode(HANDLE hCamera, INT emLutMode)
```

4.10.2 获得相机的查表变换模式函数

函数名 ： CameraGetLutMode
功能描述 ： 获得相机的查表变换模式 LUT 模式。
参数 ： hCamera 相机的句柄，由 CameraInit 函数获得。
pemLutMode 指针，返回当前 LUT 模式。意义与 CameraSetLutMode 中 emLutMode 参数相同。
返回值 ： 成功时，返回 CAMERA_STATUS_SUCCESS (0)；
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetLutMode(HANDLE hCamera, INT *pemLutMode)
```

4.10.3 设置 LUT 动态生成模式下的 Gamma 函数

函数名 ： CameraSetGamma
功能描述 ： 设定 LUT 动态生成模式下的 Gamma 值。设定的值会马上保存在 SDK 内部，但是只有当相机处于动态参数生成的 LUT 模式时，才会生效。请参考 CameraSetLutMode 的函数说明部分。
参数 ： hCamera 相机的句柄，由 CameraInit 函数获得。
lowlight 要设定的低亮度阈值。
highlight 要设定的高亮度阈值。
lowgama 要设定的低亮度对应的 gamma 值。
midgama 要设定的中亮度对应的 gamma 值。
highgama 要设定的高亮度对应的 gamma 值。
lowlightout 要设定的输出低亮度阈值。
highlightout 要设定的输出低亮度阈值。
返回值 ： 成功时，返回 CAMERA_STATUS_SUCCESS (0)；
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSetGamma(HANDLE hCamera, INT lowlight, INT highlight,
INT lowgama, INT midgama, INT highgama, INT lowlightout, INT highlightout)
```

4.10.4 获得 LUT 动态生成模式下的 Gamma 函数

函数名 ： CameraGetGamma
功能描述 ： 获得 LUT 动态生成模式下的 Gamma 值。请参考 CameraSetGamma 函数的功能描述。
参数 ： hCamera 相机的句柄，由 CameraInit 函数获得。

piGamma 指针，返回当前的 Gamma 值。
plowlight 指针，返回当前的低亮度阈值。
phighlight 指针，返回当前的高亮度阈值。
plowgama 指针，返回当前的低亮度对应的 gamma 值。
pmidgama 指针，返回当前的中亮度对应的 gamma 值。
phighgama 指针，返回当前的高亮度对应的 gamma 值。
plowlightout 指针，返回当前的输出低亮度阈值。
phighlightout 指针，返回当前的输出低亮度阈值。
返回值 : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetGamma(HANDLE hCamera, INT *plowlight, INT *phighlight, INT *plowgama, INT *pmidgama, INT *phighgama, INT *plowlightout, INT *phighlightout)
```

4.10.5 选择预设 LUT 模式下的 LUT 表函数

函数名 : CameraSelectLutPreset
功能描述 : 选择预设 LUT 模式下的 LUT 表。必须先使用 CameraSetLutMode 将 LUT 模式设置为预设模式。
参数 : hCamera 相机的句柄，由 CameraInit 函数获得。
iSel 表的索引号。表的个数由 CameraGetCapability 获得。
返回值 : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSelectLutPreset(HANDLE hCamera, INT iSel)
```

4.10.6 获得预设 LUT 模式下的 LUT 表索引号函数

函数名 : CameraGetLutPresetSel
功能描述 : 获得预设 LUT 模式下的 LUT 表索引号。
参数 : hCamera 相机的句柄，由 CameraInit 函数获得。
piSel 指针，返回表的索引号。
返回值 : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetLutPresetSel(HANDLE hCamera, INT *piSel)
```

4.10.7 设置自定义的 LUT 表函数

函数名 : CameraSetCustomLut
功能描述 : 设置自定义的 LUT 表。必须先使用 CameraSetLutMode

将 LUT 模式设置为自定义模式。
参数　　：　hCamera　相机的句柄，由 CameraInit 函数获得。
　　pLut　　指针，指向 LUT 表的地址。LUT 表为无符号短整形数组，数组大小为 4096，分别代码颜色通道从 0 到 4096(12bit 颜色精度)对应的映射值。
返回值　　：　成功时，返回 CAMERA_STATUS_SUCCESS (0);
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSetCustomLut(HANDLE hCamera, BYTE* pLut)
```

4.10.8 获得当前使用的自定义 LUT 表函数

函数名　　：CameraGetCustomLut
功能描述：获得当前使用的自定义 LUT 表。
参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。
　　pLut　　指针，指向 LUT 表的地址。LUT 表为无符号短整形数组，数组大小为 4096，分别代码颜色通道从 0 到 4096(12bit 颜色精度)对应的映射值。
返回值　　：成功时，返回 CAMERA_STATUS_SUCCESS (0);
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus __stdcall CameraGetCustomLut(HANDLE hCamera, BYTE* pLut)
```

4.11 相机触发功能设置

4.11.1 设置相机触发模式函数

函数名　　：CameraSetTriggerMode
功能描述：设置相机的触发模式。
参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。
　　iTriggerModeSel　　模式选择索引号。可设定的模式由 CameraGetCapability 函数获取。请参考 CameraDefine.h 中 tSdkCameraCapbility 的定义。一般情况，0 表示连续采集模式；1 表示软件触发模式；2 表示硬件触发模式。
返回值　　：成功时，返回 CAMERA_STATUS_SUCCESS (0);
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSetTriggerMode(HANDLE hCamera, INT iTriggerModeSel)
```

4.11.2 获得相机的触发模式函数

函数名　　：CameraGetTriggerMode
功能描述：获得相机的触发模式。
参数　　：hCamera　相机的句柄，由 CameraInit 函数获得。

```
            piTriggerModeSel  指针，返回当前选择的相机触发模式的索引号。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraGetTriggerMode(HANDLE hCamera, INT*piTriggerModeSel)
```

4.11.3 执行一次软触发函数

```
函数名    : CameraSoftTrigger
功能描述 : 执行一次软触发。
参数      : hCamera  相机的句柄，由 CameraInit 函数获得。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraSoftTrigger(HANDLE hCamera)
```

相机出厂默认就处于连续取图模式，也可以通过 CameraSetTriggerMode(hCamera,0);切换为连续取图模式。

1. 设置相机为软触发取图模式。通过 CameraSetTriggerMode(hCamera,1);切换为软触发取图模式。进入该模式后，相机停止图像采集和发送。 只有当用户调用 CameraSoftTrigger(hCamera)一次，相机就采集一次图像发送上来。发完之后相机又会进入等待状态，直到下次用户调用 CameraSoftTrigger(hCamera)。

2. 设置相机为硬触发取图模式。通过 CameraSetTriggerMode(hCamera,2);切换为硬触发取图模式。进入该模式后，相机停止图像采集和发送。 只有当用户在相机外壳上的触发端子上输入一个脉冲时，相机就采集一次图像发送上来。发完之后相机又会进入等待状态，直到下次收到触发脉冲。

4.12 检测相机掉线与自动重连

在一些极端环境下，相机可能存在掉线的风险，例如 USB 相机在电脑主机供电不稳定的时候，或者震动比较大，USB 口松动等等。 默认情况下，都有自动掉线重连的功能，SDK 内部也已经集成了该功能。

4.13 相机参数的保存与加载

4.13.1 设置参数存取的目标对象函数

```
函数名    : CameraSetParameterMode
功能描述 : 设定参数存取的目标对象。
参数      : hCamera  相机的句柄，由 CameraInit 函数获得。
            iMode  参数存取的模式。参考 CameraDefine.h
          中 emSdkParameterMode 的类型定义。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraSetParameterMode(HANDLE hCamera, INT iMode)
```

4.13.2 获得参数存取的目标对象函数

```
函数名    : CameraGetParameterMode
功能描述 : 获得参数存取的目标对象。
参数      : hCamera  相机的句柄，由 CameraInit 函数获得。
             int* piMode 指针，返回参数存取的模式
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraGetParameterMode(HANDLE hCamera, INT* piMode)
```

4.13.3 保存当前相机参数到指定的参数组函数

```
函数名    : CameraSaveParameter
功能描述 : 保存当前相机参数到指定的参数组中。相机提供了 A,B,C,D
       A,B,C,D 四组空间来进行参数的保存。
参数      : hCamera  相机的句柄，由 CameraInit 函数获得。
            iTeam      PARAMETER_TEAM_A 保存到 A 组中,
             PARAMETER_TEAM_B 保存到 B 组中,
```

PARAMETER_TEAM_C 保存到 C 组中,
PARAMETER_TEAM_D 保存到 D 组中
返回值 : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSaveParameter(HANDLE hCamera,INT iTeam)
```

4.13.4 保存当前相机参数到指定的文件中函数

函数名 : CameraSaveParameterToFile
功能描述 : 保存当前相机参数到指定的文件中。该文件可以复制到
别的电脑上供其他相机加载，也可以做参数备份用。
参数 : hCamera 相机的句柄，由 CameraInit 函数获得。
sFileName 参数文件的完整路径。
返回值 : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSaveParameterToFile
    (HANDLE  hCamera,char* sFileName)
```

4.13.5 从 PC 上指定的参数文件中加载参数函数

函数名 : CameraReadParameterFromFile
功能描述 : 从 PC 上指定的参数文件中加载参数。我公司相机参数
保存在 PC 上为.bin 后缀的文件，位于安装下的
\config 文件夹中。
参数 : hCamera 相机的句柄，由 CameraInit 函数获得。
*sFileName 参数文件的完整路径。
返回值 : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraReadParameterFromFile
    (HANDLE hCamera,char* sFileName)
```

4.13.6 加载指定组的参数到相机函数

函数名 : CameraLoadParameter
功能描述 : 加载指定组的参数到相机中。
参数 : hCamera 相机的句柄，由 CameraInit 函数获得。
iTeam PARAMETER_TEAM_A 加载 A 组参数,
PARAMETER_TEAM_B 加载 B 组参数,
PARAMETER_TEAM_C 加载 C 组参数,
PARAMETER_TEAM_D 加载 D 组参数,

PARAMETER_TEAM_DEFAULT 加载默认参数。
类型定义参考 CameraDefine.h 中 emSdkParameterTeam 类型

返回值 ：成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraLoadParameter(HANDLE hCamera, INT iTeam)
```

4.13.7 获得当前选择的参数组函数

函数名 ：CameraGetCurrentParameterGroup
功能描述 ：获得当前选择的参数组。
参数 ：hCamera 相机的句柄，由 CameraInit 函数获得。
piTeam 指针，返回当前选择的参数组。返回值
参考 CameraLoadParameter 中 iTeam 参数。
返回值 ：成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetCurrentParameterGroup(HANDLE hCamera, INT* piTeam)
```

4.13.8 将用户自定义的数据保存到相机的非易性存储器函数

函数名 ：CameraSaveUserData
功能描述 ：将用户自定义的数据保存到相机的非易性存储器中。
每个型号的相机可能支持的用户数据区最大长度不一样。
可以从设备的特性描述中获取该长度信息。
参数 ：hCamera 相机的句柄，由 CameraInit 函数获得。
uStartAddr 起始地址，从 0 开始。
pbData 数据缓冲区指针
ilen 写入数据的长度，ilen + uStartAddr 必须
小于用户区最大长度
返回值 ：成功时，返回 CAMERA_STATUS_SUCCESS (0);
否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraSaveUserData
    (HANDLE hCamera, UINT uStartAddr, BYTE *pbData, INT ilen)
```

4.13.9 从相机的非易性存储器中读取用户自定义的数据函数

函数名 ：CameraLoadUserData
功能描述 ：从相机的非易性存储器中读取用户自定义的数据。

```
                每个型号的相机可能支持的用户数据区最大长度不一样。
                可以从设备的特性描述中获取该长度信息。
参数      : hCamera     相机的句柄，由 CameraInit 函数获得。
            uStartAddr  起始地址，从 0 开始。
            pbData      数据缓冲区指针，返回读到的数据。
            ilen        读取数据的长度，ilen + uStartAddr 必须
                        小于用户区最大长度
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraLoadUserData
    (HANDLE hCamera,UINT uStartAddr,BYTE *pbData,INT ilen)
```

我公司所有型号的相机，都带一段数据存储空间，用户可以自由读写。并且掉电后也会保存，不会丢失数据。每个型号的相机，存储空间的大小是不一样的，具体的大小可以通过 CameraGetCapability 函数得到，tSdkCameraCapbility 结极体的 iUserDataMaxLen 表示相机的最大存储字节数。

4.14 相机序列号的设置

相机自带三级序列号，其中一级序列号是另读的，不可以修改，出厂的时候已经固定好了。二级和三级序列号可自由读写。每级序列号都是 32 个字节。

4.14.1 设置相机的序列号函数

```
函数名    : CameraWriteSN
功能描述 : 设置相机的序列号。我公司相机序列号分为 3 级。
        0 级的是我公司自定义的相机序列号，出厂时已经
        设定好，1 级和 2 级留给二次开发使用。每级序列
        号长度都是 32 个字节。
参数      : hCamera   相机的句柄，由 CameraInit 函数获得。
            pbySN     序列号的缓冲区。
            iLevel    要设定的序列号级别，只能是 1 或者 2。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
```

```
CKSDK_API CameraSdkStatus CameraWriteSN(HANDLE hCamera,BYTE* pbySN,UINT iLevel)
```

4.14.2 获得相机的序列号函数

```
函数名    : CameraReadSN
功能描述 : 读取相机指定级别的序列号。序列号的定义请参考
          CameraWriteSN 函数的功能描述部分。
参数      : hCamera   相机的句柄，由 CameraInit 函数获得。
            pbySN     序列号的缓冲区。
            iLevel      要读取的序列号级别。只能是 1 和 2。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraReadSN(HANDLE hCamera,BYTE* pbySN,INT iLevel)
```

4.14.3 设置用户自定义的设备昵称函数

```
函数名    : CameraGetFriendlyName
功能描述 : 读取用户自定义的设备昵称。
参数      : hCamera     相机的句柄，由 CameraInit 函数获得。
            pName       指针，返回指向 0 结尾的字符串，
              设备昵称不超过 32 个字节，因此该指针
              指向的缓冲区必须大于等于 32 个字节空间。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraGetFriendlyName(HANDLE  hCamera,char* pName)
```

4.14.4 获得用户自定义的设备昵称函数

```
函数名    : CameraSetFriendlyName
功能描述 : 设置用户自定义的设备昵称。
参数      : hCamera     相机的句柄，由 CameraInit 函数获得。
            pName       指针，指向 0 结尾的字符串，
              设备昵称不超过 32 个字节，因此该指针
              指向字符串必须小于等于 32 个字节空间。
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus CameraSetFriendlyName(HANDLE  hCamera,char* pName)
```

4.15 多相机使用对应方法

当视觉系统中同时使用多个相机时，在软件上面就需要确认确认好相机和检测工位之间的一一对应关系，但是相机的扫描顺序往往是不固定的，和相机的上电顺序有关系，因此需要在软件端进行一些特殊的识别。我们按照以下方式进行绑定：

按照相机的自定义名称。 默认情况下，在枚举阶段就被枚举出来的，也是通过 CameraEnumerateDevice 函数可以扫描到相机的个数，调用 CameraInitEx2 函数，通过传入的名称，进行相应设备初始化。

4.16 相机版本信息

4.16.1 获得 SDK 版本信息函数

```
函数名    : CameraSdkGetVersionString
功能描述 : 获得 SDK 版本信息
参数      : pVersionString 指针，返回 SDK 版本字符串。
                          该指针指向的缓冲区大小必须大于
                          32 个字节
返回值    : 成功时，返回 CAMERA_STATUS_SUCCESS (0);
            否则返回非 0 值的错误码,请参考错误码的定义。
CKSDK_API CameraSdkStatus  CameraSdkGetVersionString(char* pVersionString)
```

4.16.2 检测固件版本函数

```
函数名    : CameraCheckFwUpdate
功能描述 : 检测固件版本，是否需要升级。
参数      : hCamera 相机的句柄，由 CameraInit 函数获得。
            pNeedUpdate 指针，返回固件检测状态，TRUE 表示需要更新
```

返回值　：成功时，返回 CAMERA_STATUS_SUCCESS (0)；
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraCheckFwUpdate(HANDLE  hCamera,BOOL* pNeedUpdate)
```

4.16.3 获得设备版本函数

函数名　：CameraGetFirmwareVision
功能描述：获得固件版本的字符串
参数　　：hCamera 相机的句柄，由 CameraInit 函数获得。
　　pVersion 必须指向一个大于 32 字节的缓冲区，
　　返回固件的版本字符串。
返回值　：成功时，返回 CAMERA_STATUS_SUCCESS (0)；
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetFirmwareVision(HANDLE  hCamera,char* pVersion)
```

4.16.4 获得设备接口的版本

函数名　：CameraGetInerfaceVersion
功能描述：获得指定设备接口的版本
参数　　：hCamera 相机的句柄，由 CameraInit 函数获得。
　　pVersion 指向一个大于 32 字节的缓冲区，返回接口版本字符串。
返回值　：成功时，返回 CAMERA_STATUS_SUCCESS (0)；
　　否则返回非 0 值的错误码,请参考错误码的定义。

```
CKSDK_API CameraSdkStatus CameraGetInerfaceVersion(HANDLE  hCamera, char* pVersion)
```

5.开发手册修正时间

1.工业相机开发手册 V1.0 修正时间为 2017.07.04；

2.工业相机开发手册 V2.0 修正时间为 2017.08.04；

2.1 添加了如下功能：

添加功能	函数名称	备注

设置显示模式	CameraSetDisplayMode	
设置十字线功能	CameraSetCrossLine	
图像叠加功能	CameraImageOverlay	
显示回调功能	CameraSetDisplayCallbackFun	
设置自定义分辨率功能	CameraSetResolutionEx	
设置黑电平的功能	CameraSetBlackLevel	
直接获取图像格式数据的函数	CameraGetImageBufferEx	